Improved Issuer Hiding for BBS-based Anonymous Credentials

Nesrine Kaaniche[2], Seyni Kane[1,2], Maryline Laurent[2], and Jacques Traoré[1]

[1] Orange Innovation, Applied Crypto Group 14000 Caen, France.
{seyni.kane, jacques.traore}@orange.com
[2] SAMOVAR, Télécom SudParis, Institut Polytechnique de Paris, France.
{kaaniche.nesrine, maryline.laurent}@telecom-sudparis.eu

Abstract. Attribute-based anonymous credential systems often fail to conceal the identity of the credential issuer. Recent attempts to address this limitation either suffer from efficiency issues or rely on security models that make unrealistic assumptions about the behavior of issuers.
In this paper, we demonstrate that existing security models, which assume honest issuers, are inadequate for real-world deployments. Concretely, we present attack scenarios in which a malicious user collaborates with a single authorized issuer to forge credentials on arbitrary attributes, effectively allowing the user to assume the role of any trusted issuer without any possibility of identifying the misbehaving issuer.
To address these shortcomings, we introduce a stronger security model that explicitly accounts for malicious issuers. We also propose a novel, issuer-hiding, anonymous credential scheme based on the BBS signature scheme (Eurocrypt 2023). This scheme is proven to be secure under the new model using a signed-policy approach. Our construction is fully based on BBS signatures, facilitating integration into existing standards, and resolves several open challenges posed by Katz and Sefranek (PKC 2026): (1) it is proven secure in the Algebraic Group Model (AGM) rather than the Generic Group Model (GGM); (2) it eliminates secret policy keys, enabling verification without any secret values; and (3) it supports delegation of policy generation to a trusted certification authority.
We further observe that issuer hiding inherently creates a tension between privacy and accountability: while concealing the issuer's identity protects honest users, it may grant plausible deniability to malicious issuers that deliberately certify false attributes. To resolve this, we introduce *Accountable Issuer Hiding* (AIH), augmenting traditional issuer hiding with a controlled opening capability. Under AIH, every presentation contains an encryption of the issuer's public key under a designated opening authority's key, bound via zero-knowledge proofs to the credential and the verifier's policy. This ensures that fraudulent presentations remain attributable to the responsible issuer, even when malicious issuers provide users with opaque devices that generate valid presentations on demand.
Finally, we introduce the first pairing-free issuer-hiding anonymous credential scheme, based on algebraic MACs. Formal security proofs and implementation results confirm that our schemes achieve unforgeability, everlasting issuer-hiding anonymity, and accountable issuer hiding, es-

tablishing them as practical and secure solutions for the forthcoming European Digital Identity (EUDI) Wallet.

1 Introduction

Anonymous credentials (ACs) are cryptographic mechanisms that enables users to prove their identity and/or statements about their attributes without revealing unnecessary information. ACs involve three main entities: a user seeking to authenticate or prove statements to a relying party (RP); an issuer providing the user with credentials; and an RP accepting or rejecting the credentials presented by the user.
The concept of anonymous credentials was first introduced by David Chaum in the 1980s [20, 21]. Since then, it has evolved into a rich research domain with numerous constructions and approaches [12, 14, 15, 16, 1, 2, 44, 32, 45, 23]. Despite providing robust user privacy, most traditional AC solutions fail to hide the issuer's identity. This lack of issuer anonymity can lead to critical metadata leakage; for instance, a credential issued by a specific city administration may indirectly expose a user's place of residence.

Recently, several solutions, known as issuer hiding AC, have been proposed to mitigate this problem. However, these approaches do not rely on BBS signatures, which are particularly attractive due to their efficient support for zero-knowledge proofs, selective disclosure, and ongoing standardization efforts (cf. IETF draft [38]).

Katz and Sefranek [36] recently introduced the first BBS-based issuer-hiding AC system, advancing the state of the art by enabling efficient proof generation and practical deployment, but introducing new vulnerabilities through strong trust assumptions. They used the same approach as Sanders and Traoré [46] to hide the credential issuer among a set of authorized issuers based on what is called a Policy[3], and they demonstrated how to achieve issuer hiding, resulting in an efficient construction that satisfies the requirement for everlasting issuer-hiding anonymity and that is unforgeable in the generic group model (GGM). Their solution can be integrated into existing standards and offers several efficiency advantages over prior work. Most importantly, they identified open questions for future research, aimed at designing more efficient and secure issuer-hiding credentials. These include: (1) constructing a BBS-based issuer-hiding anonymous credential that is proven secure in the Algebraic Group Model (AGM), and (2) eliminating the need for policy keys, while maintaining acceptable efficiency, or at least reducing the size of policy keys to a constant.

[3] In the simplest case, the policy only contains the trusted issuer keys. In different approaches, policies may have an associated public key, which is required for the generation and verification of issuer-hiding presentations, and sometimes also a secret counterpart, which is required in addition to the public policy key to verify issuer-hiding presentations. In several schemes, the public policy key is used to certify, using a suitable signature scheme, the trusted issuer keys.

Very recently, Flamini et al. [31] proposed three BBS-based issuer-hiding AC constructions. While the first two are computationally inefficient, the third is more practical. However, as detailed in Appendix Section B.2, this latter construction suffers from a critical vulnerability that makes it unsuitable for real-world deployment.

This work aims to strengthen issuer-hiding guarantees in BBS-based credentials. First it presents several attack scenarios that highlight the limitations of the underlying security model of the latest constructions, specifically those by Katz and Sefranek [36] and Flamini et al. [31], as well as the issuer-hiding ACs proposed by Sanders and Traoré [46] and Bosk et al. [11]. Another limitation of [36], and [46] is that presentation verification is restricted to a designated verifier (RP), thus reducing verification to a secret-key operation. Consequently, the verification policy key, intended to check whether a given issuer's public key satisfies the policy, cannot be generated by a trusted third party, such as a certification authority.
To address these issues, we provide answers to some of the open questions raised by Katz and Sefranek [36]. Specifically, we introduce a new BBS-based issuer-hiding credential system which is (1) proven secure in the Algebraic Group Model (AGM), rather than the Generic Group Model (GGM). Our construction also (2) allows the verification policy key to be generated by a certification authority, thereby removing the need to define policy keys for each RP.

1.1 Related Works

Anonymous credential systems have long been studied as a means to provide privacy-preserving authentication. A crucial property in this context is issuer hiding, which ensures that the RP cannot infer the credential issuer's identity. Early constructions of anonymous credentials, such as those based on Camenisch–Lysyanskaya signatures [14, 15, 16], primarily focused on user privacy and selective disclosure, however issuer hiding was not considered as core requirement. More recent works [5, 6, 11, 23, 28, 41, 46] have introduced formal definitions and practical realizations of the issuer hiding property.

Early approaches to issuer hiding relied on inefficient OR-proofs [24], which allow a user to prove possession of a valid credential from one of several authorized issuers without revealing which one. Initial constructions [5, 23] produced proofs whose size was linear to the number of issuers $|\mathcal{P}|$, while later work [28] reduced this overhead to a logarithmic complexity.

Bobolz et al. [6] formalized issuer hiding in the context of anonymous credentials and proposed a framework based on a signed policy. In this approach, the RP selects a set of authorized public keys and signs each of them using an UF-CMA secure signature scheme, thereby basing the security of the verification policy scheme on the UF-CMA security of the underlying signature scheme. This design simplifies the user's task. Instead of proving possession of a credential for every possible issuer, the user only needs to demonstrate that the credential was issued by one of the keys listed in the policy. This can be achieved by proving knowledge of a signature on their issuer's public key. This results in presentation proofs

of constant size. Bobolz et al. also provided a generic construction and proved the security of this generic construction. Although proving knowledge of both a credential and a signature on a hidden issuer key increases proof complexity, their approach remains relatively simple, relies on standard cryptographic primitives, and the security proofs are based on more standard models and assumptions.

Other schemes achieve efficiency by relaxing privacy guarantees. For example, Bosk et al. [11] require the disclosure of all attributes, thereby compromising attribute privacy to achieve issuer hiding. One of the most recent advances by Sanders and Traoré [46] achieves efficient issuer-hiding with constant-size proofs using algebraic techniques rather than relying on generic zero-knowledge proofs. Their scheme leverages Pointcheval–Sanders (PS) signatures [44], a publicly verifiable variant of keyed-verification credentials [19]. In this design, the RP generates a private policy verification key $\mathrm{sk}_{\mathcal{P}}$ and a public policy key $\mathrm{pk}_{\mathcal{P}}$. A user holding a credential from an issuer within the policy can produce a proof using $\mathrm{pk}_{\mathcal{P}}$, which can be verified using $\mathrm{sk}_{\mathcal{P}}$. The verification of these presentations requires the knowledge of the secret policy key $\mathrm{sk}_{\mathcal{P}}$, which is only held by the RP. This requirement for a secret verification key represents a significant constraint also present in the scheme proposed by Katz and Sefranek [36].
Indeed, Katz and Sefranek demonstrated how to achieve issuer hiding for BBS-based anonymous credentials [36], thereby introducing the first issuer-hiding BBS anonymous credential scheme. They adapted the Sanders and Traoré approach to BBS signatures, resulting in constant-size presentations and shorter public keys than in [46]. The scheme also provides everlasting issuer-hiding anonymity and is proven unforgeable in the generic group model.

Very recently, Flamini et al. [31][4] introduced three BBS-based issuer-hiding AC constructions, called respectively IHBBS_0, IHBBSGroth and IHBBS_1. Unfortunately, IHBBS_0 is inefficient as the size of the presentations is linear in the number of authorized issuers (the size of the policy). Their second scheme uses Groth's signature scheme [35] to certify the public keys of authorized issuers and

[4] The work [31] has recently been split into two separate papers (ePrint 2026/369 and ePrint 2026/870). The Type-0 scheme (IHBBS_0) has been separated and merged into ePrint 2026/870, which now also contains a detailed study of how the linear-size overhead of OR-proof-based presentations can be reduced through more advanced proof techniques, such as Stacking Sigmas and Groth–Kohlweiss proofs. The second paper (ePrint 2026/369) focuses exclusively on the constant-size Type-1 construction (IHBBS_1), and additionally develops a common framework and security model covering all three IHAC types, together with a systematic discussion of their respective privacy and efficiency trade-offs.
The critical vulnerability described in Section B.2 still applies to the Type-1 construction (IHBBS_1) in the revised version of ePrint 2026/369. We emphasize, however, that this is not an attack on the unforgeability of their scheme in the sense of their security model: our attack relies on a corrupt issuer that is part of the policy, a scenario that is explicitly excluded from their unforgeability experiment, which assumes that all issuers behave honestly. Nevertheless, our attacks demonstrate that this security model is not realistic for real-world deployments, such as the European Digital Identity Wallet (EUDI Wallet).

BBS for the issuance of the credentials. Unfortunately, using Groth's signature scheme makes IHBBSGroth quite inefficient, as generating and verifying issuer-hiding presentations involves costly pairing computations. Furthermore, using a non-standard signature scheme (Groth's scheme) for the overall credential scheme adds complexity and delays any real-world deployment requiring proper standardization, such as the forthcoming European digital identity wallet (EUDIW) [29]. Additionally, the security proof of IHBBSGroth relies on the GGM.
Their third construction, IHBBS_1, achieves compact presentations, i.e., not scaling in the number of issuers, does not involve costly computations, and is proven secure in the AGM.

Compared to earlier issuer-hiding schemes, IHBBS_1 [31], and Katz and Sefranek's scheme [36] are highly efficient. Using BBS signatures makes them compatible with existing standardization efforts, facilitating integration into current standards and promoting their widespread adoption in the near future. However, a key limitation of these solutions is that their verification policy mechanism does not rely on a generic approach. This leaves them vulnerable to several attacks related to their policy verification mechanisms. These attacks apply to IHBBS_1 [31] and the scheme of Bosk et al., since both employ a similar technique to verify whether a public key belongs to the policy. Similarly, these attacks also apply to the schemes by Katz and Sefranek [36], and by Sanders and Traoré [46], as they both utilize the same kind of policy verification mechanism. A further limitation of these four schemes is that they all rely on pairings and do not address how to achieve a pairing-free, issuer-hiding, BBS-like credential system.

1.2 Our Contributions

We propose a novel issuer-hiding BBS-based credential scheme that addresses the open questions raised by Katz and Sefranek [36]. Our construction eliminates the requirement for secret policy keys and provides a robust realization of issuer-hiding ACs proven secure in the Algebraic Group Model (AGM). Additionally, we identify several attacks and structural limitations in previous BBS-based designs, specifically demonstrating how existing policy verification mechanisms can be bypassed.

We summarize our contributions as follows:

- We begin by presenting two attack scenarios relating to the choice of security model in the issuer-hiding anonymous credential scheme proposed by Katz and Sefranek [36], which also apply to the construction developed by Sanders and Traoré [46]. Additionally, we propose attacks targeting the issuer-hiding schemes of Bosk et al. [11], and the third construction (IHBBS_1) in [31] by Flamini et al.
- We address the two main open questions raised in [36]: (1) constructing a BBS-based issuer-hiding anonymous credential that is proven secure in the Algebraic Group Model (AGM), and (2) removing the need for secret policy keys entirely, while maintaining acceptable efficiency. To this end, we introduce a new BBS-based issuer-hiding credential system, proven secure

in the AGM, that adopts the signed-policy approach of [6], while avoiding the limitations of prior work. This demonstrates the feasibility of achieving BBS-based IHAC with this approach, in contrast to what is stated in [36]. Our construction enables the delegation of policy generation to a certification authority, removing the need for each RP to maintain its own policy keys. Furthermore, verification does not require knowledge of any secret values, thus eliminating any dependency on policy keys.
- We observe that issuer hiding inherently creates a tension between privacy and accountability: while concealing the issuer's identity protects honest users, it may simultaneously grant plausible deniability to malicious issuers that deliberately certify false attributes. To address this, we introduce the notion of *Accountable Issuer Hiding* (AIH), which reconciles issuer privacy and accountability by allowing the identification of the issuer responsible for a fraudulent presentation under a controlled opening procedure. We provide a practical realization of AIH in which every presentation includes an encryption of the issuer's public key under the public key of a designated opening authority, accompanied by zero-knowledge proofs establishing that the encrypted key both corresponds to the issuer that generated the credential and belongs to the verifier's accepted policy.
- We propose the first pairing-free issuer-hiding anonymous credential scheme by presenting a pairing-free variant of our construction based on *algebraic MACs* and *oblivious issuance of proofs*. This feature is not addressed in the constructions of Flamini et al. [31] and Katz and Sefranek [36].
- We also provide security proofs that our constructions achieve everlasting issuer-hiding anonymity and unforgeability in the AGM. This addresses another open question raised in [36] by establishing that BBS-based issuer-hiding ACs can be proven secure in the AGM.
- We also provide a concrete implementation and a comparison with the schemes of Katz and Sefranek [36], and Flamini et al. [31], demonstrating that our scheme is overall more efficient and secure, and uses a more standard approach, and relies on a more standard approach based on EUF-CMA secure digital signatures to certify the public keys of authorized issuers. Our implementation further confirms the practical viability of our construction, achieving end-to-end presentation verification in under 2ms on a standard PC, while remaining fully compatible with the hardware secure elements available on modern smartphones such as the Android Keystore and the iOS Secure Enclave.

1.3 Technical Overview

In this section, we briefly recall the Katz and Sefranek [36] BBS-based issuer-hiding ACs (See Section 4.1 for more details), and then we provide an overview of our scheme.

Katz and Sefranek BBS-based Issuer-Hiding ACs [36]. Their scheme (See Section 4.1 for more details) is based on BBS (See Section 2.6 for more details

on BBS), where the public key of an issuer is defined as $\tilde{ipk} = \tilde{g}^x$. A credential on attributes $\{m_i\}$ is formed by choosing a random exponent e and computing $A = C_m^{\frac{1}{x+e}}$, where $C_m = h_0 \prod_i h_i^{m_i}$ and $\{h_i\}_i$ are public random generators. The credential is $\sigma = (A, e)$.

Informally, to prove possession of a credential $\sigma = (A, e)$ on attributes $\{m_i\}$ (for simplicity, we assume that all attributes are disclosed) with respect to ipk, the user randomizes his credential with $\bar{A} = A^r$, $\bar{B} = \bar{A}^{-e} C_m^r = \bar{A}^x$, and then sends $\bar{A}, \bar{B}$ along with a zero-knowledge (ZK) proof such that $\bar{B} = \bar{A}^{-e} C_m^r$ for some $e, r \in \mathbb{Z}_p$. Verification checks whether $\mathbf{e}(\bar{A}, pk) = \mathbf{e}(\bar{B}, \tilde{g})$ (We focus on the pairing-based check, and ignore verification of the ZK proof for simplicity). To incorporate the issuer hiding feature into their scheme, Katz and Sefranek use the Sanders and Traoré approach [46]. In this approach, each RP defines his policy, by selecting a set of authorized issuers $\mathcal{P} = \{\tilde{\text{ipk}}_j\}_{j \in [\ell]}$, and defining a secret policy key $\text{sk}_{\mathcal{P}} = (a, b) \leftarrow\!\$\, \mathbb{Z}_p^{*2}$, along with a public policy key $\text{pk}_{\mathcal{P}} = (\tilde{S} = \tilde{g}^a, \pi, \{\tilde{T}_j\}_{j \in [\ell]})$, where $\tilde{T}_j = (\tilde{g}^b \tilde{\text{ipk}}_j)^a, j \in [\ell]$, and π is a proof that $\text{pk}_{\mathcal{P}}$ was correctly computed.

We assume now that a user wants to prove possession of a credential (A, e) on attributes $\{m_i\}$ with respect to some key in a set $\mathcal{P} = \{\tilde{\text{ipk}}_j\}_{i \in [\ell]}$ of ℓ public keys, without disclosing which specific issuer in $\mathcal{P}$ provided the credential. Let $\tilde{\text{ipk}}_{j^*} \in \mathcal{P}$ be the public key for which the user's credential is valid. A natural idea is to replace $\tilde{\text{ipk}}_{j^*}$ in the verification equation with the product of all the public keys $P = \prod_i \tilde{\text{ipk}}_j$. Since with this term the verification will not succeed, Katz and Sefranek instead use a randomized correction term $\tilde{\sigma} = \tilde{S}^t \prod_{j \neq j'} \tilde{T}_j$, with $t \leftarrow\!\$\, \mathbb{Z}_p^*$. To ensure that verification succeeds, the user must also adjust $\bar{B}$ by computing it as $\bar{B} = \bar{A}^{-e-t} C_m^r$, so the RP using his secret key policy $\text{sk}_{\mathcal{P}}$ can check:

$$e(\bar{A}, \tilde{\sigma}^{\frac{-1}{a}} \tilde{g}^{(\ell-1)b} P) = e(\bar{B}, \tilde{g}).$$

Intuitively, incorporating $\tilde{\sigma}^{\frac{1}{a}}$ into the verification equation forces the user to construct $\tilde{\sigma}$ using the provided values $\tilde{S}$ and $\{\tilde{T}_j\}$, while multiplying by $\tilde{g}^{(\ell-1)b}$ forces the user to use exactly $\ell - 1$ of the $\{\tilde{T}_j\}$ when constructing $\tilde{\sigma}$.

Attacks On Previous Issuer-Hiding Constructions. The attacks against the construction in [36] target the presentation policy: if one or more issuers collude with a malicious user, they can enable that user to obtain credentials for arbitrary attributes using self generated issuing keys. Consequently, the user effectively assumes the role of an authorized issuer. This allows the user to issue valid credentials under their own keys, even if those keys are not included in the authorized presentation policy.

For simplicity, we consider the case where a single issuer with secret key $\text{isk}_{j_0} = y_{j_0}$ and public key $\tilde{ipk}_{j_0}$ colludes with the malicious user. To assist the user, the issuer only needs to send him a commitment $\bar{\text{ipk}}_{j_0} = \tilde{S}^{y_{j_0}} = \tilde{g}^{a y_{j_0}}$ on its secret key. The user then generates a secret key $\text{sk} = y \leftarrow\!\$\, \mathbb{Z}_p^*$, and computes a public key $\tilde{\text{pk}} = \tilde{g}^y$, and a value $\bar{\text{pk}} = \tilde{S}^{-y} = \tilde{g}^{-ay}$. With the private key $\text{sk} = y$, the

user can generate valid credentials on arbitrary attributes. The user chooses a random $j' \leftarrow\!\!\$\ [\ell]$ and computes:

$$\tilde{T}_{j'} = (\tilde{g}^b \tilde{\mathrm{ipk}}_{j'})^a \bar{\mathrm{ipk}}_{j_0} \bar{\mathrm{pk}} = (\tilde{g}^b \tilde{\mathrm{ipk}}_{j'})^a \tilde{g}^{a(y_{j_0} - y)}$$

which implies:

$$\tilde{T}_{j'} = (\tilde{g}^b \tilde{\mathrm{ipk}}_{j'} \tilde{\mathrm{ipk}}_{j_0} \tilde{\mathrm{pk}}^{-1})^a$$

Since the presentation policy definition is the same in both Sanders and Traoré [46] and Katz and Sefranek [36], this attack applies to both schemes.

For example, in the case of Katz and Sefranek, the user can compute $\tilde{\sigma} = \tilde{S}^t \prod_{j \neq j_0} \tilde{T}_j = \tilde{g}^{ta} \tilde{g}^{(\ell-1)ba} \tilde{\mathrm{pk}}^{-a} \prod_{j \in [\ell]} \tilde{\mathrm{ipk}}_j^a$ and send his presentation vp with $\tilde{\sigma}$ to the RP. The RP checks the validity by testing whether:

$$\mathbf{e}(\bar{A}, \tilde{\sigma}^{\frac{-1}{a}} \tilde{g}^{(\ell-1)b} \prod_{j \in [\ell]} \tilde{\mathrm{ipk}}_j) = \mathbf{e}(\bar{B}, \tilde{g})$$

Since we have $\tilde{\sigma}^{\frac{-1}{a}} \tilde{g}^{(\ell-1)b} \prod_{j \in [\ell]} \tilde{\mathrm{ipk}}_j = \tilde{\mathrm{pk}} \tilde{g}^{-t}$, the presentation is accepted as valid by the RP, even though $(\tilde{\mathrm{pk}})$ is not part of the presentation policy. We provide a more sophisticated attack against Katz and Sefranek's scheme [36] in the sequel.
The attacks against the issuer-hiding constructions proposed by Flamini et al. [31] and Bosk et al. [11] are described in more detail later in this document (see Section B).

Our Improved BBS-based Issuer-Hiding Credentials. We adopt the signed policy approach first introduced by Bobolz et al. [6]. In this approach, the RP selects a set of authorized public keys $\mathcal{P}$ and signs each of them using a UF-CMA secure signature scheme. During the presentation, the user must prove the possession of a valid credential and demonstrate that the credential was issued by one of these authorized issuers. To achieve this, the user randomizes their credential, the issuer's public key, and the signature on that public key. Verification then involves checking that the credential is valid, and that the randomized public key and signature are verified correctly using the RP's public key.
Our construction supports two types of scenarios: either each RP generates its own policy by certifying a set of authorized issuer public keys $\mathcal{P}$, or a trusted certification authority generates the policy by certifying these keys. In both cases, we denote the certifying secret key as $\mathrm{vsk} = y$, with the corresponding verification keys $\mathrm{vpk} = g^y \in \mathbb{G}_1$ and $\tilde{\mathrm{vpk}} = \tilde{g}^y \in \mathbb{G}_2$. The issuer's public key is also represented in both groups as $\mathrm{ipk} = g^x \in \mathbb{G}_1$, and $\tilde{\mathrm{ipk}} = \tilde{g}^x \in \mathbb{G}_2$.
Our construction is based entirely on BBS signature: we rely on the BBS scheme both for credential issuance and for signing the set of authorized public keys ($\mathcal{P}$). For simplicity, we describe only the policy generation process, as the credential generation follows the standard BBS procedure. For each public key $(\mathrm{ipk}_j, \tilde{\mathrm{ipk}}_j) \in \mathcal{P}$, the issuer provides a proof of knowledge of the corresponding secret key isk_j:

$$\pi_{\text{key}} = \text{PoK}\{\text{isk}_j : \text{ipk}_j = g^{\text{isk}_j} \wedge \tilde{\text{ipk}}_j = \tilde{g}^{\text{isk}_j}\}$$

The RP checks this proof and, if valid, issues a certificate $\sigma_j = (A_j, e_j)$, where $A_j = (g_0 g^{\text{isk}_j})^{\frac{1}{x+e_j}}$ and $e_j \leftarrow\!\!\$\ \mathbb{Z}_p^*$.

For the presentation phase, let us consider a user who wants to prove the possession of a credential (A, e) on attributes $\{m_i\}$ with respect to some key $\tilde{\text{ipk}}_{j^*}$ in a set $\mathcal{P} = \{\tilde{\text{ipk}}_j\}_{j\in[\ell]}$, without revealing which issuer issued the credential. The user selects random scalars $t, r, r_1, r_2 \leftarrow\!\!\$\ \mathbb{Z}_p^*$, randomizes the issuer's public key as $(\text{ipk}'_{j^*} = g^{ty_{j^*}}, \tilde{\text{ipk}}'_{j^*} = \tilde{g}^{ty_{j^*}})$, where $\text{isk}_j = y_{j^*}$, and randomizes the corresponding certificate $\sigma_{j^*} = (A_{j^*}, e_{j^*})$ by computing $\bar{A}_{j^*} = A_{j^*}^{rt}$ and $\bar{B}_{j^*} = g_0^{t'}\,\text{ipk}'^{\,r}_{j^*}\,\bar{A}_{j^*}^{-e_{j^*}} = \bar{A}_{j^*}^{x}$, where $t' = tr$. Let $l = r_1 r_2$, $\hat{t} = tl$, and $e' = te$. The user then randomizes their credential (A, e) as $\bar{A} = A^l$ and sets $\bar{B} = C_m^{\hat{t}}\bar{A}^{-e'} = \bar{A}^{ty_{j^*}}$, where $C_m = h_0 \prod_i h_i^{m_i}$. Next, the user computes two zero-knowledge proofs, π_{cert} and π_{validity}, to prove the possession of a valid credential issued by one of the authorized issuers. Specifically:

$$\pi_{\text{cert}} = \text{PoK}\{\alpha, \beta, \delta : \bar{B}_{j^*} = g_0^{\alpha}\text{ipk}'^{\,\beta}_{j^*}\bar{A}_{j^*}^{-\delta} \wedge \alpha \neq 0\}, \text{ and}$$
$$\pi_{\text{validity}} = \text{PoK}\{\alpha, \beta, \gamma, \{\tau_i\}_{i\notin\mathcal{D}} : \bar{B} = \bar{A}^{-\alpha}\bar{D}^{\beta} \wedge h_0 \textstyle\prod_{i\in\mathcal{D}} h_i^{\tau_i} = \bar{D}^{\gamma} \prod_{i\notin\mathcal{D}} h_i^{-\tau_i}\},$$

where $\bar{D} = C_m^{r_2}$. The user sends the verifiable presentation:

$$vp = (\{m\}_{i\in\mathcal{D}}, \text{ipk}'_{j^*}, \tilde{\text{ipk}}'_{j^*}, \bar{A}_{j^*},\ \bar{B}_{j^*}, \bar{A}, \bar{B}, \bar{D}, \pi_{\text{cert}}, \pi_{\text{validity}}).$$

The RP checks the provided zero-knowledge proofs and the randomized keys and signatures:

$$\mathbf{e}(\text{ipk}'_{j^*}, \tilde{g}) = \mathbf{e}(g, \tilde{\text{ipk}}'_{j^*}) \wedge \mathbf{e}(\bar{B}_{j^*}, \tilde{h}) = \mathbf{e}(\bar{A}_{j^*}, \tilde{\text{vpk}}) \wedge \mathbf{e}(\bar{B}, \tilde{g}) = \mathbf{e}(\bar{A}, \tilde{\text{ipk}}'_{j^*}).$$

We also propose a pairing-free variant of this construction, adopting the concept of server-aided anonymous credentials introduced by Desmoulins et al. [25] and formalized by Chairattana-Apirom et al. [18] (see Section 6 for more detail).

1.4 Organization

This paper is structured as follows. Section 2 introduces our notation and the necessary cryptographic building blocks, including a recall of anonymous credential systems and the BBS signature scheme. Section 3 provides a formal definition of issuer-hiding ACs and presents our new security model. Section 4 recalls the scheme of Katz and Sefranek [36] and introduces concrete attack scenarios demonstrating its vulnerabilities. Section 5 presents our new issuer-hiding anonymous credential construction, along with its security analysis. Section 8 provides a performance analysis, including a comparison with existing schemes and concrete benchmark results illustrating the practical efficiency of our construction. Finally, Section 9 concludes the paper.

2 Preliminaries

2.1 Notation

We introduce some notations used throughout this document. To state that x is chosen uniformly at random from the set S, we use one of the two following notations $x \leftarrow\!\!\$\ S$ or $x \in_\$ S$. In addition, $\overrightarrow{m}$ will denote the vector or list $(m_1, m_2, \ldots, m_n)$, also written as $\{m_i\}_{i=1}^{n}$. λ will denote the security parameter and 1^λ will represent the security parameter in unary form. We use $(p, \mathbb{G}_1, \mathbb{G}_2, g, \tilde{g}, \mathbf{e}) \leftarrow\!\!\$\ \mathsf{BGen}(1^\lambda)$ to denote the output of a bilinear group generation algorithm, BGen is a bilinear group generator that takes as input a security parameter 1^λ, and outputs a description of a bilinear group $(\mathbb{G}_1, \mathbb{G}_2, g, \tilde{g}, p)$. Here, $\mathbb{G}_1$ and $\mathbb{G}_2$ are cyclic groups of prime order p, with respective generators g and $\tilde{g}$. All algorithms are probabilistic, unless otherwise indicated. By $y \leftarrow \mathcal{A}(x_1, x_2, \ldots, x_n)$, we denote the action of running $\mathcal{A}$ on inputs $(x_1, x_2, \ldots, x_n)$ and assigning the output to y. We write $y \leftarrow \mathcal{A}^{\mathcal{O}}(x_1, x_2, \ldots, x_n)$ to indicate that $\mathcal{A}$ is an algorithm, with oracle access to some algorithm or set of algorithms $\mathcal{O}$, that takes as inputs $(x_1, x_2, \ldots, x_n)$, and assigns the output to y.
We use the term "Experiment" in the context of security definitions and proofs. An experiment will be denoted **Exp**. An experiment **Exp** in which an adversary $\mathcal{A}$ interacts with a challenger $\mathcal{C}$ to break the security property prop of a scheme Schem is denoted by $\mathbf{Exp}^{\text{prop}}_{\mathcal{A}}(1^\lambda)$.
We analyze the security of our constructions in the Algebraic Group Model (AGM) as defined by Fuchsbauer et al. [33]. In this model, adversaries are assumed to know the representation of any group element they output. Specifically, after receiving a list of group elements $X_1, X_2, \ldots, X_n$, whenever the adversary returns a group element X, it must also provide a list of coefficients $\alpha_1, \alpha_2, \ldots, \alpha_n$ such that $X = \prod_i X_i^{\alpha_i}$. Adversaries satisfying this property are referred to as algebraic.

2.2 Zero-Knowledge Proofs of Knowledge

A Zero-Knowledge Proof of Knowledge (ZKPK) [34] is an interactive protocol between a prover $\mathcal{P}$ and a verifier $\mathcal{V}$, where the prover attempts to convince the verifier of the knowledge of some secrets verifying a given statement, without revealing any information about the said secrets. A ZKPK should satisfy three properties, namely (i) *completeness* (*i.e.* a valid prover should be able to convince an honest verifier with overwhelming probability), (ii) *soundness* (*i.e.* a malicious prover should be rejected with overwhelming probability), (ii) *zero-knowledge* (*i.e.* the proof reveals no information about the secret(s)). In our constructions, we use non-interactive zero-knowledge proofs of knowledge (also referred to as signatures of knowledge, or SoK), which can be obtained through heuristic transformations such as the Fiat–Shamir method [30]. For simplicity, we abstract away the details of these ZK proofs, as they would distract from the main ideas presented in this work.
We use the Camenisch-Stadler notation [17], where, for example, $\pi := \text{SoK}\{\alpha, \beta : y = g^\alpha \wedge z = g^\beta\}[m]$ denotes a signature of knowledge of secrets α, β, verifying

the statement on the right side of the colon. The signature of knowledge itself is generated on the message m. If the message is empty, we use the following notation to denote this signature of knowledge:

$$\pi := \text{PoK}\{\alpha, \beta : y = g^{\alpha} \wedge z = g^{\beta}\}.$$

In [43], Orrù showed that In the Algebraic Group Model (AGM), signatures of knowledge of a witness are straight-line extractable for any linear discrete-logarithm relationship, which means that one can extract the witness from a valid signature of knowledge produced by an adversary. We will need this result in our proof of unforgeability.

2.3 Oblivious Issuance of Proofs (OIP)

For our (pairing-free) BBS-based issuer-hiding anonymous credentials scheme, we will use a specific ZKPK, namely a *proof of equality of discrete logarithms* [22], denoted π_{DLEQ}, that can be requested *anonymously* and issued *obliviously* [42], i.e., in such a way that it cannot be linked back to the interaction that produced it: $\pi_{\text{DLEQ}} := \text{PoK}\{\alpha : B = A^{\alpha} \wedge h = g^{\alpha}\}$, where α is the prover's secret, g and h are two public generators of a cyclic group $\mathbb{G}$ and A and B are two generators of $\mathbb{G}$ satisfying $B = A^{\alpha}$ but which are unknown (blinded) to the prover.

More precisely, the proof will be issued in such a way that the prover will not be able to link $(A, B, \pi_{\text{DLEQ}})$ to its respective issuance (*obliviousness*) and the verifier will not be able after the issuance of l such proofs, even in a *concurrent manner*, to forge, on its own, a new valid proof (*one-more unforgeability*). The resulting proof is transferable and can be verified non-interactively by anyone.

2.4 Anonymous Credentials

In this section, we adopt the definitions presented in [36]. We begin by reviewing the formal definition of anonymous credentials before extending it to incorporate issuer-hiding in the following section. An anonymous credential scheme typically consists of four main algorithms. The key generation algorithm, IKeyGen, is executed by an issuer to generate a public/private key pair (ipk, isk). The issuance algorithm, Issue, is executed by the issuer to create a credential σ for a user associated with an attribute vector $(m_1, \ldots, m_n)$. We assume that the number of attributes n is fixed in advance. A user holding a credential σ for an attribute vector $(m_1, \ldots, m_n)$ can execute the Show algorithm to selectively disclose a subset $\mathcal{D} \subseteq [n]$ of attributes to a verifier while keeping the remaining attributes hidden. More formally, the user generates a proof that they possess a valid credential on an attribute vector consistent with the disclosed attributes. A verifier who knows the issuer's public key can then verify this proof.

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

- $\mathsf{Setup}(1^\lambda, n)$: On input a security parameter λ and an integer n, this algorithm outputs public parameters pp. The public parameters are considered implicit inputs for all the algorithms described below.

- $\mathsf{IKeyGen}(pp)$: On input the public parameters pp, this algorithm outputs a the issuer's public-private key pair (ipk, isk) .

- $(\mathsf{Obtain}(\text{usk}, \text{ipk}, \{m_i\}_{i=1}^n), \mathsf{Issue}(\text{upk}, \text{isk}, \{m_i\}_{i=1}^n)$: An interactive protocol between a user and an issuer. The user executes Obtain with their secret key usk, issuer's public key ipk, and attributes $\{m_i\}_{i=1}^n$. The issuer executes Issue with their secret key isk, user's public key upk, and the same attributes. If successful, Obtain returns a credential σ; otherwise, it returns $\perp$.

- $\mathsf{VerifyCred}(\text{ipk}, \{m_i\}_{i=1}^n, \sigma)$: On input an issuer public key ipk, a user attributes $\{m_i\}_{i=1}^n$, and a credential σ, this algorithm outputs a bit b.

- $\mathsf{Show}(\text{ipk}, \sigma, \mathcal{D}, \{m_i\}_{i=1}^n, \text{ctx})$: This algorithm is run by a user, on input a public key ipk, a credential σ, a disclosure set $\mathcal{D} \subseteq [n]$, attributes $\{m_i\}_{i=1}^n$, and a context ctx, it outputs a verifiable presentation vp.

- $\mathsf{Verify}(\text{ipk}, vp, \mathcal{D}, \{m_i\}_{i\in\mathcal{D}}, \text{ctx})$: This algorithm is run by a verifier, on input a public key ipk, a verifiable presentation vp, a disclosure set $\mathcal{D} \subseteq [n]$, attributes $\{m_i\}_{i\in\mathcal{D}}$, and a context ctx, it outputs a bit b.

Correctness requires the following to hold:

1. For all $\{m_i\}_{i=1}^n \in \mathbb{Z}_p^n$, we have
$$\Pr[\text{pp} \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda); \sigma \leftarrow\!\!\$\ \mathsf{Issue}(\text{isk}, \{m_i\}_{i=1}^n) : \mathsf{VerifyCred}(\text{ipk}, \{m_i\}_{i=1}^n, \sigma) = 1] = 1.$$
2. For all $\{m_i\}_{i=1}^n \in \mathbb{Z}_p^n$, all $\text{pp} \in\!\!\$ \{\mathsf{Setup}(1^\lambda)\}$, any $(\text{ipk}, \text{isk}) \in\!\!\$ \{\mathsf{IKeyGen}(\text{pp})$, any $\mathcal{D} \subseteq [n]$, any σ with $\mathsf{VerifyCred}(\text{ipk}, \{m_i\}_{i=1}^n, \sigma) = 1$, any ctx, and all $vp \in\!\!\$ \{\mathsf{Show}(\text{ipk}, \sigma, \mathcal{D}, \{m_i\}_{i=1}^n), \text{ctx}\}$, we have $\mathsf{Verify}(\text{ipk}, vp, \mathcal{D}, \{m_i\}_{i\in\mathcal{D}}, \text{ctx}) = 1$.

In addition to correctness, an anonymous credential scheme must satisfy two additional security properties: unforgeability (ensuring that no credential can be successfully shown unless it was legitimately issued by an authorized issuer) and anonymity (guaranteeing that no information about the user is revealed beyond the attributes they explicitly consent to disclose during the execution of the Show algorithm). We give define the formal definition of these two security requirements in Section A.1.

2.5 Keyed-Verification Anonymous Credentials (KVACs).

Traditional anonymous credential schemes rely on public key primitives, namely digital signatures with the issuer and verifier being distinct entities. In contrast,

KVACs, introduced by Chase et al. [19], represent the symmetric-key counterpart of anonymous credential schemes. They use symmetric primitives (namely algebraic MACs) and are designed for settings where the issuer also acts as the verifier, or more generally, where both share the same secret issuance key. Similar to anonymous credentials, KVACs must satisfy not only correctness but also unforgeability and anonymity, as defined earlier in the context of anonymous credentials (for formal definitions, see [19]). KVACs are based on algebraic MACs [19, 3], the symmetric-key counterparts of digital signatures. An algebraic MAC employs cyclic groups instead of hash functions or block ciphers, enabling efficient zero-knowledge proofs. This property makes algebraic MACs particularly suitable for privacy-preserving systems like anonymous credentials. Informally, a keyed-verification anonymous credential can be seen as the symmetric-key equivalent of an anonymous credential: the issuer and verifier share the same secret key, and verification is performed using this secret key. Consequently, KVACs do not provide public verifiability.
The use of symmetric key primitives like MACs makes KVACs more efficient and allows them to avoid reliance on pairings. However, the lack of public verifiability limits their applicability in scenarios where public verification is essential, such as digital identity systems, especially in the context of the European digital identity wallet (EUDIW).

Turning KVACs into Pairing-Free Anonymous Credentials Recently, Desmoulins et al. [25] showed how to transform a pairing-free KVAC system into a pairing-free public-key anonymous credential system by leveraging a recent result on oblivious issuance of proofs by Orrù et al. [42], while relying on assistance from the issuer during verification. This approach enables a user to prove possession of a credential to any verifier without requiring the verifier to know the issuer's private key. The core idea can be illustrated in the context of BBS. In BBS/BBS+ schemes, pairings are traditionally used by the verifier to check whether the equality $\bar{B} \stackrel{?}{=} \bar{A}^{\text{isk}}$ holds, where isk is the issuer's secret key (see $\mathsf{VerifyCred}(\text{isk}, \{m_i\}_{i\in\mathcal{D}})$, in Section 2.6). Desmoulins et al. proposed three alternatives to allow any verifier to perform this check without pairings:

1. The verifier asks the issuer to confirm whether $\bar{B} \stackrel{?}{=} \bar{A}^{\text{isk}}$ holds.
2. The user anonymously requests, during the Show protocol, a blind proof (also called an Oblivious Proof [42]), π_{dleq}, demonstrating that $\bar{B} \stackrel{?}{=} \bar{A}^{\text{isk}}$, which is then sent along with the VP to the verifier.
3. The user precomputes several pairs $(A_i = A^{l_i}, B_i = B^{l_i})$ for random values $l_i \in \mathbb{Z}_p^*$ and anonymously requests the issuer to produce blind proofs π_{dleq} for each, showing that $B_i = A_i^{\text{sk}_I}$. These proofs are stored for future use, particularly in offline scenarios where both user and verifier lack connectivity.

The blind proof π_{DLEQ} can be generated using the seminal Chaum-Pedersen blind signature protocol [22]. For more details on these options, we refer the reader to [25]. This concept was later formalized by Chairattana-Apirom et al.

[18] as server-aided anonymous credentials (SAACs). For simplicity, in this work we focus exclusively on the second option, which uses oblivious issuance of proofs [42] (for a formal definition of SAACs, see [18]).

2.6 BBS-Based Anonymous Credentials

In this section, we describe the BBS-based anonymous credential system. BBS exists in two variants: the original BBS scheme [7, 13, 16], which is more efficient and produces shorter signatures, and a less efficient variant known as BBS+ [1]. Tessaro and Zhu [47] recently provided security proofs for the original BBS construction. Our description follows their formulation, which corresponds to the original BBS scheme. The scheme is composed of the following algorithms:

- $\mathsf{Setup}(1^\lambda, n)$: On input a security parameter 1^λ and a bound n on the number of attributes to certify, this algorithm first runs $(p, \mathbb{G}_1, \mathbb{G}_2, g, \tilde{g}, \mathbf{e}) \leftarrow\!\!\$\ \mathsf{BGen}(1^\lambda)$, select s $g_0, h, h_0, h_1, \ldots, h_n \leftarrow\!\!\$\ \mathbb{G}_1$, and $\tilde{h} \leftarrow\!\!\$\ \mathbb{G}_2$, then returns the public parameters $\mathrm{pp} = (p, \mathbb{G}_1, \mathbb{G}_2, g, \tilde{g}, \mathbf{e}, g_0, h, h_0, h_1, \ldots h_n, \tilde{h})$.

- $\mathsf{IKeyGen}(\mathrm{pp})$: select $y \in_\$ \mathbb{Z}_p^*$, and returns $(\tilde{\mathrm{ipk}}, \mathrm{isk}) = (\tilde{g}^y, y)$.

- $\mathsf{Issue}(\mathrm{isk}, \{m_i\}_{i=1}^n)$: Let $\mathrm{isk} = y$, compute $C_m = h_0 \prod_i h_i^{m_i}$. Select $e \leftarrow\!\!\$\ \mathbb{Z}_p \backslash \{y\}$, and compute $A = C_m^{\frac{1}{y+e}}$. Return $\sigma = (A, e)$.

- $\mathsf{VerifyCred}(\tilde{\mathrm{ipk}}, \{m_i\}_{i=1}^n, \sigma = (A, e))$: Compute $C_m = h_0 \prod_i h_i^{m_i}$. If $C_m = 1$ return 0. Otherwise, return $\mathbf{e}(A, \tilde{\mathrm{ipk}}) \stackrel{?}{=} \mathbf{e}(A^{-e} C_m, \tilde{g})$.

- $\mathsf{Show}(\tilde{\mathrm{ipk}}, \sigma, \mathcal{D}, \{m_i\}_{i=1}^n)$: Parse σ as (A, e), and compute $C_m = h_0 \prod_i h_i^{m_i}$. Select $r_1, r_2 \leftarrow\!\!\$\ \mathbb{Z}_p^*$ and set $D = C_m^{r_1}, \bar{B} = \bar{A}^{-e} D^{r_2}$. Let $r_3 = r_1^{-1} \pmod p$ compute a ZKPK $\pi = \mathrm{PoK}\{(\{m_i\}_{i \notin \mathcal{D}}^n, e, r_2, r_3) : \bar{B} = \bar{A}^{-e} D^{r_2} \wedge h_0 \prod_{i \in \mathcal{D}} h_i^{m_i} = D^{r_3} \prod_{i \notin \mathcal{D}} h_i^{-m_i}\}$. Return $vp = (\{m_i\}_{i \in \mathcal{D}}^n, \bar{B}, \bar{A}, D, \pi)$.

- $\mathsf{Verify}(\tilde{\mathrm{ipk}}, vp, \mathcal{D}, \{m_i\}_{i \notin \mathcal{D}})$: vp as $\{m_i\}_{i \in \mathcal{D}}^n, \bar{B}, \bar{A}, D, \pi$. Accept if $\bar{A} \neq 1$, the verification of π succeed and if we have $\mathbf{e}(\bar{A}, \mathrm{ipk}) = (\bar{B}, \tilde{g})$.

2.7 BBS-Based KVACs

BBS-based keyed-verification anonymous credentials (KVACs) can be viewed as a variant of BBS-based anonymous credentials, but without public verifiability. In traditional BBS schemes, public verifiability relies on pairings and the issuer's public key, which enable checking whether the equality $\bar{B} \stackrel{?}{=} \bar{A}^{\mathrm{isk}}$ holds, where isk denotes the issuer's secret key. In contrast, KVACs rely on algebraic MACs and avoid pairings; verification is performed using the issuer' secret key rather than a public key. Informally, BBS-based KVACs share the same core algorithms as

BBS-based ACs: Setup, Issue, VerifyCred, Show, and Verify, but with a few key differences (for a formal definition of KVACs, see [19]). The Setup algorithm is similar, except that KVACs do not require a bilinear group; a single cyclic group $\mathbb{G}$ of prime order suffices. The IKeyGen algorithm is also similar, but the public key becomes optional and, when present, it lies in $\mathbb{G}$. During issuance, in addition to the standard BBS Issue procedure that outputs a credential $\sigma = (A, e)$, the issuer computes a zero-knowledge proof of knowledge (ZKPoK) $\pi_{\text{user}} = \text{PoK}\{\alpha : B = A^{\alpha}\}$, where $B = C_m A^{-e} = A^{\text{isk}}$. Along with the credential (A, e), the issuer returns π_{user}. Credential verification VerifyCred, consists of checking the ZKPoK π_{user}. The Show algorithm remains unchanged. The final verification Verify still checks π, but instead of using pairings to verify $\bar{B} \stackrel{?}{=} \bar{A}^{\text{isk}}$, it relies on the issuer's secret key isk to perform this check.

Remark 1. As discussed in Section 2.5, recent results [25, 18] show how to transform KVACs into pairing-free anonymous credentials, thereby eliminating the need for the verifier to know the issuer's secret key. These constructions have attracted significant interest within the anonymous credential community, particularly in the context of EUDIW, as they enable public verifiability without relying on pairings. However, a major remaining challenge is how to incorporate issuer-hiding properties into such protocols. This is a question we aim to address in this work.

3 Issuer-Hiding Anonymous Credentials

Earlier anonymous credentials [16, 44, 32] focused on concealing the user's identity while revealing the issuer's identity. However, in certain scenarios, disclosing the issuer's identity may compromise the user's anonymity, meaning that most existing schemes do not provide complete anonymity guarantees. Introducing issuer-hiding addresses this issue by preventing any inference about the user's identity based on the knowledge of the issuer's identity.

In this section, we recall the definition of issuer-hiding anonymous credentials (AC) [6, 45, 36] by extending the basic anonymous credential model presented in Section 2.4 to include additional algorithms required for issuer-hiding. While some algorithms from the basic model, such as Setup, may require modifications to support issuer-hiding, others namely IKeyGen, Issue, and VerifyCred (see Definition 1) remain unchanged. The main differences occur in the Show and Verify algorithms, which must be adapted to incorporate the issuer-hiding feature.

Two new algorithms are introduced: GenPolicy, and VerifyPolicy. The entity responsible for generating the presentation policy pol receives a set of authorized issuers $\mathcal{P}$ (identified by their public keys, with $|\mathcal{P}| > 1$), runs GenPolicy($\mathcal{P}$), and outputs pol. Our construction allows GenPolicy to be executed either by the RP or by a certification authority; w.l.o.g, we assume throughout this paper that the certification authority runs GenPolicy. The resulting presentation policy pol can be verified by anyone by running the VerifyPolicy($\mathcal{P}$, pol) algorithm.

Definition 2. *(Issuer-Hiding Anonymous Credentials).* An issuer-hiding anonymous credential scheme consists of the algorithms $\mathsf{Setup}, \mathsf{IKeyGen}, \mathsf{Obtain}, \mathsf{Issue}$, and $\mathsf{VerifyCred}$ as defined in Definition 1, along with the following additional or extended algorithms:

- $\mathsf{GenPolicy}(\mathcal{P})$: Given a set of public keys $\mathcal{P}$ as input, this algorithm outputs a presentation policy pol.
- $\mathsf{VerifyPolicy}(\mathcal{P}, \text{pol})$: Given a set of issuers public keys $\mathcal{P}$, and a presentation policy pol as input, this algorithm outputs a bit b.
- $\mathsf{Show}(\text{ipk}, \text{pol}, \sigma, \mathcal{D}, \{m_i\}_{i=1}^n, \text{ctx})$: Given a public key ipk, a credential σ, a disclosure set $\mathcal{D} \subseteq [n]$, attributes $\{m_i\}_{i=1}^n$, a presentation policy pol, and a context ctx as input, it outputs a verifiable presentation vp.
- $\mathsf{Verify}(\text{pol}, vp, \mathcal{D}, \{m_i\}_{i\in\mathcal{D}}, \mathcal{D}, \text{ctx})$: This algorithm is run by a RP. Given a verifiable presentation vp, a disclosure set $\mathcal{D} \subseteq [n]$, attributes $\{m_i\}_{i\in\mathcal{D}}$, a presentation policy pol and ctx a as input, it outputs a bit b.

The correctness requires the following to hold:

1. For all $\text{pp} \in_\$ \{\mathsf{Setup}(1^\lambda)\}$, any set of keys $\{(\text{isk}_1, \text{ipk}_1, \tilde{\text{ipk}}_j), \ldots, (\text{isk}_\ell, \text{ipk}_\ell, \tilde{\text{ipk}}_\ell)\} \in_\$ \{\mathsf{IKeyGen}(pp)\}^\ell$, such that $\mathcal{P} = \{(\text{ipk}_1, \tilde{\text{ipk}}_j), \ldots, (\text{ipk}_\ell, \tilde{\text{ipk}}_\ell)\}$, any $\text{pol} \in_\$ \{\mathsf{GenPolicy}(\mathcal{P})\}$, we have:
$$\mathsf{VerifyPolicy}(\mathcal{P}, \text{pol}) = 1$$
2. For all $\{m_i\}_{i=1}^n \in \mathbb{Z}_p^n$, all $\text{pp} \in_\$ \{\mathsf{Setup}(1^\lambda)\}$, any set of keys $\mathcal{P}$, any $\text{pol} \in_\$ \{\mathsf{GenPolicy}(\mathcal{P})\}$, any $\mathcal{D} \subseteq [n]$, any σ with $\mathsf{VerifyCred}(\text{ipk}_j, \{m_i\}_{i=1}^n, \sigma) = 1$, any ctx and all $vp \in_\$ \{\mathsf{Show}(\text{ipk}_j, \text{pol}, \sigma, \mathcal{D}, \{m_i\}_{i=1}^n, \text{ctx})\}$, we have:
$$\mathsf{Verify}(vp, \text{pol}, \mathcal{D}, \{m_i\}_{i\in\mathcal{D}}, \text{ctx}) = 1.$$

In addition to the correctness, issuer-hiding anonymous credentials must satisfy two security properties which are formalized below: unforgeability and anonymity. However, unlike the definition in Section A.1 for basic anonymous credentials, unforgeability here encompasses not only credential unforgeability but also the policy unforgeability. This means that a user should not be able to produce a verifiable presentation that passes the verification without possessing a credential issued by an authorized issuer within $\mathcal{P}$.
Similarly, anonymity extends beyond user anonymity, as defined in Definition 5, to include issuer anonymity, namely issuer-hiding. In practice, this is achieved by allowing the adversary to specify two issuer public keys in addition to two sets of attributes and two credentials, as in Section A.1.

We define these two security requirements through the experiments described in Figure 1 which use the following oracles along with some global lists that are shared the among oracles. We defined some new global list specific to the issuer-hiding AC model, such as HI, the list containing the identities of *honest*

issuers, and CI, the list containing the identities of *corrupt* issuers, and IPK ISK respectively the list of issuers public and secret keys. Following [32, 45], we additionally define the list ATT, that stores $(u, \{m_i\}_{i=1}^n, \sigma)$, each time a credential is generated for a user u on $\{m_i\}_{i=1}^n$ by the oracle $\mathcal{O}\mathsf{Issue}$ defined below. We say that $(u, \{m_i\}_{i\in\mathcal{D}}, \sigma) \in \text{ATT}$ if $\exists (u, \{m'_i\}_{i=1}^n, \sigma) \in \text{ATT}$ with $m'_i = m_i$ for all $i \in \mathcal{D}$.

- $\mathcal{O}\text{HI}(j)$: on input an issuer identity j, this oracle returns $\perp$ if $j \in \text{HI} \cup \text{CI}$. Else it generates a key pair $(\text{ISK}[j], \text{IPK}[j]) \leftarrow\!\$\ \mathsf{IKeyGen}(\text{pp})$ and returns $\text{IPK}[j]$. The identity j is then added to HI.
- $\mathcal{O}\text{CI}(j, \text{ipk})$: on input a issuer identity j , and optionally a public key ipk [5], this oracle registers a new corrupted issuer with public key ipk if j does not already exist (meaning $j \notin \text{HI} \cup \text{CI}$). Otherwise if $j \notin \text{HI}$ it removes j from HI. In all cases, j is added to CI.
- $\mathcal{O}\mathsf{Issue}(u, j, \text{ipk}, \{m_i\}_{i=1}^n)$: on input a user identity u, an issuer identities j, and optionally his public key ipk, and a set of attributes $\{m_i\}_{i=1}^n$, this oracle runs $\sigma \leftarrow\!\$\ (\mathcal{A}^{\mathcal{O}\mathsf{Obtain}(.)}, \mathsf{Issue}(\text{upk}, \text{isk}, \{m_i\}_{i=1}^n))$. If $j \notin \text{HI}$, the oracle returns $\perp$. Else it appends $(u, \{m_i\}_{i=1}^n, \sigma)$ to ATT. This oracle is used by an adversary playing the role of a malicious user to obtain a credential from an honest issuer.
- $\mathcal{O}\mathsf{Show}(k, \text{pol}, \mathcal{D}, \text{ctx})$: On input an issuance query index k, verification policy pol, and and attributes-subset $\mathcal{D}$, this oracle parse $\text{ATT}[k]$ as $(u, \{m_i^k\}_{i=1}^n, \sigma^k)$. This oracle runs $vp \leftarrow\!\$\ \mathsf{Show}(\text{ipk}, \text{usk}, \{m_i^k\}_{i=1}^n, \mathcal{D}, \sigma^k, \text{ctx})$.

Anonymity.

Definition 3. *(Anonymity).* We denote by AnoSucceeds the event that $\mathcal{A}$ succeeds in the $\mathbf{Exp}_{\mathcal{A}}^{\text{ih}-\text{ano}}(1^\lambda)$, which occurs when $\mathbf{Exp}_{\mathcal{A}}^{\text{ih}-\text{ano}}(1^\lambda)$ output 1. The advantage of $\mathcal{A}$ is then defined as:

$$\mathsf{Adv}_{\mathcal{A},\Pi}^{\text{ano}}(1^\lambda) = |\Pr[\text{AnoSucceeds}] - \tfrac{1}{2}|.$$

We say that the scheme Π satisfies the issuer-hiding anonymity if $\mathsf{Adv}_{\mathcal{A},\Pi}^{\text{ano}}(1^\lambda)$ is negligible.

Unforgeability. In their security models for the unforgeability requirement, Flamini et al. [31] and Katz and Sefranek [36] only consider the case of honest issuers, meaning the adversary does not have access to their private keys. In our model, however, we consider malicious issuers, meaning the adversary holds the relevant private keys. More precisely, we allow the adversary to generate their own issuance keys by granting them access to the oracle $\mathcal{O}\mathsf{IKeyGen}$.
Nevertheless, we exclude trivial forgeries consisting of the presentation of a valid credential issued with a corrupted private key.

[5] The issuer proves knowledge of the private key corresponding to ipk before it is certified (and therefore, in the security proof, the reduction will be able to extract the private key corresponding to ipk, when the adversary attempts to register a public key ipk via the oracle $\mathcal{O}\text{CI}(j, \text{ipk})$).

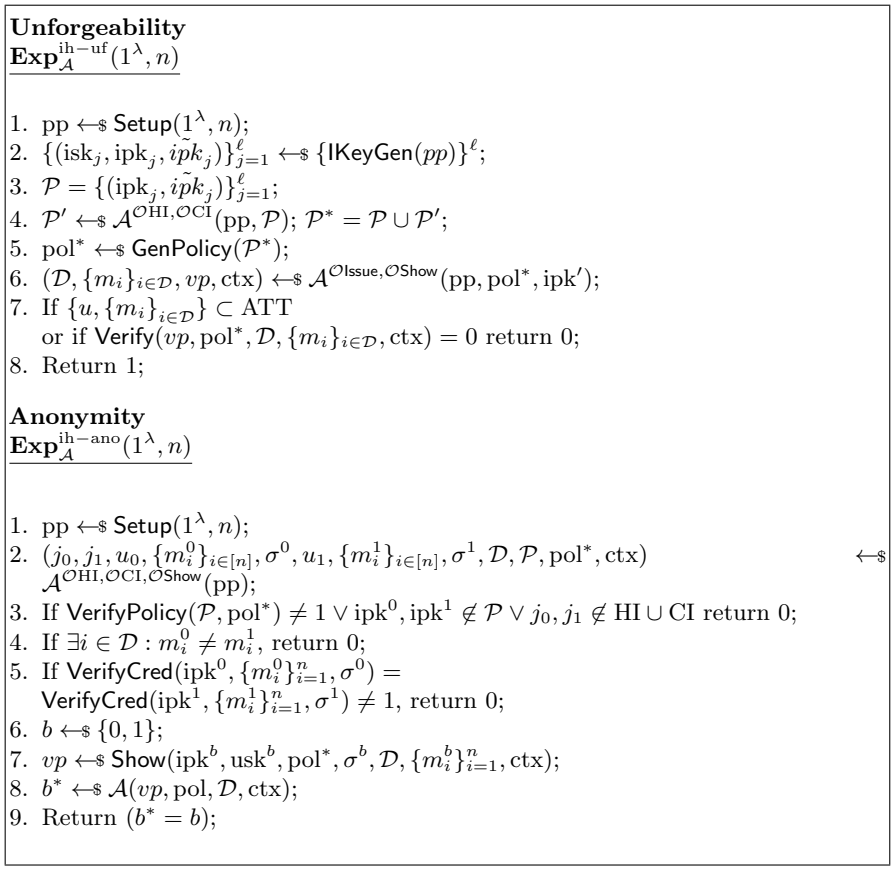


Fig. 1: Security requirements for issuer-hiding anonymous credentials

Definition 4. *(Unforgeability).* We denote by UfSucceeds the event that $\mathcal{A}$ succeeds in the $\mathbf{Exp}^{\text{ih}-\text{uf}}_{\mathcal{A}}(1^\lambda)$, which occurs when $\mathbf{Exp}^{\text{ih}-\text{uf}}_{\mathcal{A}}(1^\lambda)$ output 1. The advantage of $\mathcal{A}$ is then defined as:

$$\mathsf{Adv}^{\text{ih}-\text{uf}}_{\mathcal{A},\Pi}(1^\lambda) = \Pr[\text{UfSucceeds}].$$

We say that the scheme Π is unforgeable if $\mathsf{Adv}^{\text{ih}-\text{uf}}_{\mathcal{A},\Pi}(1^\lambda)$ is negligible.

4 Katz and Sefranek BBS-based issuer-hiding

The recent work introduced by Katz and Sefranek [36] presents the first issuer hiding AC scheme based on BBS signatures. A key achievement of their work is that issuer-hiding was integrated without modifying the underlying BBS scheme. Their approach is more efficient than the Sanders and Traoré construction [46], because it benefits from the inherent efficiency of BBS-based credentials compared to Pointcheval-Sanders (PS) based credentials, on which the Sanders and Traoré scheme relies on. Specifically, the Katz and Sefranek scheme uses constant size issuer public keys, whereas Sanders and Traoré's public keys grow linearly with the number of attributes n. Consequently, Katz and Sefranek achieve significantly shorter policy keys. In this section, we revisit the Katz and Sefranek construction, outline attacks applicable to both schemes [36] and [46].

4.1 The construction

We now recall the BBS-based issuer-hiding anonymous credential system introduced by Katz and Sefranek [36]. We omit the Setup, $\mathsf{IKeyGen}$, Issue and $\mathsf{VerifyCred}$ algorithms, as they are identical to those described in Section 2.6. The remaining algorithms are defined as follows:

- $\mathsf{GenPolicy}(\mathcal{P})$: Parse $\mathcal{P}$ as $\{\tilde{\text{ipk}}_j\}_{j\in[\ell]}$. Select $a \leftarrow\!\!\$\, \mathbb{Z}_p^*$ and $b \leftarrow\!\!\$\, \mathbb{Z}_p$, and then compute $\tilde{S} = \tilde{g}^a$. For $j \in [\ell]$, set $\tilde{T}_j = (\tilde{\text{ipk}}_j \tilde{g}^b)^a$. Set $\text{sk}_{\mathcal{P}} = (a, b)$. The policy key $\text{pk}_{\mathcal{P}}$ consists of the pair $(S, \{\tilde{T}_j\}_{j\in[\ell]})$ and a ZKPK proving the following statement:

$$\exists a, b \text{ s.t. } \tilde{S} = \tilde{g}^a \wedge \forall j : \tilde{T}_j = (\tilde{\text{ipk}}_j \tilde{g}^b)^a. \tag{1}$$

- $\mathsf{VerifyPolicy}(\mathcal{P}, \text{pk}_{\mathcal{P}})$: Parse $\text{pk}_{\mathcal{P}}$ as $(S, \{\tilde{T}_j\}_{j\in[\ell]})$ plus a ZPK of statement (eq. (1)). Accept if $\ell = |\mathcal{P}|, \tilde{S} \neq 1$, and the verification of the ZKPK succeeds.

- $\mathsf{Show}(\tilde{\text{ipk}}, \sigma, \mathcal{D}, \{m_i\}_{i=1}^n, \mathcal{P})$: If $|\mathcal{P}| = 1$, the original Show algorithm is used, so we assume $|\mathcal{P}| > 1$. Parse $\mathcal{P}$ as $\{\tilde{\text{ipk}}_j\}_{j\in[\ell]}$ and σ as (A, e), and let j^* be s.t. $\tilde{\text{ipk}} = \tilde{\text{ipk}}_{j^*}$. Select $r_1, r_2 \leftarrow\!\!\$\, \mathbb{Z}_p^*$ and $t \leftarrow\!\!\$\, \mathbb{Z}_p$. Compute $C_m = h_0 \prod_{i\in[n]} h_i^{m_i}, \bar{A} = A^{r_1 r_2}, D = C_m^{r_1}, \bar{B} = \bar{A}^{-e-t} D^{r_2}$, and $\tilde{\sigma} = \tilde{S}^t \prod_{j\neq j^*} \tilde{T}_j$. The verifiable presentation vp consists of $\bar{A}, \bar{B}, D$, and $\tilde{\sigma}$, along with a ZKPK such as:

$$\exists \{m_i\}_{i\in\mathcal{D}}, e', r_1, r_2 s.t.$$
$$\bar{B} = \bar{A}^{-e'} D^{r_2} \wedge h_0 \prod_{i\in\mathcal{D}} h_i^{m_i} = D^{\frac{1}{r_1}} \prod_{i\notin\mathcal{D}} h_i^{-m_i} \tag{2}$$

– $\mathsf{Verify}(vp, \mathcal{D}, \{m_i\}_{i\in\mathcal{D}}, \mathcal{P})$: If $|\mathcal{P}| = 1$, the original Verify algorithm is used, so we assume $|\mathcal{P}| > 1$. Parse vp as $\bar{A}, \bar{B}, D$, and $\tilde{\sigma}$ and a ZPK of statement (eq. (2)), and parse $\mathrm{sk}_{\mathcal{P}}$ as (a, b). Accept iff (1) $\bar{A} \neq 1$, (2) the verification of the ZKPK succeeds, and (3) eq. (3) holds:

$$\mathbf{e}(\bar{A}, \tilde{\sigma}^{\frac{-1}{a}} \tilde{g}^{(\ell-1)b} \prod_{j\in[\ell]} i\tilde{pk}_j) \stackrel{?}{=} \mathbf{e}(\bar{B}, \tilde{g}). \tag{3}$$

4.2 Attacks

Our attack targets the presentation policy and can be viewed as an attack on the policy unforgeability of the Katz and Sefranek scheme. It enables a user to authenticate using a verifiable presentation for a credential that was not issued by any of the authorized issuers in $\mathcal{P}$. The two following realistic scenarios are defined below, where the user must collaborate with one or more authorized issuers.

Scenario 1. In this scenario, an authorized issuer assists a malicious user. The issuer helps the user to issue new credentials using their own secret keys, even though the corresponding public key is not included in the authorized issuer's public key set, $\mathcal{P}$. Consequently, the user can obtain credentials for arbitrary attributes using the unauthorized issuance keys that they will be able to successfully present to any relying party. We describe the attack as follows:

– **Setup:** an authorized issuer with secret key $\mathrm{isk}_{j_0} = y_{j_0}$ and public key $i\tilde{pk}_{j_0} = \tilde{g}^{y_{j_0}}$ sends to the malicious user the commitment $i\bar{pk}_{j_0} = \tilde{S}^{y_{j_0}} = \tilde{g}^{ay_{j_0}}$ on his secret key. The malicious user then generates a secret key $\mathrm{sk} = y \leftarrow\!\$\ \mathbb{Z}_p^*$, and computes a public key $\tilde{\mathrm{pk}} = \tilde{g}^y$, and $\bar{\mathrm{pk}} = \tilde{S}^{-y} = \tilde{g}^{-ay}$.
– **Credential generation:** To issue a credential on attribute $\{m_i\}_{i\in[n]}$, the user runs a typical BBS-credential generation on $\{m_i\}_{i\in[n]}$ with $\mathrm{sk} = y$ being the issuance key here. He outputs a credential $\sigma^* = (A^*, e^*)$, which can be verified using the corresponding verification algorithm.
– **Showing:** The only difference with the Show algorithm presented in Section 4.1 is the computation of $\tilde{\sigma}$. To do so, the user chooses a random $j' \leftarrow\!\$\ [\ell]$ and computes:

$$\tilde{T}_{j'} = (\tilde{g}^b i\tilde{pk}_{j'})^a \mathrm{i}\bar{\mathrm{pk}}_{j_0} \bar{\mathrm{pk}} = (\tilde{g}^b \mathrm{i}\tilde{\mathrm{pk}}_{j'})^a \tilde{g}^{a(y_{j_0}-y)}$$

This implies:

$$\tilde{T}_{j'} = (\tilde{g}^b \mathrm{i}\tilde{\mathrm{pk}}_{j'} \mathrm{i}\tilde{\mathrm{pk}}_{j_0} \tilde{pk}^{-1})^a$$

He computes $\tilde{\sigma} = \tilde{S}^t \prod_{j \neq j_0} \tilde{T}_j = \tilde{g}^{ta} \tilde{g}^{(\ell-1)ba} \tilde{pk}^{-a} \prod_{j \in [\ell]} \tilde{\text{ipk}}_j^a$. He outputs a verifiable presentation vp^* while running the remaining steps of the Show algorithm as described in Section 4.1.

– **Verification:** The user verifiable presentation is verified using the Verify algorithm defined in Section 4.1. The verifier checks the presentation by testing whether:

$$\mathbf{e}(\bar{A}^*, \tilde{\sigma}^{\frac{-1}{a}} \tilde{g}^{(\ell-1)b} \prod_{j \in [\ell]} \tilde{\text{ipk}}_j) = \mathbf{e}(\bar{B}^*, \tilde{g})$$

Since we have $\tilde{\sigma}^{\frac{-1}{a}} \tilde{g}^{(\ell-1)b} \prod_{j \in [\ell]} \tilde{\text{ipk}}_j = \tilde{\text{pk}} \tilde{g}^{-t}$, the presentation is accepted by the verifier, even though $\tilde{\text{pk}}$ is not included in $\mathcal{P}$. This constitutes the attack in *scenario 1*.

Remark 2. The attack in *scenario 1* is very powerful, because it effectively grants the user the capabilities of an authorized issuer. However, its main drawback lies in accountability: if the user's fraudulent actions are detected, for example, a service provider discovers that a minor has presented a proof of majority to access age-restricted content, the verifier can initiate legal procedures against the user. Under pressure, the user may choose to cooperate and reveal the identity of the issuer who assisted them, providing $\bar{\text{ipk}}_{j_0}$ as evidence. From $\bar{\text{ipk}}_{j_0}$, the verifier can easily recover the issuer's identity using the policy secret key $\text{sk}_{\mathcal{P}} = (a, b)$ by computing $\tilde{ipk}_{j_0} = \bar{\text{ipk}}_{j_0}^{\frac{1}{a}}$.
Consequently, this scenario enables investigators to identify the malicious issuer and punish them, which should deter issuers from engaging in such activities.

Scenario 2. Unlike the previous scenario, the attack in this scenario involves two authorized issuers collaborating to generate a credential on arbitrary attributes for a given user using a new issuance key that differs from both of their original keys. The corresponding public key is not included in $\mathcal{P}$. During this process, the issuers randomize the commitments to their public keys so that their identities cannot be recovered. We describe the attack as follows:

– **Setup:** we consider two authorized issuers. For sake of simplicity, we assume that the index of their public keys in $\mathcal{P}$ are $j = 1$ for the first one and $j = 2$ for the second one. Therefore we have issuer-1 with secret key $\text{isk}_1 = y_1$ and public key $\text{ipk}_1 = \tilde{g}^{y_1}$, and issuer-2 with secret key $\text{isk}_2 = y_2$ and public key $\text{ipk}_2 = \tilde{g}^{y_2}$. Both issuers randomize their secret key $\bar{\text{isk}}_1 = \bar{y}_2 = \alpha y_1, \bar{\text{isk}}_2 = \bar{y}_2 = \beta y_2$, and commit to their respective new keys $\bar{\text{ipk}}_1 = \tilde{S}^{\bar{y}_1} = \tilde{g}^{a\alpha y_1}, \bar{\text{ipk}}_2 = \tilde{S}^{\bar{y}_2} = \tilde{g}^{a\beta y_2}$, with $\alpha, \beta \in_{\$} \mathbb{Z}_p^*$. Now they can generate credentials on arbitrary attributes with the issuance key $\text{sk} = y = \bar{y}_1 + \bar{y}_2 = \alpha y_1 + \beta y_2$, and the corresponding public key $\tilde{\text{pk}} = \tilde{g}^y$ is not included in $\mathcal{P}$.

– **Credential generation:** To issue a credential on arbitrary attributes $\{m_i\}_{i \in [n]}$, both issuers can use a threshold BBS scheme such as the one proposed by Doerner et al. [26].

Let us denote by $\sigma^* = (A^*, e^*)$ this credential, which is a valid credential on $\{m_i\}$, with respect to $\tilde{\text{pk}}$.
The issuers then jointly compute $\tilde{T} = \tilde{T}_1 \tilde{S}^{y_2} i\tilde{\text{pk}}_1^{-1} i\tilde{\text{pk}}_2^{-1} = (\tilde{g}^b i\tilde{\text{pk}}_1 i\tilde{\text{pk}}_2 \tilde{\text{pk}}^{-1})^a$ and then $\bar{T} = \tilde{T} \prod_{j\neq 1, j\neq 2} i\tilde{\text{pk}}_j = \tilde{g}^{(l-1)ba} \tilde{\text{pk}}^{-a} \prod_{j\in[l]} i\tilde{\text{pk}}_j^a$. Eventually, the issuers send the credential $\sigma^* = (A^*, e^*)$ along with $\bar{T}$ to the malicious user. They may also publish these values on the dark web.

– **Showing:** The user selects $t \leftarrow\$ \mathbb{Z}_p^*$ and computes:

$$\tilde{\sigma} = \tilde{S}^t \bar{T} = \tilde{g}^{ta} \tilde{g}^{(\ell-1)ba} \tilde{\text{pk}}^{-a} \prod_{j\in[\ell]} i\tilde{\text{pk}}_j^a$$

The rest is the same as with the Show algorithm in Section 4.1. He outputs a verifiable presentation vp^*.

– **Verification:** The user verifiable presentation is verified using the Verify algorithm defined in Section 4.1. The verification succeeds, as the verifier checks the presentation by testing whether:

$$\mathbf{e}(\bar{A}^*, \tilde{\sigma}^{\frac{-1}{a}} \tilde{g}^{(\ell-1)b} \textstyle\prod_{j\in[\ell]} \tilde{\text{ipk}}_j) = \mathbf{e}(\bar{B}^*, \tilde{g})$$

Since we have $\tilde{\sigma}^{\frac{-1}{a}} \tilde{g}^{(\ell-1)b} \prod_{j\in[\ell]} \tilde{\text{ipk}}_j = \tilde{\text{pk}} \tilde{g}^{-t}$, the presentation is accepted by the verifier, even though $(\tilde{\text{ipk}})$ is not included in $\mathcal{P}$. This constitutes the attack in *scenario 2*.

This scenario is the most practical among those presented in this paper and the most likely to occur. In this case, the issuers avoids any risk of having their identity disclosed or being blacklisted. We assume that if an issuer can misbehave without any risk of being exposed, they are highly likely to do so, especially when it serves their financial interests or involves other forms of bribery. While this scenario does not grant the user the authority of a new authorized issuer, as in the previous case (scenario 1), it remains the most realistic option we have identified so far.

Remark 3. In the above attack scenario, we have assumed that at least two issuers are malicious. At first glance, this assumption may seem too strong as one could argue that if an issuer is malicious, they could issue fake credentials to users — for instance, a proof of majority to a minor. However, by doing so, the malicious issuer would risk being exposed or blackmailed by a minor who could threaten to present the false proof of majority to a child protection agency. The issuer could not deny the misdeed since the proof of majority would bear their signature (a BBS signature, in this case). Therefore, an issuer would not take the risk of committing such fraud. For our attack to succeed, it is therefore necessary that the issuers behind such fraud cannot be identified.

Remark 4. We would like to emphasize that the above attacks do not show that Katz and Sefranek's scheme is insecure with respect to the trust and security

models they adopted. In these models, they tacitly consider "honest" issuers keys, which means that the adversary does not have access to the corresponding private keys. Consequently, our attacks would not work. However, our attacks do suggest that these models are not realistic for real-world applications, such as the European digital identity wallet.

Remark 5. Since the presentation policy definition is the same for both the Sanders and Traoré [46] and Katz and Sefranek [36] constructions, the attacks described in both scenarios also apply to the Sanders and Traoré scheme.

In addition to these attacks, we also present attacks against the issuer-hiding constructions proposed by Flamini et al. [31] and Bosk et al. [11], which are described in detail in Section B.

5 Improved Issuer-Hiding For BBS-Based ACs

We present our construction, which takes a different approach to that proposed by Katz and Sefranek [36] for achieving issuer-hiding in BBS-based anonymous credentials. Our design follows a more standard method, namely the verifier-signed policy approach [6] by relying on well-established primitives. Specifically, we use BBS for both credential generation and presentation policy generation, meaning our scheme is fully BBS-based. The decision to diverge from the Sanders and Traoré [46] approach is motivated by our goal of using a more standardized method to reduce exposure to attacks such as those discussed in Section 4.2. Unlike the constructions of Katz–Sefranek and Sanders–Traoré, it would be extremely difficult for a user to prove possession of a valid credential that was not issued by an authorized issuer in $\mathcal{P}$. Doing so would require breaking policy unforgeability, which in our case is equivalent to forging a valid BBS signature.

5.1 Construction

We now describe how to incorporate issuer-hiding into BBS-based anonymous credentials using a verifier-signed policy. We omit the Setup, Issue, and $\mathsf{VerifyCred}$ algorithms, as they are identical to those in the underlying scheme described in Section 2.6. The remaining algorithms are defined as follows:

- $\mathsf{IKeyGen}(\mathrm{pp})$: select $y \in_{\$} \mathbb{Z}_p^*$, and return $(\mathrm{ipk}, \tilde{\mathrm{ipk}}, \mathrm{isk}) = (g^y, \tilde{g}^y, y)$. The issuer also computes a ZKPK $\pi_{\mathrm{key}} = \mathrm{PoK}\{\alpha : \mathrm{ipk} = g^\alpha \wedge \tilde{\mathrm{ipk}} = \tilde{g}^\alpha\}$.
- $\mathsf{KeyGen}(\mathrm{pp})$: select $x \in_{\$} \mathbb{Z}_p^*$, and returns $(\tilde{\mathrm{vpk}}, \mathrm{vsk}) = (\tilde{h}^x, x)$.
- $\mathsf{GenPolicy}(\mathcal{P})$: Parse $\mathcal{P}$ as $\{(\mathrm{ipk}_j, \tilde{\mathrm{ipk}}_j)\}_{j \in [\ell]}$. Run $(\tilde{\mathrm{vpk}}, \mathrm{vsk}) \leftarrow_{\$} \mathsf{KeyGen}(\mathrm{pp})$. For $j \in [\ell]$ verify the provided π_{key} if the verification succeeds, certify the issuer j, by computing

$$\sigma_j = (A_j, e_j), \text{ where } A_j = (g_0 \mathrm{ipk}_j)^{\frac{1}{x+e_j}}, \text{ and } e_j \in \mathbb{Z}_p \backslash \{-x\}.$$

We assume that for $j \in [\ell]$ the verification of π_{key} succeeds. Output the presentation policy

$$\text{pol} = (\tilde{\text{vpk}}, \{(\text{ipk}_j, \tilde{\text{ipk}}_j, \sigma_j, \pi_{\text{key}})\}_{j\in[\ell]}).$$

– $\mathsf{VerifyPolicy}(\mathcal{P}, \text{pol})$: Parse pol as $\tilde{\text{vpk}}, \{(\text{ipk}_j, \tilde{\text{ipk}}_j, \sigma_j, \pi_{\text{key}})\}_{j\in[\ell]}$. For $j \in [\ell]$ set $B_j = g_0 \text{ipk}_j A_j^{-e_j} = A_j^x$. Accept if $\ell = |\mathcal{P}|$, and for all $j \in [\ell]$

$$\mathbf{e}(B_j, \tilde{h}) = \mathbf{e}(A_j, \tilde{\text{vpk}}), \text{ and } \pi_{\text{key}} \text{ is valid.}$$

– $\mathsf{Show}(\text{ipk}_{j^*}, \sigma, \mathcal{D}, \{m_i\}_{i=1}^n, \text{pol}, \mathcal{P})$: If $|\mathcal{P}| = 1$, the original Show algorithm is used, so we assume $|\mathcal{P}| > 1$. Parse $\mathcal{P}$ as $\{\tilde{\text{ipk}}_j\}_{j\in[\ell]}$, σ as (A, e), and σ_{j^*} as (A_{j^*}, e_{j^*}). Select $t, r, r_1, r_2 \leftarrow\!\!\$\ \mathbb{Z}_p^*$, compute $(\text{ipk}'_{j^*} = g^{ty_{j^*}}, \tilde{\text{ipk}}'_{j^*} = \tilde{g}^{ty_{j^*}})$, and $\bar{A}_{j^*} = A_{j^*}^{rt}$ and $\bar{B}_{j^*} = g_0^{t'} {\text{ipk}'_{j^*}}^r \bar{A}_{j^*}^{-e_{j^*}} = \bar{A}_{j^*}^x$, where $t' = tr$. Let $l = r_1 r_2$, $\hat{t} = tl$, and $e' = te$. Compute $\bar{A} = A^l$ and set $\bar{B} = C_m^{\hat{t}} \bar{A}^{-e'} = \bar{A}^{ty_{j^*}}$, $\bar{D} = C_m^{r_2}$, $\bar{B} = \bar{D}^{tr_1} \bar{A}^{-e'}$, where $C_m = h_0 \prod_i h_i^{m_i}$, and two zero-knowledge proofs, π_{cert} and π_{validity}:

$$\pi_{\text{cert}} = \text{PoK}\{\alpha_0, \beta_0, \delta : \bar{B}_{j^*} = g_0^{\alpha} {\text{ipk}'_{j^*}}^{\beta} \bar{A}_{j^*}^{-\delta} \wedge \alpha_0 \neq 0\}, \text{ and}$$
$$\pi_{\text{validity}} = \text{PoK}\{\alpha, \beta, \gamma, \{\tau_i\}_{i\notin\mathcal{D}} : \bar{B} = \bar{A}^{-\alpha} \bar{D}^{\beta} \wedge h_0 \textstyle\prod_{i\in\mathcal{D}} h_i^{\tau_i} = \bar{D}^{\gamma} \prod_{i\notin\mathcal{D}} h_i^{-\tau_i} \wedge \beta \neq 0\},$$

the user sends the verifiable presentation

$$vp = (\{m\}_{i\in\mathcal{D}}, \text{ipk}'_{j^*}, \tilde{\text{ipk}}'_{j^*}, \bar{A}_{j^*}, \bar{B}_{j^*}, \bar{A}, \bar{B}, \bar{D}, \pi_{\text{cert}}, \pi_{\text{validity}}).$$

– $\mathsf{Verify}(\text{ipk}, vp, \mathcal{D}, \{m_i\}_{i\in\mathcal{D}}, \mathcal{P})$: If $|\mathcal{P}| = 1$, the original Verify algorithm is used, so we assume $|\mathcal{P}| > 1$. Parse vp as $\text{ipk}'_{j^*}, \tilde{\text{ipk}}'_{j^*}, \bar{A}_{j^*}, \bar{B}_{j^*}, \bar{A}, \bar{B}, \bar{D}, \pi_{\text{cert}}$, and π_{validity}. Accept if (1) $\bar{A} \neq 1$, (2) verification of the PoKs succeeds, and (3) it holds that:

$$\mathbf{e}(\text{ipk}'_{j^*}, \tilde{g}) = \mathbf{e}(g, \tilde{\text{ipk}}'_{j^*})$$
$$\mathbf{e}(\bar{B}_{j^*}, \tilde{h}) = \mathbf{e}(\bar{A}_{j^*}, \text{vpk})$$
$$\mathbf{e}(\bar{B}, \tilde{g}) = \mathbf{e}(\bar{A}, \tilde{\text{ipk}}'_{j^*}) \ .$$

5.2 Security Analysis

We begin by showing that our construction's presentation constitutes a proof that the user knows a valid credential on the attributes requested by the verifier, tha has been issued by an authorized issuer. To achieve this, we will demonstrate both the *correctness* and *soundness* of our presentation protocol.

Proof of Knowledge of a Valid Credential

Completeness.

Proof. (sketch) Let $(\mathrm{ipk} = g^{y_{j^*}}, \tilde{\mathrm{ipk}}_{j^*} = \tilde{g}^{y_{j^*}})$ be the public key of an authorized issuer and $\sigma_{j^*} = (A_{j^*}, e_{j^*})$ be a certificate certifying its public key. Let $\sigma = (A, e)$ be a credential under the public key $(\mathrm{ipk} = g^{y_{j^*}}, \tilde{\mathrm{ipk}}_{j^*} = \tilde{g}^{y_{j^*}})$ on a honest user attributes $\boldsymbol{m}$. Let $vp = (\{m\}_{i\in\mathcal{D}}, \mathrm{ipk}'_{j^*}, \tilde{\mathrm{ipk}}'_{j^*}, \bar{A}_{j^*}, \bar{B}_{j^*}, \bar{A}, \bar{B}, \bar{D}, \pi_{\mathrm{cert}}, \pi_{\mathrm{validity}})$ be the VP generated by the user during the presentation protocol. As the user honestly computed π_{cert} and π_{validity}, their verification will succeed. We therefore only have to check wether the following equality still hold:

$$\mathbf{e}(\mathrm{ipk}'_{j^*}, \tilde{g}) = \mathbf{e}(g, \tilde{\mathrm{ipk}}'_{j^*}) \tag{4}$$

$$\mathbf{e}(\bar{B}_{j^*}, \tilde{h}) = \mathbf{e}(\bar{A}_{j^*}, \tilde{\mathrm{vpk}}) \tag{5}$$

$$\mathbf{e}(\bar{B}, \tilde{g}) = \mathbf{e}(\bar{A}, \tilde{\mathrm{ipk}}'_{j^*}). \tag{6}$$

The Equation (4) trivially verifies since ipk'_{j^*} and $\tilde{\mathrm{ipk}}'_{j^*}$, are randomized versions of the same public key respectively in the group $\mathbb{G}_1$, and $\mathbb{G}_2$. For Equation (6), since the vp was correctly computed, we know by construction that $\bar{B} = \bar{A}^{ty_{j^*}}$ therefore $\mathbf{e}(\bar{B}, \tilde{g}) = \mathbf{e}(\bar{A}^{ty_{j^*}}, \tilde{g}) = \mathbf{e}(\bar{A}, \tilde{g}^{ty_{j^*}}) = \mathbf{e}(\bar{A}, \tilde{\mathrm{ipk}}'_{j^*})$, therefore Equation (6) is verified.
Since the vp was correctly computed, we also know by construction that $\bar{B}_{j^*} = \bar{A}^x_{j^*}$ therefore $\mathbf{e}(\bar{B}_{j^*}, \tilde{h}) = \mathbf{e}(\bar{A}^x_{j^*}, \tilde{h}) = \mathbf{e}(\bar{A}_{j^*}, \tilde{h}^x) = \mathbf{e}(\bar{A}_{j^*}, \tilde{\mathrm{vpk}})$, and Equation (5) also verifies. Thus, our construction satisfies the correctness property.

Soundness For the soundness property, we need to prove that whenever the verifier accepts a VP from a user, then the user (who can be malicious) knows a valid VC on the disclosed attributes $\mathcal{D}$, issued by an authorized issuer.

Proof. (sketch) Let us consider a VP $vp = (\{m\}_{i\in\mathcal{D}}, \mathrm{ipk}'_{j^*}, \tilde{\mathrm{ipk}}'_{j^*}$,
$\bar{A}_{j^*}, \bar{B}_{j^*}, \bar{A}, \bar{B}, \bar{D}, \pi_{\mathrm{cert}}, \pi_{\mathrm{validity}})$, that is presented by a potentially malicious user, during a presentation to the verifier, such that the equations Equation (4), and Equation (6), Equation (5) are true.
From the knowledge extractors of the ZKPKs π_{cert}, and π_{validity} we can recover the values $(\alpha, \beta, \gamma, \delta, \{m_i\}_{i\notin D})$, and from there we have the attribute $\boldsymbol{m}$, and the VC is of the form $\mathrm{cred} = ((A^*, e^*), \boldsymbol{m})$. Now it remains to find A^* and e^* in order to recover the VC. If $\gamma = 0$ from the second equation in π_{validity} we have $h_0 \prod h_i^{m_i} = \bar{D}^0 = 1_{\mathbb{G}_1}$, which imply that the user would know a representation of h_0 in the base $(h_i)_{i\in[n]}$. This is impossible unless he is able to solve the discrete logarithm problem. If $\gamma \neq 0$, from the second equation of π_{validity} we have $\bar{B} = \bar{A}^{-\alpha}\bar{D}^{\beta}$ and from Equation (6) we have $\mathbf{e}(\bar{B}, \tilde{h}) = \mathbf{e}(\bar{A}, \tilde{\mathrm{ipk}}'_{j^*})$, which implies that $\bar{B} = \bar{A}^{y'_{j^*}}$, where we denote by y'_{j^*} the discrete logarithm of the $\tilde{\mathrm{ipk}}'_{j^*}$ in the base g. Therefore we have $\bar{A}^{y'_{j^*}+\alpha} = \bar{D}^{\beta}$. We know from π_{validity} that $\beta \neq 0$. We also have $y'_{j^*} + \alpha \neq 0$, otherwise we would have $\bar{D}^{\beta} = 0$, that is $(h_0 \prod_i h_i^{m_i})^{\beta} = 0$. The user would therefore know a representation of h_0 in the

base $(h_i)_{i\in[n]}$. This is impossible unless, he is able to solve the discrete logarithm problem.

To summarize, we have $\bar{A}^{y'_{j^*}+\alpha} = (h_0 \prod_i h_i^{m_i})^{\frac{\beta}{\gamma}}$. This means that $(A^* = \bar{A}^{\frac{\gamma}{\beta}}, \alpha)$ is a valid BBS signature issued by an issuer whose secrete key is $\text{isk} = y'_{j^*}$.

So far we have proved that if the verifier accept a VP from a user, then the user possesses a valid VC on the disclosed attribute $\mathcal{D}$, issued by the issuer with a public key $(\text{ipk} = g^{y_{j^*}}, \tilde{\text{ipk}}_{j^*} = \tilde{g}^{y_{j^*}})$ (*). Now it remains to show that the public key of that issuer was certified by the certification authority, meaning that it is included in the policy.

From the equality in π_{cert} and the equation Equation (5) we have $\bar{B}_{j^*} = \bar{A}^x_{j^*} = g_0^{\alpha_0} \text{ipk}'^{\ \beta_0}_{j^*} \bar{A}^{-\delta}_{j^*}$ (the Greek letters here are from π_{cert}). We know from Equation (4), that $\text{ipk}'_{j^*} = g^{y'_{j^*}}$, and from π_{cert} that $\alpha_0 \neq 0$. Therefore, we have $(\bar{A}^{\alpha_0^{-1}}_{j^*})^{x+\delta} = g_0 g^{\frac{\beta_0}{\alpha_0} y'_{j^*}}$. We necessarily have $x + \delta \neq 0$, otherwise this would mean that $x = \delta$, and that the user knows the secret key of the certification authority (RP in some cases). From this we can extract a valid signature $(A_j, e_j) = (\bar{A}^{\alpha_0^{-1}}_{j^*}, \delta)$ on $y_{j^*} = \frac{\beta_0}{\alpha_0} y'_{j^*}$. Thus, this proves that the public key of the putative issuer was certified by the verifier using its secret key $\text{vsk} = x$ (**).

(*) and (**) show that if the verifier accepts a presentation, it means that the user possesses a valid credential (namely, $\text{cred} = (\hat{A} = A^{*\frac{\alpha_0}{\beta_0}}, \frac{\alpha_0}{\beta_0}\alpha)$ issued by an authorized issuer (namely the issuer with $\text{ipk}'_{j^*} = g^{y'_{j^*}}$, and certificate (A_j, e_j)). Thus, this proves the soundness of our issuer-hiding anonymous credential system.

Issuer-Hiding Unforgeability. Since we use *the signed-policy approach* introduced by Bobolz et al. [6], in which authorized public keys are signed using an EUF-CMA signature scheme, their generic security proof applies to our construction.

Intuitively, the adversary has two potential ways of breaking unforgeability: (1) they can forge a certificate on a corrupted public key ipk' that is not part of the challenge policy pol*, or (2) they can forge a credential by forging a signature with respect to an honest issuer's public key (ipk_i).

In the scheme proposed by Bobolz et al. [6], a structure-preserving signature scheme (SPS) is used to certify the public keys of authorized issuers. Therefore, for a type-1 adversary, if the reduction can retrieve the corrupted key ipk', this constitutes a forgery against the SPS.

Since for our construction, we use BBS to certify the public keys of authorized issuers, we need to retrieve both the corrupted public key ipk' and its corresponding private key isk' to obtain a forgery against BBS. We show that, in the algebraic group model (AGM), a type-1 adversary necessarily knows isk' and that our reduction can retrieve this value (which constitutes a forgery against BBS).

We therefore provide a proof of our construction in the AGM.

Theorem 1. *If π_{setup}, π_{user}, π_{cert}, and π_{validity} are straight-line extractable* ZKPOK*, and BBS is EUF-CMA-secure, then our protocol in Section 5 is unforgeable in the AGM with advantage:*

$$\mathsf{Adv}^{\text{uf}}_{\mathcal{A}}(1^\lambda) \leq \mathsf{Adv}^{\text{suf-cma}}_{\mathcal{B},\text{BBS}}(1^\lambda) + \tfrac{1}{n_J}\mathsf{Adv}^{\text{suf}}_{\mathcal{B},\text{BBS}}(1^\lambda) + \mathsf{Adv}^{\text{ksnd}}_{\text{EXT},\pi_{\text{setup}}}(1^\lambda) + \mathsf{Adv}^{\text{ksnd}}_{\text{EXT},\pi_{\text{user}}}(1^\lambda) + \mathsf{Adv}^{\text{ksnd}}_{\text{EXT},\pi_{\text{cert}}}(1^\lambda) + \mathsf{Adv}^{\text{ksnd}}_{\text{EXT},\pi_{\text{validity}}}(1^\lambda).$$

The proof of Theorem *Theorem 1* is presented in the Appendix Section C.1.

Issuer-Hiding Anonymity. For the issuer-hiding anonymity, we adapted the proof from [31] to our construction. We observe that issuer-hiding anonymity holds unconditionally for our scheme.

Theorem 2. *In the RO model, our construction satisfies the issuer-hiding anonymity requirement.*

The proof of Theorem *Theorem 2* is presented in he Appendix Section C.2.

We also introduce the first pairing-free issuer-hiding BBS-based anonymous credential system. For the sake of conciseness, the full description of this variant are deferred to Appendix Section 6.

6 Extension to a Pairing-free Scheme

In addition to our pairing-based construction, we propose the first pairing-free issuer-hiding anonymous credential scheme based on BBS signatures. This construction avoids pairings by leveraging the approach introduced by Desmoulins et al. [25]. This pairing-free design is of independent interest because it may represent a potential candidate for integration into the EUDIW. The Architecture Reference Framework (ARF) document [29] for EUDIW identifies the issuer-hiding feature as a requirement. Moreover, since it is based on BBS signatures which are currently being standardized by the IETF (see draft [39]), this adds even more value to our construction, which will be easier to deploy in practice.

6.1 Construction

We now describe how to achieve pairing-free issuer-hiding BBS-based anonymous credentials using the technique by Desmoulins et al. [25] and the verifier-signed policy approach [6]. The remaining algorithms are defined as follows:

- $\mathsf{Setup}(1^\lambda, n)$: On input a security parameter 1^λ and a bound n on the number of attributes to certify, return the public parameters $\text{pp} = (p, \mathbb{G}_1, g, g_0, h, h_0, h_1, \ldots h_n)$, where $\mathbb{G}_1$ is a cyclic group of prime order p, and $g_0, h, h_0, h_1, \ldots, h_n \leftarrow\!\!\$\ \mathbb{G}_1$ are $n+4$ generators of $\mathbb{G}_1$.

– $\mathsf{IKeyGen}(\text{pp})$: select $y \in_\$ \mathbb{Z}_p^*$, and return $(\text{ipk}, \text{isk}) = (g^y, y)$. The issuer also computes a ZKPK $\pi_{\text{key}} = \text{PoK}\{\alpha : \text{ipk} = g^\alpha\}$.

– $\mathsf{Issue}(\text{isk}, \{m_i\}_{i=1}^n)$: Let $\text{isk} = y$, compute $C_m = h_0 \prod_i h_i^{m_i}$. Select $e \leftarrow\!\!\$\ \mathbb{Z}_p \backslash \{y\}$, and compute $A = C_m^{\frac{1}{y+e}}$. The issuer also build a ZKPK $\pi_{\text{issue}}^{\text{dleq}} := \text{PoK}\{\alpha : B = A^\alpha \wedge \text{ipk} = g^\alpha\}$ where $B = C_m A^{-e} = A^{\text{isk}}$. Then the issuer returns the pair (A, e) along with the proof $\pi_{\text{issue}}^{\text{dleq}}$ to the user.

– $\mathsf{VerifyCred}(\text{ipk}, \{m_i\}_{i=1}^n, \sigma = (A, e))$: Compute $C_m = h_0 \prod_i h_i^{m_i}$. If $B = 1$ return 0. Otherwise, if proof $\pi_{\text{issue}}^{\text{dleq}}$ is valid return 1 else return 0.

– $\mathsf{KeyGen}(\text{pp})$: select $x \in_\$ \mathbb{Z}_p^*$, and returns $(\text{vpk}, \text{vsk}) = (h^x, x)$.

– $\mathsf{GenPolicy}(\mathcal{P})$: Parse $\mathcal{P}$ as $\{(\text{ipk}_j\}_{j\in[\ell]}$. Run $(\text{vpk}, \text{vsk}) \leftarrow\!\!\$\ \mathsf{KeyGen}(pp)$. For $j \in [\ell]$ verify the provided ZKPK π_{key} and if the verification succeeds, certify the issuer j, by computing

$$\sigma_j = (A_j, e_j) = ((g_0 \text{ipk}_j)^{\frac{1}{x+e_j}}, e_j), \text{ where } e_j \in \mathbb{Z}_p \backslash \{-x\}.$$

This algorithm also computes a ZKPK $\pi_{\text{cert}}^{\text{dleq}} := \text{PoK}\{\alpha : B_j = A_j^\alpha \wedge \text{vpk} = h^\alpha\}$ where $B_j = g_0 \text{ipk}_j A_j^{-e_j} = A_j^{\text{vsk}}$. We assume that for $j \in [\ell]$ the verification of π_{key} succeeds. Output the presentation policy

$$\text{pol} = (\text{vpk}, \{(\text{ipk}_j, \sigma_j, \pi_{\text{cert,j}}^{\text{dleq}})\}_{j\in[\ell]}).$$

– $\mathsf{VerifyPolicy}(\mathcal{P}, \text{pol})$: Parse pol as $\text{vpk}, \{(\text{ipk}_j, \sigma_j, \pi_{\text{cert,j}}^{\text{dleq}})\}_{j\in[\ell]})$. For $j \in [\ell]$ set $B_j = g_0 \text{ipk}_j A_j^{-e_j} = A_j^x$. Accept if $\ell = |\mathcal{P}|$, and for all $j \in [\ell], \pi_{\text{cert,j}}^{\text{dleq}}$ is valid.

– $\mathsf{Show}(\text{ipk}_{j^*}, \sigma, \mathcal{D}, \{m_i\}_{i=1}^n, \text{pol}, \mathcal{P})$: If $|\mathcal{P}| = 1$, the Show algorithm in [25] is used, so we assume $|\mathcal{P}| > 1$. Parse $\mathcal{P}$ as $\{\text{ipk}_j\}_{j\in[\ell]}$, σ as (A, e), and σ_{j^*} as (A_{j^*}, e_{j^*}). Select $t, r, r_1, r_2 \leftarrow\!\!\$\ \mathbb{Z}_p^*$. Compute $\text{ipk}'_{j^*} = g^{t y_{j^*}}$, the credential and certificate randomization is the same as in Section 5. In addition to the two zero-knowledge proofs, π_{cert} and π_{validity}, compute $\pi_{\text{rand-cred}}^{\text{dleq}}$

$$\pi_{\text{rand-cred}}^{\text{dleq}} = \text{PoK}\{\alpha : \bar{B} = \bar{A}^\alpha \wedge \text{ipk}'_{j^*} = g^\alpha\},$$

where $(\alpha = y'_{j^*} = t y_{j^*})$, t (respectively y_{j^*}) is known only by the user (respectively the issuer). Since the user does not know y_{j^*}, they cannot compute this proof / OIP ($\pi_{\text{rand-cred}}^{\text{dleq}}$) themselves. Therefore, they must obtain it anonymously from the issuer. This OIP can be generated using the technique presented by Orrù et al. in [42]. For the sake of simplicity, this technique is not detailed in this paper.

$$vp = (\{m\}_{i\in\mathcal{D}}, \text{ipk}'_{j^*}, \bar{A}_{j^*}, \bar{B}_{j^*}, \bar{A}, \bar{B}, \bar{D}, \pi_{\text{cert}}, \pi_{\text{validity}}, \pi_{\text{rand-cred}}^{\text{dleq}}).$$

– $\mathsf{Verify}(\mathsf{ipk}, vp, \mathcal{D}, \{m_i\}_{i\in\mathcal{D}}, \mathcal{P})$: If $|\mathcal{P}| = 1$, the original Verify algorithm in [25] is used, so we assume $|\mathcal{P}| > 1$. Parse vp as $\mathsf{ipk}'_{j^*}, \bar{A}_{j^*}, \bar{B}_{j^*}, \bar{A}, \bar{B}, \bar{D}, \pi_{\text{cert}}$, and π_{validity}. Accept if (1) $\bar{A} \neq 1$, (2) verification of the PoKs π_{cert}, π_{validity}, and $\pi^{\text{dleq}}_{\text{rand}-\text{cred}}$ succeeds, and (3) it holds that:

$$\bar{B}_{j^*} = \bar{A}_{j^*}^{\text{ysk}},$$

this last verification can be performed by the RP provided it acts as its own certification authority.

Remark 6. It is important to note that the same approach can be extended to PS-based anonymous credentials [44] and to PS-MAC [19] in order to achieve, respectively, pairing-based issuer-hiding anonymous credentials and pairing-free anonymous credentials. This can be done using structure-preserving signatures, such as the one proposed by Fuchsbauer et al. [32], for the policy definition. However, since the protocols based on BBS signatures appear to be more appealing within the context of EUDIW, and considering the ongoing standardization process of BBS signatures at the IETF, we have chosen to focus this work on BBS.

6.2 Security Analysis

Unforgeability. The proof of unforgeability can be derived from that of the pairing-based construction, but it is much simpler. This is because, for the type-1 adversary, it is easy to recover y_{j^*} using the OIP extractor.

Issuer-hiding Anonymity. The issuer-hiding anonymity proof also follows from that of the pairing-based construction.

7 Accountable Issuer Hiding

Issuer hiding is a desirable privacy property in anonymous credential systems: a verifier should be able to check that a credential was issued by one of its accepted issuers, without learning which particular issuer produced it. This property is especially relevant in large-scale identity ecosystems such as the EUDI Wallet, where the issuer's identity may reveal sensitive information about the user, such as their country of residence, administrative affiliation, or the type of institution they interacted with.

However, hiding the issuer also removes an evidentiary link that is normally available in credential systems without issuer hiding. In a standard credential presentation, the issuer's public key is revealed to the relying party (RP), so a fraudulent credential can be directly attributed to the issuer whose key verifies it. In contrast, in an issuer-hiding presentation, the verifier only learns that the credential was issued by *some* authorized issuer. This creates a fundamental tension between privacy and accountability: hiding the issuer protects honest users, but may also prevent the identification of a malicious issuer that deliberately issued a credential on false attributes.

This distinction matters because an authorized issuer can always misbehave at the issuance stage. For example, an issuer entitled to certify age credentials could issue a credential falsely stating that a minor is over eighteen. Cryptography cannot prevent an issuer from lying about the attributes it attests to. What a credential system *can* provide, however, is *accountability*: if a fraudulent presentation is later detected, the system should offer a controlled mechanism for identifying the misbehaving issuer. Without such a mechanism, issuer hiding may grant a malicious issuer plausible deniability and remove the deterrent effect that normally comes from the risk of exposure.

Our constructions described in Section 5.1, addressed attacks in which a malicious issuer shares their key material either in plain, or in a randomized form with a malicious user. A malicious issuer do not need to disclose its secret key, or even an inspectable fraudulent credential, to the user. Instead, it may provide the user with an opaque device, a tamper-resistant token, or a presentation service that internally embeds the fraudulent credential and the necessary signing material. Such a device may output valid presentations on demand while preventing the user from extracting any evidence identifying the issuer.

This observation motivates a stronger and more explicit security notion, which we call *Accountable Issuer Hiding* (AIH). Informally, AIH augments traditional issuer hiding with an *opening* capability that allows a designated authority to identify the issuer that produced the credential underlying a given presentation, in case of suspected fraud. The goal is *not* to reveal the issuer's identity to every verifier during ordinary presentations. Rather, the goal is to ensure that every accepted presentation contains sufficient cryptographic evidence to identify the responsible issuer under a controlled opening procedure.

Our notion is intentionally weaker than full black-box traitor tracing [10, 40, 8, 9], and does not attempt to identify a malicious issuer solely through oracle access to a decoder device. Instead, we require that every accepted presentation contains enough information to identify the issuer that enabled it. This notion is specifically tailored to issuer-hiding anonymous credentials and addresses the practical accountability issues arising in such systems.

Informal definition. An issuer-hiding anonymous credential scheme satisfies Accountable Issuer Hiding if it provides an additional algorithm Open which, given a dedicated opening key and an accepted presentation, outputs the public key of the issuer who produced the credential underlying that presentation. More precisely, the following properties should hold informally:

- **Opening correctness.** Every presentation generated from an honestly issued credential opens to the public key of the issuer that issued the credential, and this public key belongs to the verifier's accepted policy.
- **Opening accountability.** For every accepted presentation generated by any adversary—including adversaries colluding with one or more authorized issuers—the opening algorithm outputs the public key of an issuer that (i) belongs to the verifier's policy, and (ii) issued the credential enabling that presentation.

– **Opening soundness.** No adversary should be able to generate an accepted presentation that opens to $\perp$, to an issuer outside the verifier's accepted policy, or to an issuer unrelated to the credential used to generate the presentation.

Design intuition. Our approach follows the same general philosophy as group signatures [7] and traceable signatures [37]. In these primitives, a signer remains anonymous during normal verification, but a designated opening authority can later identify the signer in case of abuse. We adapt this idea to issuer-hiding anonymous credentials.

Concretely, the user includes in every presentation an encryption of the issuer's public key under the public key of an opening authority. To avoid concentrating excessive power in a single entity, this opening key may be threshold-shared among several independent authorities, so that issuer identification requires a joint and auditable decision.

The presentation proof cryptographically binds this ciphertext to the credential being presented. More precisely, the user proves in zero knowledge that:

– it possesses a valid credential on the disclosed attributes;
– this credential was issued under the issuer public key encrypted in the ciphertext; and
– the encrypted issuer public key belongs to the verifier's policy.

The verifier therefore learns neither the issuer's identity nor the plaintext encrypted in the ciphertext, yet is assured that the hidden issuer's public key belongs to the authorized policy and is cryptographically bound to the credential being presented.

This binding provides accountability. If a fraudulent presentation is later detected, the opening authority decrypts the ciphertext contained in the presentation to recover the issuer's public key, and the misbehaving issuer can then be identified from the verifier's policy.

Crucially, a malicious issuer cannot evade exposure by providing the user with an opaque device that generates presentation proofs as a service, without the issuance keys ever leaving the device. Any valid presentation produced by such a device must still contain an encryption of the issuer's public key under which the credential verifies, and the zero-knowledge proof enforces this binding. Therefore, even if the user never obtains the issuer's secret key or an inspectable credential, a fraudulent presentation can still be attributed to the issuer that generated it.

Our construction does not attempt to prevent dishonest issuance itself, which is inherently impossible when issuers are trusted to certify real-world attributes. Instead, it restores the accountability that issuer hiding would otherwise remove: honest users continue to benefit from issuer privacy during ordinary presentations, while malicious issuers lose the ability to exploit issuer hiding to deny wrongdoing.

7.1 Construction

We now describe how to extend our pairing-based and pairing-free issuer-hiding constructions (Section 5.1) into accountable issuer-hiding systems. Two additional algorithms are introduced:

– OKeyGen, which generates the key pair of the trusted opening authority (also called the *opener*); and
– Open, which, given the opening key and an accepted presentation, outputs the public key of the issuer who produced the underlying credential.

The algorithms Setup, Issue, and VerifyCred are identical to those described in Section 2.6 and are omitted for brevity. The remaining algorithms are defined as follows.

– IKeyGen(pp): Same as in Section 5.1.
– KeyGen(pp): Same as in Section 5.1.
– OKeyGen(pp): Select $x_T \in_\$ \mathbb{Z}_p^*$ and let $g_T \in_\$ \mathbb{G}_1$ be a fresh random generator. Return

$$(\text{opk}, \text{osk}) \;=\; \big(g_T^{x_T},\; x_T\big).$$

Set $h_T = g_T^{x_T} = \text{opk}$. The opening key osk may be threshold-shared among several independent authorities to avoid concentrating excessive power in a single entity.
– GenPolicy($\mathcal{P}$): Same as in Section 5.1.
– VerifyPolicy($\mathcal{P}$, pol): Same as in Section 5.1.
– Show($\text{ipk}_{j^*}, \sigma, \mathcal{D}, \{m_i\}_{i=1}^n, \text{pol}, \mathcal{P}$): If $|\mathcal{P}| = 1$, the original Show algorithm is used; we therefore assume $|\mathcal{P}| > 1$. Parse $\mathcal{P}$ as $\{\tilde{\text{ipk}}_j\}_{j\in[\ell]}$, σ as (A, e), and σ_{j^*} as (A_{j^*}, e_{j^*}). Select $t, r, r_1, r_2, f, w \in_\$ \mathbb{Z}_p^*$ and proceed as follows.

1. **Randomize the issuer key and certificate.** Compute

$$\text{ipk}'_{j^*} = g^{t y_{j^*}}, \qquad \tilde{\text{ipk}}'_{j^*} = \tilde{g}^{t y_{j^*}},$$

$$\bar{A}_{j^*} = A_{j^*}^{rt}, \qquad \bar{B}_{j^*} = g_0^{t'}\, {\text{ipk}'_{j^*}}^{r}\, \bar{A}_{j^*}^{-e_{j^*}} = \bar{A}_{j^*}^{x},$$

   where $t' = tr$.
2. **Randomize the credential.** Let $\ell = r_1 r_2$, $\hat{t} = t\ell$, and $e' = te$. Compute

$$\bar{A} = A^{\ell}, \qquad \bar{D} = C_m^{r_2}, \qquad \bar{B} = C_m^{\hat{t}}\, \bar{A}^{-e'} = \bar{A}^{t y_{j^*}},$$

   where $C_m = h_0 \prod_i h_i^{m_i}$, so that $\bar{B} = \bar{D}^{t r_1}\, \bar{A}^{-e'}$.
3. **Encrypt the issuer's public key.** Compute the ElGamal-style [27] ciphertext

$$C = (C_1, C_2) = \big(g_T^f,\; \text{ipk}_{j^*} \cdot h_T^f\big) = \Big(g_T^f,\; {\text{ipk}'_{j^*}}^{1/t} \cdot h_T^f\Big), \tag{7}$$

   and the re-randomized ciphertext

$$E = (E_1, E_2) = \big(C_1^{t'} \cdot g_T^w,\; C_2^{t'} \cdot h_T^w\big) = \big(g_T^{t' f + w},\; {\text{ipk}'_{j^*}}^{r} \cdot h_T^{t' f + w}\big). \tag{8}$$

4. **Compute the zero-knowledge proofs.**
   - π_{cert}: Same as in Section 5.1.
   - π_{validity}: Same as in Section 5.1.

- $\pi_{\text{accountability}}$:

$$\text{PoK}\Big\{(\mu,\theta,\alpha,\omega,\tau,\beta)\ :$$

$$C_1 = g_T^{\mu} \ \wedge \ C_2 = \text{ipk}'^{\,\theta}_{j^*} \cdot h_T^{\mu} \qquad \text{(enc. well-formedness)}$$

$$E_1 = C_1^{\alpha} \cdot g_T^{\omega} \ \wedge \ E_2 = C_2^{\alpha} \cdot h_T^{\omega} \qquad \text{(re-randomization consistency)}$$

$$E_1 = g_T^{\tau} \ \wedge \ E_2 = \text{ipk}'^{\,\beta}_{j^*} \cdot h_T^{\tau} \qquad \text{(binding to credential)}$$

$$\Big\}$$

Remark. The scalars α and β appearing in $\pi_{\text{accountability}}$ are the same as those used in π_{cert}, ensuring that the encrypted issuer key is bound to the certificate verified in π_{cert}.

The user sends the verifiable presentation:

$$vp = (\{m_i\}_{i\in\mathcal{D}}, \text{ipk}'_{j^*}, \tilde{\text{ipk}}'_{j^*}, \bar{A}_{j^*}, \bar{B}_{j^*}, \bar{A}, \bar{B}, \bar{D}, C_1, C_2, E_1, E_2, \pi_{\text{cert}}, \pi_{\text{validity}}, \pi_{\text{accountability}}).$$

– $\mathsf{Open}(\text{osk}, \text{opk}, vp)$: Parse vp and retrieve (C_1, C_2). Return

$$\text{ipk} \ = \ \frac{C_2}{C_1^{\text{osk}}}.$$

– $\mathsf{Verify}(vp, \mathcal{D}, \{m_i\}_{i\in\mathcal{D}}, \mathcal{P})$: Same as in Section 5.1, with the additional step of verifying $\pi_{\text{accountability}}$.

7.2 Security Discussion

We now argue informally that if the verifier accepts a presentation vp, then:

1. the user possesses a valid credential on the disclosed attributes;
2. this credential was issued under the issuer public key encrypted in the ciphertext;
3. the encrypted public key belongs to the verifier's accepted policy;
4. the opening algorithm recovers exactly this public key.

Consequently, any fraudulent presentation can be unambiguously attributed to the authorized issuer that enabled it.

Theorem 3 (Informal Accountability Guarantee). *Assuming the knowledge soundness of the zero-knowledge proofs π_{cert}, π_{validity}, and $\pi_{\text{accountability}}$, and the correctness of the ElGamal encryption scheme [27], every accepted presentation opens to the public key of an issuer that:*

1. *belongs to the verifier's accepted policy, and*
2. *issued the credential underlying the presentation.*

Proof (Proof sketch). From the knowledge-soundness argument of Section 5.2, whenever the verifier accepts vp, the user possesses a valid credential issued by an authorized issuer whose (de-randomized) public key is

$$\mathrm{ipk}_{j^*} \;=\; {\mathrm{ipk}'_{j^*}}^{1/\beta} \cdot g^{y'_{j^*}\,\beta/\alpha},$$

where α and β are the witnesses extracted from π_{cert}.

Let $\bar{\mathrm{ipk}}$ denote the key returned by Open. From relation (7) and the first statement of $\pi_{\mathrm{accountability}}$, we have

$$\bar{\mathrm{ipk}} \;=\; \frac{C_2}{C_1^{\mathrm{osk}}} \;=\; {\mathrm{ipk}'_{j^*}}^{\theta}, \tag{9}$$

for the witness θ extracted from $\pi_{\mathrm{accountability}}$.

The re-randomization consistency relation then gives

$$E_2 \;=\; {\mathrm{ipk}'_{j^*}}^{\alpha\theta} \cdot h_T^{\alpha\mu+\omega}.$$

Combined with the binding-to-credential relation, this yields

$${\mathrm{ipk}'_{j^*}}^{\alpha\theta} \cdot h_T^{\alpha\mu+\omega} \;=\; {\mathrm{ipk}'_{j^*}}^{\beta} \cdot h_T^{\tau},$$

which, in the exponent, reads

$$y'_{j^*}\,\alpha\theta \;=\; y'_{j^*}\,\beta \qquad \text{and} \qquad \alpha\mu+\omega \;=\; \tau. \tag{10}$$

Since $\alpha \neq 0$ (enforced by π_{cert}) and breaking the discrete logarithm assumption is infeasible, equation (10) implies $\theta = \beta/\alpha$. Substituting into (9)) gives

$$\bar{\mathrm{ipk}} \;=\; {\mathrm{ipk}'_{j^*}}^{\beta/\alpha} \;=\; g^{y'_{j^*}\,\beta/\alpha} \;=\; \mathrm{ipk}_{j^*}.$$

Therefore, the opening algorithm returns exactly ipk_{j^*}, the public key of the authorized issuer that produced the credential underlying the presentation. This concludes the proof.

A formal treatment of Accountable Issuer Hiding, including a complete security model and formal proofs, is left for future work.

8 Implementation and Performance Evaluation

We present the implementation results of our issuer-hiding protocol and evaluate its practical efficiency.

8.1 Comparative Analysis

In this section, we analyze the computational complexity of our paring-free BBS based issuer-hiding scheme and compare it to recent schemes by Flamini et al [31] (namely IHBBS_1), and by Katz and Sefranek [36], focusing on computational complexity and size.

Size Complexity Table 1 summarizes the sizes of the various elements across the respective constructions. When comparing the policies denoted in the table as $\mathrm{pk}_{\mathrm{pol}}$, we observe that all three constructions have similar sizes, though our construction is slightly less compact. However, we note that, unlike the other two approaches, ours is not vulnerable to the attacks described in this paper. Therefore, it is a better choice for real-world applications where security is paramount, such as the EUDI wallet.

Table 1: Size complexities 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.

Scheme	$\mathrm{pk}_{\mathrm{pol}}$	$\mathrm{sk}_{\mathrm{pol}}$	Proof
[36]	$(1+\lvert\mathcal{P}\rvert)\lvert\mathbb{G}_2\rvert + 3\lvert\mathbb{Z}_p\rvert$	$2\lvert\mathbb{Z}_p\rvert$	$3\lvert\mathbb{G}_1\rvert + 1\lvert\mathbb{G}_2\rvert + (3+n)\lvert\mathbb{Z}_p\rvert$
[31]	$\lvert\mathcal{P}\rvert\lvert\mathbb{G}_2\rvert + 2\lvert\mathbb{G}_1\rvert$	✗	$2\lvert\mathbb{G}_1\rvert + 2\lvert\mathbb{G}_2\rvert + (n+1)\lvert\mathbb{Z}_p\rvert$
Our paper	$(1+\lvert\mathcal{P}\rvert)\lvert\mathbb{G}_1\rvert + 2\lvert\mathcal{P}\rvert\lvert\mathbb{Z}_p\rvert$	✗	$6\lvert\mathbb{G}_1\rvert + (n+6)\lvert\mathbb{Z}_p\rvert$

Computational Complexity Table 2 focuses on the computational cost of critical operations specifically, proof generation and verification, across the three schemes. All three protocols exhibit comparable efficiency in terms of proof generation, though the construction by Katz and Sefranek is slightly more efficient than the other two. However, our verification procedure is more efficient, as it avoids pairing computations entirely.
Overall, regarding computational complexity, our protocol achieves the lowest total cost compared to the schemes of Flamini et al., and Katz and Sefranek. By offering competitive performance in proof generation while substantially reducing verification overhead, our approach is particularly well-suited for practical, resource constrained devices (which do not support pairing computations) and real-world applications such as the forthcoming EUDI Wallet, which relies on (resource constrained) trusted phone hardware to achieve the so-called *holder-binding* feature. This feature states that only the legitimate holder of a credential shall be able to perform transactions with that credential. In practice, this is achieved by binding that credential to a private key stored in a trusted hardware (or Secure Element) of the credential holder's mobile device and making presentation of such a credential impossible without that private key.

In conclusion, although our scheme is slightly less compact, it is more efficient than those recently introduced by Flamini et al. [31] and Katz and Sefranek in [36]. Unlike the other two schemes, ours is not vulnerable to the attacks described in this paper and is better suited to settings such as the EUDI Wallet, which relies on trusted phone hardware that does not support bilinear maps and pairing-friendly curves.

Table 2: Computational complexities in terms of exponentiations in $\mathbb{G}_1, \mathbb{G}_2$, and pairing operations $\mathbf{p}$, assuming one disclosed attribute.

Scheme	Proof generation	Verification
[36]	$(n+6)exp_1 + 1exp_2$	$(n+5)exp_1 + 2\mathbf{p}$
[31]	$(n+5)exp_1 + 2exp_2$	$(n+1)exp_1 + 2\mathbf{p}$
Our paper	$(n+14)exp_1$	$(n+13)exp_1$

8.2 Benchmarks

Our construction extends two well-established anonymous credential schemes: the pairing-free variant extends BBS# [25], a variant of BBS designed to achieve strong holder binding by storing the user's private key in a hardware secure element (such as the Android Keystore or the iOS Secure Enclave) and splitting cryptographic operations between the secure element and the wallet. The pairing-based variant extends the classical BBS [47] scheme. In both cases, our contribution consists in adding the *issuer-hiding* property to these protocols, enabling the user to present credentials without revealing the identity of the issuer.

Experimental Setup. Wallet operations were measured on a Samsung S24 Ultra (SM-S928B) with a 3.4GHz Octa-core processor running Android 15. This device embeds a *StrongBox*-backed Keystore offering hardware security comparable to a smartcard, where private keys are securely managed and never exposed outside the secure element. Biometric authentication can optionally be required for each cryptographic operation. However, the StrongBox only supports the NIST-256 (secp256r1) curve, which motivated our pairing-free design. Issuer and verifier operations were measured on a PC equipped with an Intel Core i7-8565U CPU @ 1.80GHz running Linux. All cryptographic operations were performed using our C++ library built on top of the MCL library[6]. Benchmarks were obtained through a Java wrapper around this library. Each reported value is the average over 1000 executions, without parallelization, and includes the cost of binary serialization and deserialization of cryptographic artifacts. We assume the issuer certifies 10 user attributes, of which only one is revealed during presentation.

Pairing-Free Variant: Extending BBS# [25]. BBS# is a variant of BBS [47] that achieves strong holder binding by anchoring the user's credential to a key pair stored in a hardware secure element. The protocol is split between the secure element, which handles raw ECDSA signature generation (*NONEwithECDSA*), and the wallet, which performs public key and signature randomization as well as all remaining BBS#-specific operations. Our pairing-free issuer-hiding protocol builds upon this architecture and adds the issuer-hiding property to BBS#.

Table 3 reports the timings for each step of the protocol. The blind validity proof generation protocol between the user and the issuer was run in batch

[6] https://github.com/herumi/mcl

mode with 10 proofs in parallel. Importantly, the credential randomization (step 1) and the associated blind validity proofs are intended to be precomputed in advance, so that during an actual presentation, the wallet only needs to: (1) request an ECDSA signature from the Keystore and randomize the resulting signature and public key, and (2) perform *anonymize credentials step 2*. The most costly operation is the NONEwithECDSA signature performed by the StrongBox (74ms), which nonetheless remains imperceptible to the user. The total VP verification time on PC amounts to only 2ms, confirming the practical efficiency of our construction.

Table 3: Timings (ms) of the pairing-free issuer-hiding protocol on the NIST-256 curve (wallet on smartphone, issuer/verifier on PC).

Operation	**Time (ms)**
PKBlind and Blind Signature Generation	
ECDSA public key randomization (wallet)	0.18
ECDSA prepare data to sign (wallet)	0.02
NONEwithECDSA signature generation (StrongBox)	74
ECDSA signature randomization (wallet)	0.005
Issuance (10 attributes)	
BBS# signature over attributes (issuer)	0.53
Verification of issuer signature (wallet)	1
Blind Validity Proof Generation (user $\leftrightarrow$ issuer), batch $\times 10$	
Step 1 (wallet)	2.8
Step 2 (issuer)	3.5
Step 3 (wallet)	5.4
Step 4 (issuer)	$<$0.01
Step 5 (wallet)	7.6
VP Generation (1 attribute revealed among 10)	
Anonymize credentials step 1 (wallet, batch $\times$10)	9.2
Anonymize credentials step 2 (wallet)	2.2
VP Verification (PC)	
ECDSA signature verification (Java JCE)	0.7
Issuer blind certificate verification	0.07
Blind validity proof	0.27
Blind credentials signature	0.9
Total verification time	**2**

Pairing-Based Variant: Extending classical BBS. We also provide the benchmarks of our pairing-based variant, which extends classical BBS scheme with the issuer-hiding property. In this setting, pairing operations enable more efficient verification, as the blind validity proof is no longer required. Since the BLS12-381 curve is not supported by the Android Keystore, no ECDSA-related timings are reported for this variant. As shown in Table 4, this variant achieves a total VP verification time of only 3.7ms on PC, demonstrating its competitive efficiency.

Table 4: Timings (ms) of the pairing-based issuer-hiding protocol on the BLS12-381 curve, with pairings for verification (wallet on smartphone, issuer/verifier on PC).

Operation	**Time (ms)**
Issuance (10 attributes)	
BBS signature over attributes (issuer)	0.7
Verification of issuer signature (wallet)	1
VP Generation (1 attribute revealed among 10)	
Anonymize credentials step 1 + 2 (wallet)	4.6
VP Verification (PC)	
Blind credentials (incl. issuer cert.) verification	3.7

9 Conclusion

In this paper, we have highlighted the limitations of the approach introduced by Sanders and Traoré [46], and adopted by Katz and Sefranek [36], to achieve issuer hiding in BBS-based anonymous credential systems. By presenting concrete attack scenarios against the Katz and Sefranek construction, we have demonstrated that these attacks also apply to the Sanders and Traoré scheme, as well as to the constructions of Bosk et al. [11] and Flamini et al. [31]. These attacks are presented within realistic scenarios to illustrate their relevance to real-world deployments, such as the European Digital Identity Wallet (EUDI Wallet).

To address these vulnerabilities, we proposed an improved BBS-based issuer-hiding anonymous credential scheme that adopts the signed-policy approach of Bobolz et al. [6]. Our construction mitigates the identified attacks by relying on well-established cryptographic primitives, eliminates the need for secret policy keys, and enables policy generation to be delegated to a trusted certification authority. Unlike the Katz and Sefranek construction, verification in our scheme does not require any secret values and can be performed by any party, not only a designated verifier. Furthermore, we proved the security of our construction in the Algebraic Group Model (AGM), thereby addressing two of the main open questions raised by Katz and Sefranek [36].

We further observed that issuer hiding inherently creates a tension between privacy and accountability: while concealing the issuer's identity protects honest users during ordinary presentations, it may simultaneously grant plausible deniability to malicious issuers that deliberately certify false attributes. To resolve this tension, we introduced the notion of *Accountable Issuer Hiding* (AIH), which augments traditional issuer hiding with a controlled opening capability. Under AIH, a designated opening authority can identify the issuer responsible for any fraudulent presentation, even in the strongest collusion scenarios where a malicious issuer provides a user with an opaque device that generates valid presentations on demand. Our practical realization of AIH ensures that every accepted presentation contains an encryption of the issuer's public key, bound via zero-knowledge proofs to the credential and the verifier's policy, so that fraudulent presentations always remain attributable to the responsible issuer.

In addition, we introduced the first pairing-free issuer-hiding anonymous credential scheme, based on algebraic MACs and oblivious issuance of proofs. This construction avoids pairings entirely, making it particularly well-suited for resource-constrained devices and broadening the applicability of issuer-hiding credentials beyond pairing-based settings.

The formal security proofs and implementation results confirm that our constructions achieve unforgeability, everlasting issuer-hiding anonymity, and accountable issuer hiding. Since our scheme is fully based on BBS signatures, which are currently being standardized by the IETF [39], it is well-positioned for adoption within the EUDI Wallet framework and for integration into future privacy-preserving identity standards. We believe that our work establishes a solid foundation for practical, secure, and accountable issuer-hiding credential systems, and we hope it will stimulate further research into the interplay between issuer privacy, accountability, and real-world deployability.

Bibliography

[1] Au, M.H., Susilo, W., Mu, Y.: Constant-size dynamic k-taa. In: Security and Cryptography for Networks: 5th International Conference, SCN 2006, Maiori, Italy, September 6-8, 2006. Proceedings 5. pp. 111–125. Springer (2006)

[2] Baldimtsi, F., Lysyanskaya, A.: Anonymous credentials light. In: Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security. pp. 1087–1098 (2013)

[3] Barki, A., Brunet, S., Desmoulins, N., Traoré, J.: Improved algebraic macs and practical keyed-verification anonymous credentials. In: Selected Areas in Cryptography–SAC 2016: 23rd International Conference, St. John's, NL, Canada, August 10-12, 2016, Revised Selected Papers 23. pp. 360–380. Springer (2017)

[4] Berlekamp, E.R.: Factoring Polynomials Over Finite Fields. Bell System Technical Journal **46**(8), 1853–1859 (Oct 1967). https://doi.org/10.1002/j.1538-7305.1967.tb03174.x, https://ieeexplore.ieee.org/document/6768643

[5] Bicer, O., Kupcu, A.: Versatile ABS: Usage Limited, Revocable, Threshold Traceable, Authority Hiding, Decentralized Attribute Based Signatures (2019), https://eprint.iacr.org/2019/203

[6] Bobolz, J., Eidens, F., Krenn, S., Ramacher, S., Samelin, K.: Issuer-Hiding Attribute-Based Credentials. In: Conti, M., Stevens, M., Krenn, S. (eds.) Cryptology and Network Security. pp. 158–178. Springer International Publishing, Cham (2021). https://doi.org/10.1007/978-3-030-92548-2_9

[7] Boneh, D., Boyen, X., Shacham, H.: Short group signatures. In: Annual international cryptology conference. pp. 41–55. Springer (2004)

[8] Boneh, D., Partap, A., Rotem, L.: Accountability for Misbehavior in Threshold Decryption via Threshold Traitor Tracing (2023), https://eprint.iacr.org/2023/1724

[9] Boneh, D., Partap, A., Rotem, L.: Traceable Secret Sharing: Strong Security and Efficient Constructions (2024), https://eprint.iacr.org/2024/405

[10] Boneh, D., Sahai, A., Waters, B.: Fully Collusion Resistant Traitor Tracing with Short Ciphertexts and Private Keys. In: Vaudenay, S. (ed.) Advances in Cryptology - EUROCRYPT 2006. pp. 573–592. Springer. https://doi.org/10.1007/11761679_34

[11] Bosk, D., Frey, D., Gestin, M., Piolle, G.: Hidden Issuer Anonymous Credential. Proceedings on Privacy Enhancing Technologies **2022**(4), 571–607 (Oct 2022). https://doi.org/10.56553/popets-2022-0123, https://petsymposium.org/popets/2022/popets-2022-0123.php

[12] Brands, S.A.: Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy. MIT Press, Cambridge, MA, USA (Aug 2000)

[13] Camenisch, J., Drijvers, M., Lehmann, A.: Anonymous attestation using the strong diffie hellman assumption revisited. In: Trust and Trustworthy

Computing: 9th International Conference, TRUST 2016, Vienna, Austria, August 29-30, 2016, Proceedings 9. pp. 1–20. Springer (2016)
[14] Camenisch, J., Lysyanskaya, A.: An Efficient System for Non-transferable Anonymous Credentials with Optional Anonymity Revocation. In: Pfitzmann, B. (ed.) Advances in Cryptology — EUROCRYPT 2001. pp. 93–118. Springer, Berlin, Heidelberg (2001). https://doi.org/10.1007/3-540-44987-6_7
[15] Camenisch, J., Lysyanskaya, A.: A signature scheme with efficient protocols. In: Security in Communication Networks: Third International Conference, SCN 2002 Amalfi, Italy, September 11–13, 2002 Revised Papers 3. pp. 268–289. Springer (2003)
[16] Camenisch, J., Lysyanskaya, A.: Signature schemes and anonymous credentials from bilinear maps. In: Annual international cryptology conference. pp. 56–72. Springer (2004)
[17] Camenisch, J., Stadler, M.: Efficient group signature schemes for large groups. In: Annual international cryptology conference. pp. 410–424. Springer (1997)
[18] Chairattana-Apirom, R., Harding, F., Lysyanskaya, A., Tessaro, S.: Server-aided anonymous credentials. Cryptology ePrint Archive, Paper 2025/513 (2025)
[19] Chase, M., Meiklejohn, S., Zaverucha, G.: Algebraic macs and keyed-verification anonymous credentials. In: Proceedings of the 2014 acm sigsac conference on computer and communications security. pp. 1205–1216 (2014)
[20] Chaum, D.: Security without identification: Transaction systems to make big brother obsolete. Communications of the ACM **28**(10), 1030–1044 (1985)
[21] Chaum, D.: Showing Credentials Without Identification. In: Pichler, F. (ed.) Advances in Cryptology — EUROCRYPT' 85. pp. 241–244. Springer, Berlin, Heidelberg (1986). https://doi.org/10.1007/3-540-39805-8_28
[22] Chaum, D., Pedersen, T.P.: Wallet databases with observers. In: Annual international cryptology conference. pp. 89–105. Springer (1992)
[23] Connolly, A., Lafourcade, P., Perez Kempner, O.: Improved Constructions of Anonymous Credentials from Structure-Preserving Signatures on Equivalence Classes. In: Hanaoka, G., Shikata, J., Watanabe, Y. (eds.) Public-Key Cryptography – PKC 2022. pp. 409–438. Springer International Publishing, Cham (2022). https://doi.org/10.1007/978-3-030-97121-2_15
[24] Cramer, R., Damgård, I., Schoenmakers, B.: Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols. In: Desmedt, Y.G. (ed.) Advances in Cryptology — CRYPTO '94. pp. 174–187. Springer, Berlin, Heidelberg (1994). https://doi.org/10.1007/3-540-48658-5_19
[25] Desmoulins, N., Dumanois, A., Kane, S., Traoré, J.: Making BBS Anonymous Credentials eIDAS 2.0 Compliant (2025), https://eprint.iacr.org/2025/619
[26] Doerner, J., Kondi, Y., Lee, E., shelat, a., Tyner, L.: Threshold BBS+ Signatures for Distributed Anonymous Credential Issuance (2023), https://eprint.iacr.org/2023/602
[27] ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE transactions on information theory **31**(4), 469–472 (1985)

[28] Elkhiyaoui, K., Caro, A.D., Androulaki, E.: Multi-Issuer Anonymous Credentials Without a Root Authority (2021), https://eprint.iacr.org/2021/1669
[29] EU, c.: Architecture and reference framework - EUDI Wallet (2021), https://eu-digital-identity-wallet.github.io/eudi-doc-architecture-and-reference-framework/1.1.0/arf/
[30] Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Conference on the theory and application of cryptographic techniques. pp. 186–194. Springer (1986)
[31] Flamini, A., Friedrichs, K., Lehmann, A.: Issuer-hiding for BBS anonymous credentials via randomizable keys. Cryptology ePrint Archive, Paper 2026/369 (2026), https://eprint.iacr.org/2026/369
[32] Fuchsbauer, G., Hanser, C., Slamanig, D.: Structure-preserving signatures on equivalence classes and constant-size anonymous credentials. Journal of Cryptology **32**, 498–546 (2019)
[33] Fuchsbauer, G., Kiltz, E., Loss, J.: The algebraic group model and its applications. In: Advances in Cryptology–CRYPTO 2018: 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19–23, 2018, Proceedings, Part II 38. pp. 33–62. Springer (2018)
[34] Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof-systems. In: Providing sound foundations for cryptography: On the work of shafi goldwasser and silvio micali, pp. 203–225 (2019)
[35] Groth, J.: Efficient fully structure-preserving signatures for large messages. Cryptology ePrint Archive, Paper 2015/824 (2015), https://eprint.iacr.org/2015/824
[36] Katz, J., Sefranek, M.: Issuer hiding for BBS-based anonymous credentials. Cryptology ePrint Archive, Paper 2025/2080 (2025), https://eprint.iacr.org/2025/2080
[37] Kiayias, A., Tsiounis, Y., Yung, M.: Traceable Signatures. In: Cachin, C., Camenisch, J.L. (eds.) Advances in Cryptology - EUROCRYPT 2004. pp. 571–589. Springer, Berlin, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24676-3_34
[38] Looker, T., Kalos, V., Whitehead, A., Lodder, M.: The BBS Signature Scheme. Internet-Draft draft-irtf-cfrg-bbs-signatures-05, Internet Engineering Task Force (Dec 2023), work in Progress
[39] Looker, T., Kalos, V., Whitehead, A., Lodder, M.: The BBS Signature Scheme. Internet-Draft draft-irtf-cfrg-bbs-signatures-09, Internet Engineering Task Force (Jul 2025), https://datatracker.ietf.org/doc/draft-irtf-cfrg-bbs-signatures/09/, work in Progress
[40] Maitra, S., Wu, D.J.: Traceable PRFs: Full Collusion Resistance and Active Security. In: Hanaoka, G., Shikata, J., Watanabe, Y. (eds.) Public-Key Cryptography – PKC 2022. pp. 439–469. Springer International Publishing, Cham (2022). https://doi.org/10.1007/978-3-030-97121-2_16
[41] Mir, O., Bauer, B., Griffy, S., Lysyanskaya, A., Slamanig, D.: Aggregate Signatures with Versatile Randomization and Issuer-Hiding Multi-Authority Anonymous Credentials (2023), https://eprint.iacr.org/2023/1016

[42] Orrù, M., Tessaro, S., Zaverucha, G., Zhu, C.: Oblivious issuance of proofs. In: Annual International Cryptology Conference. pp. 254–287. Springer (2024)
[43] Orrù, M.: Revisiting keyed-verification anonymous credentials. Cryptology ePrint Archive, Paper 2024/1552 (2024)
[44] Pointcheval, D., Sanders, O.: Short randomizable signatures. In: Topics in Cryptology-CT-RSA 2016: The Cryptographers' Track at the RSA Conference 2016, San Francisco, CA, USA, February 29-March 4, 2016, Proceedings. pp. 111–126. Springer (2016)
[45] Sanders, O.: Efficient redactable signature and application to anonymous credentials. In: IACR International Conference on Public-Key Cryptography. pp. 628–656. Springer (2020)
[46] Sanders, O., Traoré, J.: Compact Issuer-Hiding Authentication, Application to Anonymous Credential (2023), https://eprint.iacr.org/2023/1845
[47] Tessaro, S., Zhu, C.: Revisiting bbs signatures. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 691–721. Springer (2023)

A Additional Preliminaries

A.1 Security of Anonymous Credentials

In this section give the formal definition of unforgeability and anonymity for ACs, through the experiments described in Figure 2 which use the following oracles along with some global lists that are shared among the oracles: HU, the list containing the identities of *honest* users, and CU, the list containing the identities of *corrupt* users, and UPK, USK respectively the list of users public and secret keys. Following [32, 45], we additionally define the list ATT, that stores $(u, \{m_i\}_{i=1}^n, \sigma)$, each time a credential is generated for user u on $\{m_i\}_{i=1}^n$ by the oracles $\mathcal{O}\mathsf{ObtIssue}$ and $\mathcal{O}\mathsf{Issue}$ below. We say that $(u, \{m_i\}_{i\in\mathcal{D}}, \sigma) \in \mathrm{ATT}$ if $\exists (u, \{m'_i\}_{i=1}^n, \sigma) \in \mathrm{ATT}$ with $m'_i = m_i$ for all $i \in \mathcal{D}$.

- $\mathcal{O}\mathrm{HU}(u)$: on input a user identity u, this oracle returns $\perp$ if $u \in \mathrm{HU} \cup \mathrm{CU}$. Else it generates a key pair $(\mathrm{usk}, \mathrm{upk}) \leftarrow\!\!\$\ \mathsf{KeyGen}(\mathrm{pp})$ and returns upk. The identity u is then added to HU.
- $\mathcal{O}\mathrm{CU}(u, \mathrm{upk})$: on input a user identity u and optionally a public key upk this oracle registers a new corrupted user with public key upk if $u \notin \mathrm{HU}$ and returns usk = usk and all associated credentials items $(u, \{m_i^k\}_{i=1}^n, \sigma_k)$ of $\mathrm{ATT}[u]$ otherwise. In the latter case, u is removed from HU. In all cases, u is added to CU.
- $\mathcal{O}\mathsf{ObtIssue}(u, \{m_i\}_{i=1}^n)$: on input an identity $u \in \mathrm{HU}$ and a set of attributes $\{m_i\}_{i=1}^n$, this oracle runs $\sigma \leftarrow\!\!\$\ (\mathsf{Obtain}(\mathrm{usk}, \mathrm{ipk}, \{m_i\}_{i=1}^n), \mathsf{Issue}(\mathrm{upk}, \mathrm{isk}, \{m_i\}_{i=1}^n))$. If $u \notin \mathrm{HU}$, the oracle returns $\perp$. Else it appends $(u, \{m_i\}_{i=1}^n, \sigma)$ to ATT.

- $\mathcal{O}\mathsf{Obtain}(u, \{m_i\}_{i=1}^n)$: on input an identity $u \in \mathrm{HU}$ and a set of attributes $\{m_i\}_{i=1}^n$, this oracle runs
  $\sigma \leftarrow\!\$\ (\mathsf{Obtain}(\mathrm{usk}, \mathrm{ipk}, \{m_i\}_{i=1}^n, \mathcal{A}^{\mathcal{O}\mathsf{Issue}(.)})$. If $u \notin \mathrm{HU}$, the oracle returns $\perp$. Else it appends $(u, \{m_i\}_{i=1}^n, \sigma)$ to ATT.This oracle is used by an adversary impersonating the issuer to issue a credential to an honest user.
- $\mathcal{O}\mathsf{Issue}(u, \{m_i\}_{i=1}^n)$: on input an identity $u \in \mathrm{CU}$ and a set of attributes $\{m_i\}_{i=1}^n$, this oracle runs
  $\sigma \leftarrow\!\$\ (\mathcal{A}^{\mathcal{O}\mathsf{Obtain}(.)}, \mathsf{Issue}(\mathrm{upk}, \mathrm{isk}, \{m_i\}_{i=1}^n))$. If $u \notin \mathrm{CU}$, the oracle returns $\perp$. Else it appends $(u, \{m_i\}_{i=1}^n, \sigma)$ to ATT. This oracle is used by an adversary playing the role of a malicious user to obtain a credential from an honest issuer.
- $\mathcal{O}\mathsf{Show}(k, \mathcal{D}, \mathrm{ctx})$: On input an issuance query index k and and attributes-subset $\mathcal{D}$, and a context ctx. This oracle check if $0 < k \leq |\mathrm{ATT}|$, if the check passes, it parse $\mathrm{ATT}[k]$ as $(u, \{m_i^k\}_{i=1}^n, \sigma^k)$. If $u \notin \mathrm{HU}$, this oracle returns $\perp$. Else, this oracle runs $vp \leftarrow\!\$\ \mathsf{Show}(\mathrm{ipk}, \mathrm{usk}, \{m_i^k\}_{i=1}^n, \mathcal{D}, \sigma^k, \mathrm{ctx})$.
- $\mathcal{O}\mathrm{Ano}^b(k_0, k_1, \mathcal{D})$: On input two issuance query index k_0 and k_1, and and attributes-subset $\mathcal{D}$, this oracle first check if $0 < k_0, k_1 \leq |\mathrm{ATT}|$, if the check passes it parses $\mathrm{ATT}[k_0]$ as $(u_0, \{m_i^{k_0}\}_{i=1}^n, \sigma^{k_0})$, and $\mathrm{ATT}[k_1]$ as $(u_1, \{m_i^{k_1}\}_{i=1}^n, \sigma^{k_1})$. If not it return $\perp$. If it exist $i \in \mathcal{D}$ such that $m_i^{k_0} \neq m_i^{k_1}$, or $u_0, u_1 \notin \mathrm{HU}$, then return $\perp$. Else, this oracle runs $b^* \leftarrow\!\$\ (\mathsf{Show}(\mathrm{ipk}, \mathrm{USK}[u_b], \{m_i^{k_b}\}_{i=1}^n, \mathcal{D}, \sigma^{k_b}, \mathrm{ctx}), \mathcal{A})$, with the adversary playing a malicious verifier.
  This oracle is used in the the anonymity game as a challenge oracle for the adversary, who must distinguish (multiple) showings of two credentials σ^{k_0} and σ^{k_1}.

Anonymity. In essence, a verifiable presentation (VP) should reveal only the attributes that the user chooses to disclose, without leaking information about undisclosed attributes. Furthermore, a VP generated from a credential must not be linkable to the issuance of that credential, and multiple VPs derived from the same credential, regardless of whether they disclose the same attributes or different ones, must remain unlinkable, except for any linkability inherently implied by the disclosed attributes themselves. The first aspect follows directly from the definition below. The latter two aspects are ensured by the fact that in this definition $\mathcal{A}$ knows the credentials σ^{k_0} and σ^{k_1}, and can therefore simulate proofs using these credentials independently.

Definition 5. *(Anonymity).* $\mathcal{A}$ succeeds in the $\mathbf{Exp}_{\mathcal{A}}^{\mathrm{ano}}(1^\lambda)$, when $\mathbf{Exp}_{\mathcal{A}}^{\mathrm{ano}}(1^\lambda)$ output 1. The advantage of $\mathcal{A}$ is then defined as:

$$\mathsf{Adv}_{\mathcal{A},\Pi}^{\mathrm{ano}}(1^\lambda) = |\Pr\left[\mathbf{Exp}_{\mathcal{A}}^{\mathrm{ano}}(1^\lambda) = 1\right] - \tfrac{1}{2}|.$$

Anonymity requires that the advantage of any efficient adversary $\mathcal{A}$ remains negligible.

Unforgeability
$\underline{\mathbf{Exp}^{\mathrm{uf}}_{\mathcal{A}}(1^\lambda, n)}$

1. $\mathrm{pp} \leftarrow\!\$\ \mathsf{Setup}(1^\lambda, n)$;
2. $(\mathrm{isk}, \mathrm{ipk}) \leftarrow\!\$\ \mathsf{IKeyGen}(\mathrm{pp})$;
3. $(\mathcal{D}, \{m_i\}_{i\in\mathcal{D}}, vp, \mathrm{ctx}) \leftarrow\!\$\ \mathcal{A}^{\mathcal{O}\mathrm{HU},\mathcal{O}\mathrm{CU},\mathcal{O}\mathsf{ObtIssue},\mathcal{O}\mathsf{Issue},\mathcal{O}\mathsf{Show}}(\mathrm{pp}, \mathrm{ipk})$;
4. If $\{u, \{m_i\}_{i\in\mathcal{D}}\} \subset \mathrm{ATT}$ with $u \in \mathrm{CU}$ or if $\mathsf{Verify}(\mathrm{ipk}, vp, \mathcal{D}, \{m_i\}_{i\in\mathcal{D}}, \mathrm{ctx}) = 0$, return 0;
5. Return 1;

Anonymity
$\underline{\mathbf{Exp}^{\mathrm{ano}}_{\mathcal{A}}(1^\lambda, n)}$

1. $\mathrm{pp} \leftarrow\!\$\ \mathsf{Setup}(1^\lambda, n)$;
2. $(\mathrm{isk}, \mathrm{ipk}) \leftarrow\!\$\ \mathsf{IKeyGen}(\mathrm{pp})$;
3. $b \leftarrow\!\$\ \{0, 1\}$;
4. $(k_0, k_1, \mathcal{D}) \leftarrow\!\$\ \mathcal{A}^{\mathcal{O}\mathrm{HU},\mathcal{O}\mathrm{CU},\mathcal{O}\mathsf{Obtain},\mathcal{O}\mathsf{Show}}(\mathrm{pp}, \mathrm{ipk})$;
5. $b^* \leftarrow\!\$\ \mathcal{A}^{\mathcal{O}\mathrm{Ano}^b}$;
6. Return $(b^* = b)$;

Fig. 2: Security requirements for anonymous credentials

Unforgeability. Unforgeability ensures that it is infeasible for an adversary to produce a valid verifiable presentation (VP) without having previously obtained a credential from an accepted issuer on attributes satisfying the disclosure policy $\mathcal{D}$. In other words, a user cannot generate a valid VP for a given set of disclosed attributes unless it possesses a credential whose attribute vector matches those attributes exactly. This property should also prevent 'mix-and-match' attacks, where attributes from multiple credentials could be combined to construct a fraudulent proof.

Definition 6. *(Unforgeability).* $\mathcal{A}$ succeeds in the $\mathbf{Exp}^{\mathrm{uf}}_{\mathcal{A}}(1^\lambda)$, when $\mathbf{Exp}^{\mathrm{uf}}_{\mathcal{A}}(1^\lambda)$ output 1. The advantage of $\mathcal{A}$ is then defined as:

$$\mathsf{Adv}^{\mathrm{uf}}_{\mathcal{A},\Pi}(1^\lambda) = \Pr\Big[\mathbf{Exp}^{\mathrm{uf}}_{\mathcal{A}}(1^\lambda) = 1\Big].$$

Unforgeability requires that the advantage of any efficient adversary $\mathcal{A}$ remains negligible.

B Attacks on some issuer-hiding ACs

B.1 Weaknesses of Bosk et al. Issuer Hiding AC Scheme [11]

Below we provide a short description of the scheme of Bosk et al. of (PETS 2022). Let $(p, G_1, G_2, G_T, g_1, g_2)$ be a bilinear environment. We denote by pol the set of

issuers trusted by the verifier and by l the cardinality of pol. Let $(x_i, y_i) \in \mathbb{Z}_p^2$ be the private key of the issuer I_i for $i \in [1, l]$, and let $X_i = g_2^{x_i}$, $\overline{X}_i = g_1^{1/x_i}$, $Y_i = g_2^{y_i}$, $\overline{Y}_i = g_1^{1/y_i}$ be its corresponding public key.

Generation of the policy The verifier V proceeds as follows to define its key pair, that is, a secret key sk_{pol} and a public key pk_{pol}, for its policy *pol*:

- Choose at random $(x, y) \in \mathbb{Z}_p^2$;
- For $i \in [1, l]$, compute $W_i = \overline{X}_i^x = g_1^{x/x_i}$;
- For $i \in [1, l]$, compute $W_i^{'} = \overline{Y}_i^y = g_1^{y/y_i}$;
- Set $sk_{pol} = (x, y)$ and publish or send $pk_{pol} = \{\overline{X}_i, \overline{Y}_i, W_i, W_i^{'}\}_{i=1}^{l}$.

Credential issuance The PS signature scheme [44] is used by Issuers to issue credentials to users. To issue a credential σ to user U on their attributes *att*, the issuer I_i proceeds as follows:

- Choose at random two random generators h_1 and h_2 of G_1;
- Compute $\sigma_1 = h_1^{x_i + y_i R \cdot H(att)}$ and $\sigma_2 = h_2^{x_i + y_i R \cdot H(H(att))}$, where R is a random value chosen by U and H is a secure hash function;
- Set $\sigma = (\sigma_1, \sigma_2, h_1, h_2)$.

Presentation If U wants to anonymously show (i.e., without revealing the identity of the credential Issuer) their credential on their attributes *att* to V, they generate at random $r_u, r_{u,x}, r_{u,y} \in \mathbb{Z}_p^*$ and compute using pk_{pol} and their credential:

$$
\begin{aligned}
X_i^{'} &= X_i^{r_u} = g_2^{x_i \cdot r_u} \\
Z_i &= W_i^{r_{u,x}} = g_1^{x \cdot r_{u,x}/x_i} \\
h_x &= g_2^{r_{u,x} \cdot r_u} \\
Y_i{}' &= Y_i^{r_u R} = g_2^{y_i \cdot r_u R} \\
Z_i{}' &= W_i^{r_{u,y}\prime} = g_1^{y \cdot r_{u,y}/y_i} \\
h_y &= g_2^{r_{u,y} \cdot r_u \cdot R}
\end{aligned}
$$

A randomized version $\sigma^{'}$ of the credential σ is also generated using r_u, R, and new random values; this is done using the classical technique to randomize PS signatures. The presentation consists of $(X_i', Z_i, Y_i', Z_i', h_x, h_y, \sigma')$.

Verification The verifier V can verify, using sk_{pol}, the validity of this presentation by checking that:

$$
\begin{aligned}
e(Z_i, X_i') &= e(g_1^x, h_x) \\
e(Z_i', Y_i') &= e(g_1^y, h_y)
\end{aligned}
$$

The verification of the randomized credential σ' is done as usual using the verification algorithm of PS signatures. The above pairing equations are used by V to check whether the user makes use of a valid issuer's public key (i.e., a public key that belongs to an issuer in pol). In their paper, Bosk et al. consider the possibility that an issuer may act maliciously (see Footnote 2). One could argue that a malicious issuer could issue fake credentials to any user they choose (for example, they could provide proof of majority to a minor). However, they would risk this fraud being discovered if the fake credentials were ever revealed. Even worse, an unscrupulous user could blackmail the issuer by threatening to expose the fake credentials. Therefore, such fraud (that is issuing fake credentials) does not seem very plausible.

The attacks we describe below are completely different in that they are undetectable. This means that the identity provider responsible for them cannot be identified. More precisely, they will enable a malicious issuer and a dishonest user to deceive the verifier into accepting a credential issued with a public key that is not included in pk_{pol}.

First attack A malicious issuer I_j in pol could use its witnesses $W_j = \overline{X}_j^x = g_1^{x/x_j}$ and $W_j{}' = \overline{Y}_j^y = g_1^{y/y_j}$ as well as its private key (x_j, y_j) to compute $S = g_1^x = W_j^{x_j}$ and $S' = g_1^y = W_j^{y_j\prime}$, then disclose these two values S and S'. Using these values, anyone could compute false valid witnesses, $W = S^{1/x^*} = g_1^{x/x^*}$ and $W' = S'^{1/y^*} = g_1^{y/y^*}$, for a private key (x^*, y^*) of their choice. With (x^*, y^*), they could then forge credentials on attributes of their choice that would be accepted by the verifier V whose secret key is $sk_{pol} = (x, y)$. The pair of keys (x^*, y^*) does not correspond to the private key of any legitimate issuer. It would also be impossible for the verifier to identify the issuer responsible for the fraud (i.e., the one who disclosed S and S'). To thwart this attack, Bosk et al. suggest that the verifier choose a new secret key sk_{pol} for each verifiable presentation with a user. However, this would not be practical, as the verifier would have to manage many secrets and generate a new public key pk_{pol} (denoted Agg $= (S, W)$ in their paper) for each presentation.

In section 6.2.1, Bosk et al. propose an alternative solution: "Our interactive aggregator version modifies the Gen and WitCreate algorithms in Protocol 1. It modifies the witnesses making sure that no malicious issuer can output the verifier's private key by computing $W_l^{x_l} = g_1^{x_l}$" (that is, $S = g_1^x = W_j^{x_j}$ and $S' = g_1^y = W_j^{y_j}$ with our notations). They therefore propose to compute the witnesses W_j and W_j' as follows:

$$W_j = g_1^{x(1+\frac{1}{x_j})} \quad \text{and} \quad W_j^{'} = g_1^{y(1+\frac{1}{y_j})}$$

Nevertheless, it is still relatively straightforward for a dishonest issuer to retrieve $S = g_1^x = W_j^{\frac{x_j}{x_j+1}}$ and $S' = g_1^y = W_j^{\frac{y_j}{y_j+1}}$, even with this modified version of the witnesses. Once again, if they want to protect themselves from this attack,

verifiers will not be exempt from choosing a new secret key sk_{pol} for each verifiable presentation with a user.

Second attack If the verifier used a new secret key $sk_{pol} = (x_{new}, y_{new})$ for each presentation, the previous attack would be difficult to implement. Indeed, dishonest users would need to interact systematically with the malicious issuer to obtain the values $S = g_1^{x_{new}}$ and $S' = g_1^{y_{new}}$ corresponding to the new secret key sk_{pol}. Therefore, we propose an alternative attack in which the issuer publishes a set of values that dishonest users can later use to both reconstruct S and S' and generate fake credentials for arbitrary attributes. This can be done without users having to interact with the malicious issuer (and without the latter having to reveal its private key). The attack works as follows, with the malicious issuer I_i with private keys $(x_i, y_i) \in \mathbb{Z}_p^2$:

- chooses at random $x^*, y^*, R_1^*, R_2^*, R_3^*, R_4^* \in \mathbb{Z}_p^*$;
- computes

$$\tilde{x} = \frac{x_i \cdot R_2^*}{x^*}, \quad \tilde{x}^{'} = R_1^* x^*$$
$$h_1^{'} = g_2^{R_1^* \cdot R_2^*}$$
$$\tilde{y} = \frac{y_i \cdot R_4^*}{y^*}, \quad \tilde{y}^{'} = R_3^* y^*$$
$$h_2^{'} = g_2^{R_3^* \cdot R_4^*}$$

The malicious issuer then sends

$$S_{malicious} = (\tilde{x}, \tilde{x}^{'}, h_1^{'}, \tilde{y}, \tilde{y}^{'}, h_2^{'})$$

to the dishonest user U. Using $\tilde{x}^{'}$ and $\tilde{y}^{'}$, U can sign any message m. U will then be able to forge credentials for arbitrary attributes att that will be accepted by any verifier V. If U wants to anonymously show their (fake) credential on their attributes att to V, they generate at random $r_u, r_{u,x}, r_{u,y}, R \in \mathbb{Z}_p^*$ and compute using pk_{pol} (namely the values W_i and $W_i^{'}$ corresponding to the malicious issuer I_i), the set $S_{malicious}$ and their fake credential:

$$X_i' = g_2^{\tilde{x}' \cdot r_u} = g_2^{r_u R_1^* x^*}$$
$$Z_i = W_i^{r_{u,x}\tilde{x}} = g_1^{\frac{x \cdot r_{u,x} R_2^*}{x^*}}$$
$$h_x^* = h_1 \cdot r_u r_{u,x} = g_2^{R_1^* R_2^* r_{u,x} r_u}$$
$$Y_i' = g_2^{\tilde{y}' \cdot r_u R} = g_2^{r_u R_3^* y^* R}$$
$$Z_i' = W_i^{r_{u,y}\tilde{y}'} = g_1^{\frac{y \cdot r_{u,y} R_4^*}{y^*}}$$
$$h_y^* = h_2' \cdot r_u r_{u,y} R = g_2^{R_3^* R_4^* r_{u,y} r_u R}$$

and sends $(X'_i, Z_i, Y'_i, Z'_i, h^*_x, h^*_y)$ along with the fake credential to the verifier. Again, the attack works, as:

$$e(Z_i, X'_i) = e(g_1^x, h^*_x)$$
$$e(Z'_i, Y_i^{'}) = e(g_1^y, h^*_y)$$

B.2 Weaknesses of IHBBS1 from (eprint report 2026/369) [31]

In [31], Flamini et al. show how to add issuer-hiding to BBS-based credentials. They use a technique similar to that employed by Bosk et al. [11] to hide the issuer of a credential among a set of issuers. This technique is referred to as an "aggregator" in [11] and a "tag" in [31]. A tag scheme can be seen as a relaxed variant of a signature scheme.

Below we provide a short description of IHBBS1.

Let $(p, G_1, G_2, G_T, g_1, g_2)$ be a bilinear environment. We denote by pol the set of issuers trusted by the verifier and by l the cardinality of pol.

Let $x_i \in \mathbb{Z}_p^*$ be the private key of the issuer I_i for $i \in [1, l]$, and let $X_i = g_2^{x_i}$ be its corresponding BBS public key.

Generation of the policy A trusted authority A proceeds as follows to define its key pair, that is a secret key sk_{pol} and a public key pk_{pol}, for the policy pol:

- Choose at random $x \in \mathbb{Z}_p^*$ and a random generator G in G_1;
- Compute $T = G^{1/x}$;
- For $i \in [1, l]$, compute $W_i = X_i^{1/x} = g_2^{x_i/x}$;
- Set $sk_{pol} = x$ and publish $pk_{pol} = (G, T, \{W_i\}_{i=1}^l)$.

Credential issuance The BBS signature scheme [47] is used by issuers to issue credentials to users. Let $(h_1, h_2, \ldots, h_n)$ be n public generators of G_1. To issue a credential σ to user U on their attributes $att = (a_1, a_2, \ldots, a_n)$, the issuer I_i proceeds as follows:

- Choose at random $e \in \mathbb{Z}_p^*$;
- Compute $C = g_1 \prod_{j=1}^n h_j^{a_j}$ and $A = C^{1/(x_i+e)}$;
- Set $\sigma = (A, e)$.

Presentation To anonymously show (i.e., without revealing the identity of the credential issuer) their credential on their attributes att to a verifier V, U proceeds as follows:

- Choose at random $\alpha \in \mathbb{Z}_p^*$;
- Randomize the issuer's public key X_i and its corresponding tag W_i using α: $X^* = X_i^\alpha$ and $T^* = W_i^\alpha$;
- Randomize the credential $\sigma = (A, e)$ using α: $\sigma^* = (A^{1/\alpha}, \alpha e)$;
- Generate, using classical BBS techniques, a zero-knowledge proof of knowledge π proving the knowledge of a credential σ^* on att that is valid under the public key X^*.

The presentation consists of (X^*, T^*, π).

Verification The verifier V can verify, using pk_{pol}, the validity of this presentation by checking that:

$$e(T, X^*) = e(G, T^*)$$

The generation and verification of the proof π are done as usual using the showing and verification algorithms of BBS. The above pairing equation is used by V to check whether the user makes use of a valid issuer's public key (i.e., a public key that belongs to an issuer in pol). The attack described below is similar to that against Bosk et al.'s scheme [11]. This is understandable, given that both schemes employ the same technique to hide the issuer's public key (an aggregator or tag). As with Bosk et al. [11], we again consider the possibility that an issuer may act maliciously (see footnote 2 in [11]). One could argue that a malicious issuer could issue fake credentials to any user they choose in any case (for example, they could provide proof of majority to a minor). However, they would risk this fraud being discovered if the fake credentials were ever revealed. Even worse, an unscrupulous user could blackmail the issuer by threatening to expose the fake credentials. Therefore, such fraud (that is issuing fake credentials) does not seem very plausible.
The attack we describe below is completely different in that it is undetectable. This means that the malicious issuer cannot be identified. More precisely, our attack will enable a malicious issuer and a dishonest user to deceive the verifier into accepting a (fake) credential issued with a public key that is not included in pk_{pol}.

The attack A malicious issuer I_i in pol could use its witnesses $W_i = X_i^{1/x} = g_2^{x_i/x}$ as well as its private key x_i to compute $\tilde{W} = g_2^{1/x} = W_i^{1/x_i}$ and then disclose $\tilde{W}$. Using $\tilde{W}$, anyone could compute false valid witnesses, $T^* = \tilde{W}^{x^*} = g_2^{x^*/x}$, for a private key x^* of their choice. With x^*, they could then forge credentials on attributes of their choice that would be accepted by the verifier V. Indeed, the fake public key $X^* = g_2^{x^*}$ will be accepted as valid by any verifier (i.e., belonging to an issuer in the policy *pol*) as:

$$e(T, X^*) = e(G, T^*)$$

The private key x^* does not correspond to the private key of any legitimate issuer. It would also be impossible for the verifier to identify the issuer responsible for this fraud (i.e., the one who disclosed $\tilde{W}$). It does not seem easy to counteract the above attack unless the scheme is modified to use a secure digital signature scheme rather than a tag-based authentication scheme to authenticate the public keys of the issuers in *pol*. This is the option we chose for both of our issuer-hiding constructions: we use BBS to authenticate the issuers' public keys. Therefore, they are not susceptible to an attack involving a malicious issuer.

C Proofs

C.1 Proof of Theorem 1

Proof of Unforgeability.

Proof. (sketch)

Let $\mathcal{A}$ be a P.P.T adversary against the unforgeability of our protocol, and let $\mathrm{EXT} = \{\mathrm{EXT}_{\mathrm{setup}}, \mathrm{EXT}_{\mathrm{user}}, \mathrm{EXT}_{\mathrm{cert}}, \mathrm{EXT}_{\mathrm{validity}}\}$. We define the extractors for the credential system as follows.

- $\mathrm{EXT}_{\mathrm{setup}}$ receives as input $(\mathrm{ipk}_j, \tilde{ipk}_j, \pi_{\mathrm{setup}})$ and recovers a witness isk_j for the instance $(\mathrm{ipk}_j, \tilde{ipk}_j)$ such that $\mathrm{ipk}_j = g^{\mathrm{isk}_j} \wedge \tilde{\mathrm{ipk}}_j = \tilde{g}^{\mathrm{isk}_j}$.
  If the extraction fails, meaning if π_{setup} passes the verification but the extractor fails to output a valid witness (isk_j, such that $\mathrm{ipk}_j = g^{\mathrm{isk}_j} \wedge \tilde{\mathrm{ipk}}_j = \tilde{g}^{\mathrm{isk}_j}$), then the extractor aborts. Let $\mathrm{BadExtract}_{\mathrm{setup}}$ be the event that $\mathcal{A}$ wins but the extractor fails to output a valid witness.
- $\mathrm{EXT}_{\mathrm{user}}$ receives as input $(\mathrm{upk}, \pi_{\mathrm{user}})$ and the user attribute $\{m_i\}_1^n$ recovers a witness usk for the instance upk such that $\mathrm{upk} = g^{\mathrm{usk}}$.
  If the extraction fails, then the extractor aborts. Let
  $\mathrm{BadExtract}_{\mathrm{issue}}$ be the event that $\mathcal{A}$ wins but the extractor fails to output a valid witness
- $\mathrm{EXT}_{\mathrm{cert}}$ receives as input $(\tilde{\mathrm{vpk}}, \mathrm{ipk}'_j, \bar{B}_j, \bar{A}_j, \pi_{\mathrm{cert}})$ and recovers a witness (t', r, e_j) for the instance $(\tilde{\mathrm{vpk}}, \mathrm{ipk}'_j, \bar{B}_j, \bar{A}_j)$ such that $\mathbf{e}(\bar{B}_j, \tilde{h}) = \mathbf{e}(\bar{A}_j, \tilde{\mathrm{vpk}})$, and $\bar{B}_j$ was well formed given the provided instance. From the recovered witness the adversary can recover ipk_j.
  If the extraction fails, then the extractor abort. Let
  $\mathrm{BadExtract}_{\mathrm{cert}}$ be the event that $\mathcal{A}$ wins but the extractor fails to output a valid witness.
- $\mathrm{EXT}_{\mathrm{validity}}$ receives as input $(\mathcal{D}, \mathrm{ipk}_j, \tilde{\mathrm{ipk}}'_j, \bar{B}, \bar{A}, \bar{D}, \pi_{\mathrm{validity}})$ and recovers a witness $(e, \hat{t}, r_3, \{m_i\}_i^n)$ for the instance
  $(\mathrm{ipk}_j, \mathrm{ipk}'_j, \bar{B}_j, \bar{A}_j, \mathcal{D})$ such that the disclosed message satisfies some predicate, $\mathbf{e}(\bar{B}, \tilde{g}) = \mathbf{e}(\bar{A}, \tilde{\mathrm{ipk}}_j)$, and $\bar{B}$ was well formed given the provided instance. From the recovered witness the adversary can recover ipk_j.
  If the extraction fails, then the extractor aborts. Let
  $\mathrm{BadExtract}_{\mathrm{validity}}$ be the event that $\mathcal{A}$ wins but the extractor fails to output a valid witness.

$\mathcal{A}$ wins if one of the following happens:

(1) The adversary $\mathcal{A}$ returns $(\mathrm{ipk}_j, \tilde{\mathrm{ipk}}_j, \pi_{\mathrm{setup}})$ such that π_{setup} verifies for the statement $(\mathrm{ipk}_j, \tilde{\mathrm{ipk}}_j)$ but extraction failed, which correspond to the event $\mathrm{BadExtract}_{\mathrm{setup}}$.
(2) $\mathcal{A}$ returns $\mu = (\mathrm{upk}, \pi_{\mathrm{user}})$ such that π_{user} verifies for upk but extraction failed, which corresponds to the event
$\mathrm{BadExtract}_{\mathrm{issue}}$.

(3) $\mathcal{A}$ returns $\rho_1^* = (\text{ipk}_{j^*}{}', \bar{B}_{j^*}, \bar{A}_{j^*}, \pi^*_{\text{cert}})$ such that π_{cert} verifies for the statement $(\text{ipk}_{j^*}{}', \bar{B}_{j^*}, \bar{A}_{j^*})$ but extraction failed, which corresponds to the event $\text{BadExtract}_{\text{cert}}$.
(4) $\mathcal{A}$ returns $(\mathcal{D}, \rho_2^* = (\bar{B}^*, \bar{A}^*, \bar{D}^*, \pi^*_{\text{validity}}))$ such that π^*_{validity} verifies for the statement $(\mathcal{D}, \bar{B}^*, \bar{A}^*, \bar{D}^*)$ but extraction failed, which correspond to the event $\text{BadExtract}_{\text{validity}}$.
(5) The presentation returned $\rho_1^* = (\text{ipk}_{j^*}{}', \bar{B}_{j^*}, \bar{A}_{j^*}, \pi^*_{\text{cert}})$ by the adversary $\mathcal{A}$ satisfies the verification, and we have $\text{ipk}_{j^*} \notin \mathcal{P}^*$ (ipk_{j^*}, is the de-randomized version of $\text{ipk}_{j^*}{}'$). Lets denote this event CertForge.
(6) The presentation returned $\rho_2^* = (\bar{B}^*, \bar{A}^*, \bar{D}^*, \pi^*_{\text{validity}})$ by $\mathcal{A}$ satisfies the verification, and we have $\boldsymbol{m}^* \notin \text{ATT}$. Lets denote this event CredForge.

Therefore we can quantify the advantage of $\mathcal{A}$, by the following:

$$\mathsf{Adv}^{\text{uf}}_{\mathcal{A}}(1^\lambda) \leq \Pr[\text{CertForge}] + \Pr[\text{CredForge}] + \Pr[\text{BadExtract}_{\text{setup}}] + \Pr[\text{BadExtract}_{\text{issue}}] + \Pr[\text{BadExtract}_{\text{cert}}] + \Pr[\text{BadExtract}_{\text{validity}}].$$

Items (1), (2), (3), and (4) can be reduced to knowledge soundness (ksnd) of the corresponding PoKs. Therefore $\Pr[\text{BadExtract}_{\text{setup}}]$, $\Pr[\text{BadExtract}_{\text{issue}}], \Pr[\text{BadExtract}_{\text{cert}}]$, and $\Pr[\text{BadExtract}_{\text{validity}}]$ correspond respectively to the quantities $\mathsf{Adv}^{\text{ksnd}}_{\text{EXT},\pi_{\text{setup}}}(1^\lambda)$, $\mathsf{Adv}^{\text{ksnd}}_{\text{EXT},\pi_{\text{user}}}(1^\lambda), \mathsf{Adv}^{\text{ksnd}}_{\text{EXT},\pi_{\text{cert}}}(1^\lambda)$, and $\mathsf{Adv}^{\text{ksnd}}_{\text{EXT},\pi_{\text{validity}}}(1^\lambda)$.
Now it remain to study the case of the item (5) and (6), which reduces both to the unforgeability of BBS. To do so we build a reduction $\mathcal{B}$ against the SUF-CMA security of BBS, which uses $\mathcal{A}$ as a black box. We consider two types of adversaries Type-1 which corresponds to the item (5) meaning the event CertForge, and a Type-2 adversaries which corresponds to the item (6) meaning the event CredForge.

– Type-1 adversaries: Our adversary $\mathcal{A}$ is an algebraic adversary, which means that when $\mathcal{A}$ returns a group element, it return also a description of the later with respect to all the group element he has previously seen. Our reduction $\mathcal{B}$ simulates the unforgeability experiment of our protocol for $\mathcal{A}$, by interacting with the BBS challenger $\mathcal{C}$ who gives him access to the oracle $\mathcal{O}\mathsf{Sign}$ [7].
  - **$\mathcal{B}$ simulation to answer $\mathcal{A}$ queries**:
    * $\mathcal{B}$ generates the public parameters $\text{pp} = (p, \mathbb{G}_1, \mathbb{G}_2, g, \tilde{g}, \mathbf{e}, g_0, h, h_0, h_1, \ldots h_n, \tilde{h})$, by first running the bilinear group generators $(p, \mathbb{G}_1, \mathbb{G}_2, g, \tilde{g}, \mathbf{e}) \leftarrow\!\$\ \mathsf{BGen}(1^\lambda)$ and selecting $\lambda_i \leftarrow\!\$\ \mathbb{Z}_p^*$ for all $i = 0, \ldots, n$, $\nu, \lambda, \tilde{\lambda} \leftarrow\!\$$, and computing $h_i = g^{\lambda_i}$, $g_0 = g^\nu$, $h = g^\lambda$, $\tilde{h} = \tilde{g}^{\tilde{\lambda}}$. He sends pp to the his challenger $\mathcal{C}$, which uses it to generate the verification keys. After that $\mathcal{C}$ sends the verifier's public key vpk.
    * $\mathcal{B}$ generates $(\text{isk}_j, \text{ipk}_j, \tilde{\text{ipk}}_j) \leftarrow\!\$\ \mathsf{IKeyGen}(\text{pp})$ for $j = 1, \ldots n_J$, set $\mathcal{P} = \{(\text{ipk}_j, \tilde{\text{ipk}}_j)\}_i^{n_J}$ and runs

[7] The oracle $\mathcal{O}\mathsf{Sign}$ here, is a BBS signing oracle.

$(\mathcal{P}') \leftarrow\!\!\$\ \mathcal{A}^{\mathcal{O}\mathrm{HI},\mathcal{O}\mathrm{CI}}(\mathrm{pp},\mathcal{P})$. For every public key $(\mathrm{ipk}_j, \tilde{\mathrm{ipk}}_j) \in \mathcal{P}'$, the adversary $\mathcal{A}$ must provide a proof of knowledge π_{setup} of the corresponding secret keys. $\mathcal{B}$ checks all these provided PoK, and if one of them fails $\mathcal{B}$ aborts, else $\mathcal{B}$ set $\mathcal{P}^* = \mathcal{P} \cup \mathcal{P}'$.

* For every $\mathrm{ipk}_j \in \mathcal{P}^*$ if $\mathrm{ipk}_j \in \mathcal{P}$, $\mathcal{B}$ queries $\mathcal{O}\mathsf{Sign}$ on isk_j, which returns a BBS signature on sk_j, $\sigma_j = (A_j = (g_0 g^{\mathrm{isk}_j})^{\frac{1}{x+e_j}}, e_j)$. If $\mathrm{ipk}_j \notin \mathcal{P}$, $\mathcal{B}$ extracts the isk_j form the provided PoK π_{setup}, and queries $\mathcal{O}\mathsf{Sign}$ on isk_j. $\mathcal{B}$ sets $\mathrm{pol}^* = (\mathrm{vpk}, \{(\mathrm{ipk}_j, \tilde{\mathrm{ipk}}_j), \sigma_j\}_j)$.
* The adversary $\mathcal{A}$ is allowed to submit credential queries from the issuer of his choice in $\mathcal{P}^*$, and we note $qS = |\mathrm{ATT}|$ the number of this queries. When $\mathcal{A}$ submit an issue query with $(\boldsymbol{m}, \mathrm{upk}, \pi_{\mathrm{user}})$, if π_{user} does not verify, $\mathcal{B}$ aborts. Else $\mathcal{B}$ extracts the witness usk from π_{user} answers this query with a BBS signature $\sigma = (A = (h_0 h^{\mathrm{usk}} \prod_{i=1}^{n} h_i^{m_i})^{\frac{1}{x+e}}, e)$.

- **Output**: We consider that our reduction $\mathcal{B}$ perfectly simulates the unforgeability experiment for $\mathcal{A}$.

  * Finally $\mathcal{B}$ runs $(vp^*, \mathcal{D}^*, \mathrm{ctx}^*) \leftarrow\!\!\$\ \mathcal{A}^{\mathcal{O}\mathsf{Issue},\mathcal{O}\mathsf{Show}}(st, \mathrm{pol}^*)$.
    Where $vp^* = (\{m_i\}_{i\in\mathcal{D}}, \mathrm{ipk}_{j^*}{}', \tilde{\mathrm{ipk}}'_{j^*}, \bar{B}_{j^*}, \bar{A}_{j^*}, \bar{B}, \bar{A}, \bar{D}, \pi_{\mathrm{cert}}, \pi_{\mathrm{validity}})$, and for every group element in vp^*, $\mathcal{A}$ has accompanied it with its algebraic representation, which are $\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}, \boldsymbol{\delta}, \boldsymbol{\varphi}, \boldsymbol{\omega}, \boldsymbol{\eta}$, for respectively $\bar{B}_{j^*}, \bar{A}_{j^*}, \bar{B}, \bar{A}, \bar{D}, \mathrm{ipk}_{j^*}{}', \tilde{\mathrm{ipk}}'_{j^*}$.
  * If $\mathcal{A}$ winning condition is not satisfied, then $\mathcal{B}$ aborts.
  * Otherwise $\mathcal{B}$ extracts a witness (t', r, e_j) from π_{cert} and a witness $(e, \hat{t}, r_3, \{m_i^*\}_i^n)$ from π_{validity}.
  * With the extracted values $\mathcal{B}$ recovers ipk_{j^*}, and the corresponding certificate $\sigma_{j^*} = (A_{j^*}, e_{j^*})$. If $\mathrm{ipk}_{j^*} \in \mathcal{P}^*$, then $\mathcal{B}$ aborts.
  * Otherwise $\mathcal{B}$ tries to recover $\mathrm{isk}_{j^*} = y_{j^*} = log_g(\mathrm{ipk}_{j^*})$, so that (y_{j^*}, σ_{j^*}) will be a BBS forgery. To do so $\mathcal{B}$ uses the algebraic representation of the different group element, provided by the adversary $\mathcal{A}$. We have the following:

$$\bar{B}_{j^*} = g^{\alpha_g} g_0^{\alpha_{g,0}} \prod_{i\in[n]} h_i^{\alpha_{h,i}} \prod_{j\in\mathcal{P}^*} (A_j^{\alpha_{a,j}} \mathrm{ipk}_j^{\alpha_{y,j}}) \prod_{k\in[qS]} A_k^{\alpha_{a,k}}$$
$$\bar{A}_{j^*} = g^{\beta_g} g_0^{\beta_{g,0}} \prod_{i\in[n]} h_i^{\beta_{h,i}} \prod_{j\in\mathcal{P}^*} (A_j^{\beta_{a,j}} \mathrm{ipk}_j^{\beta_{y,j}}) \prod_{k\in[qS]} A_k^{\beta_{a,k}}.$$
$$\bar{B} = g^{\gamma_g} g_0^{\gamma_{g,0}} \prod_{i\in[n]} h_i^{\gamma_{h,i}} \prod_{j\in\mathcal{P}^*} (A_j^{\gamma_{a,j}} \mathrm{ipk}_j^{\gamma_{y,j}}) \prod_{k\in[qS]} A_k^{\gamma_{a,k}}.$$
$$\bar{A} = g^{\delta_g} g_0^{\delta_{g,0}} \prod_{i\in[n]} h_i^{\delta_{h,i}} \prod_{j\in\mathcal{P}^*} (A_j^{\delta_{a,j}} \mathrm{ipk}_j^{\delta_{y,j}}) \prod_{k\in[qS]} A_k^{\delta_{a,k}}.$$
$$\mathrm{ipk}_{j^*}{}' = g^{\omega_g} g_0^{\omega_{g,0}} \prod_{i\in[n]} h_i^{\omega_{h,i}} \prod_{j\in\mathcal{P}^*} (A_j^{\omega_{a,j}} \mathrm{ipk}_j^{\omega_{y,j}}) \prod_{k\in[qS]} A_k^{\omega_{a,k}}.$$
$$\tilde{\mathrm{ipk}}'_{j^*} = \tilde{g}^{\eta_g} \tilde{h}^{\eta_h} \tilde{\mathrm{vpk}}^{\eta_x} \prod_{j\in\mathcal{P}^*} \tilde{\mathrm{ipk}}_j^{\eta_{y,j}}.$$

From here we can write the discrete logarithm of the group element in vp^* to the base $g, \tilde{g}$:

$$\bar{b}_{j^*} = \alpha_g + \nu_0\alpha_{g,0} + \nu_1\alpha_{g,1} + \sum_{i=0}^{n}\lambda_i\alpha_{h,i} + \sum_{j\in\mathcal{P}^*}(\alpha_{a,j}\frac{\nu_0+y_j}{x+e_j} + \alpha_{y,j}y_j) + \sum_{k\in[qS]}\alpha_{a,k}\frac{\lambda_0+\text{usk}+\sum_{i=1}^{n}\lambda_i m_{i,k}}{y_j+e_k}.$$

$$\bar{a}_{j^*} = \beta_g + \nu_0\beta_{g,0} + \nu_1\beta_{g,1} + \sum_{i=0}^{n}\lambda_i\beta_{h,i} + \sum_{j\in\mathcal{P}^*}(\beta_{a,j}\frac{\nu_0+y_j}{x+e_j} + \beta_{y,j}y_j) + \sum_{k\in[qS]}\beta_{a,k}\frac{\lambda_0+\text{usk}+\sum_{i=1}^{n}\lambda_i m_{i,k}}{y_j+e_k}.$$

$$\bar{b}^* = \gamma_g + \nu_0\gamma_{g,0} + \nu_1\gamma_{g,1} + \sum_{i=0}^{n}\lambda_i\gamma_{h,i} + \sum_{j\in\mathcal{P}^*}(\gamma_{a,j}\frac{\nu_0+y_j}{x+e_j} + \gamma_{y,j}y_j) + \sum_{k\in[qS]}\gamma_{a,k}\frac{\lambda_0+\text{usk}+\sum_{i=1}^{n}\lambda_i m_{i,k}}{y_j+e_k}.$$

$$\bar{a}^* = \delta_g + \nu_0\delta_{g,0} + \nu_1\delta_{g,1}\sum_{i=0}^{n}\lambda_i\delta_{h,i} + \sum_{j\in\mathcal{P}^*}(\delta_{a,j}\frac{\nu_0+y_j}{x+e_j} + \delta_{y,j}y_j) + \sum_{k\in[qS]}\delta_{a,k}\frac{\lambda_0+\text{usk}+\sum_{i=1}^{n}\lambda_i m_{i,k}}{y_j+e_k}.$$

$${y_{j^*}}' = \omega_g + \nu_0\omega_{g,0} + \nu_1\omega_{g,1} + \sum_{i=0}^{n}\lambda_i\omega_{h,i} + \sum_{j\in\mathcal{P}^*}(\omega_{a,j}\frac{\nu_0+y_j}{x+e_j} + \omega_{y,j}y_j) + \sum_{k\in[qS]}\omega_{a,k}\frac{\lambda_0+\text{usk}+\sum_{i=1}^{n}\lambda_i m_{i,k}}{y_j+e_k}.$$

$$\tilde{y}'_{j^*} = \eta_g + \tilde{\lambda}\eta_h + \tilde{\lambda}\eta_x x + \sum_{j\in\mathcal{P}^*}\eta_{y,j}y_j.$$

By Plugging the forgery into the verification relations we obtain the following equations:

$${y_{j^*}}' = \tilde{y}'_{j^*}.$$
$$\bar{b}_{j^*}\tilde{\lambda} = \bar{a}_{j^*}x.$$
$$\bar{b} = \bar{a}y_{j^*}.$$

Since all the scalars involved in the system of equations are known to the reduction, except for y_{j^*}, we can associate each element with a formal polynomial representing its exponent in base g (for the elements in $\mathbb{G}_1$) and $\tilde{g}$ (for those in $\mathbb{G}_2$) where the variables are the scalars unknown to the adversary, namely y_{j^*} and x. Therefore we have $\tilde{y}'_{j^*} = P(X) = pX + c$ is a polynomial of degree 1 and the indeterminate X, where c is known to $\mathcal{B}$ (i.e. revealed by $\mathcal{A}$). On the other side $y'_{j^*} = \frac{Q(X)}{R(X)}$ is a rational polynomial in X.
The equality $\tilde{y}'_{j^*} = y'_{j^*}$ implies that $S(X) = P(X)R(X) - Q(X) = 0$. Therefore

1. If $S(X)$ is the zero polynomial, this means all its coefficients are zero. It is then easy to show since $R(X)$ is of degree q_S and $Q(X)$ is at most degree q_S, this implies $p = 0$. Therefore, $\mathcal{B}$ can recover y'_{j^*}, which would be equal to c.
2. If $S(X)$ is not the zero polynomial, then x is one of its roots. $\mathcal{B}$ can thus find it using the Berlekamp algorithm for polynomial factorization [4]. Therefore, $\mathcal{B}$ is capable of solving the q-SDH problem since it can recover x.

In both cases (1 and 2 above), $\mathcal{B}$ is able to break the unforgeability of BBS. Since $\mathcal{B}$ now knows y_{j^*}, it can produce a valid BBS forgery $(y_{j^*}, \sigma_{j^*} = (A_{j^*}, e_{j^*})$. Therefore we have:

$$\Pr[\text{CertForge}] \leq \mathsf{Adv}^{\text{suf}-\text{cma}}_{\mathcal{B},\text{BBS}}(1^\lambda)$$

– Type-2 adversaries: Here our reduction $\mathcal{B}$ runs the adversary $\mathcal{A}$ as a black box to try to produce a BBS forge. $\mathcal{A}$ here is an adversary who tries to forge a VP with a credential from an honest issuer. Our reduction $\mathcal{B}$ simulates the unforgeability experiment of our protocol for the $\mathcal{A}$, by interacting with the BBS challenger $\mathcal{C}$ who gives him access to the oracle $\mathcal{O}\mathsf{Sign}$.
  - $\mathcal{B}$ **simulation to answer** $\mathcal{A}$ **queries**:
    * $\mathcal{B}$ runs the setup of the protocol to generate the public parameters pp, and send it to his BBS challenger. Who returns $\mathcal{B}$ the verification public key vpk.
    * $\mathcal{B}$ randomly selects $j_0^* \leftarrow\!\$\ \{1, \ldots, n_J\}$.
    * $\mathcal{B}$ generates $(\mathrm{isk}_j, \mathrm{ipk}_j, \tilde{\mathrm{ipk}}_j) \leftarrow\!\$\ \mathsf{IKeyGen}(\mathrm{pp})$ for $j = 1, \ldots n_J$, sets $\mathcal{P} = \{(\mathrm{ipk}_j, \tilde{\mathrm{ipk}}_j)\}_i^{n_J}$ and runs $(\mathcal{P}') \leftarrow\!\$\ \mathcal{A}^{\mathcal{O}\mathrm{HI}, \mathcal{O}\mathrm{CI}}(\mathrm{pp}, \mathcal{P})$. $\mathcal{B}$ sets $\mathcal{P}^* = \mathcal{P} \cup \mathcal{P}'$, and $\mathrm{pol}^* = \mathsf{GenPolicy}(\mathcal{P}^*)$.
    * The adversary $\mathcal{A}$ is allowed to submit credential queries from the issuer of his choice in $\mathcal{P}^*$ except for the issuer with signing key $\mathrm{isk}_{j_0{}^*}$ (knowing that $j_0^* \leftarrow\!\$\ \{1, \ldots, n_J\}$), if $\mathcal{B}$ receives such queries it aborts. We note qS the number of this queries. When $\mathcal{A}$ submit an issue query with $(\boldsymbol{m}, \mathrm{upk}, \pi_{\mathrm{user}})$, if π_{user} does not verify, $\mathcal{B}$ aborts. Else $\mathcal{B}$ extracts the witness usk from π answer this query with a BBS signature $\sigma = (A = (h_o h^{\mathrm{usk}} \prod_{i=1}^{n} h_i^{m_i})^{\frac{1}{x+e}}, e)$.
  - **Output**: We consider that our reduction $\mathcal{B}$ perfectly simulates the unforgeability experiment for $\mathcal{A}$.
    * Finally $\mathcal{B}$ runs $(vp^*, \mathcal{D}^*, \mathrm{nonce}^*) \leftarrow\!\$\ \mathcal{A}^{\mathcal{O}\mathsf{Issue}, \mathcal{O}\mathsf{Show}}(st, \mathrm{pol}^*)$.
    Where $vp^* = (\{m_i\}_{i \in \mathcal{D}}, \mathrm{ipk}_{j^*}{}', \tilde{\mathrm{ipk}}_{j^*}', \bar{B}_{j^*}, \bar{A}_{j^*}, \bar{B}^*, \bar{A}^*, \bar{D}^*, \pi^*_{\mathrm{cert}}, \pi^*_{\mathrm{validity}})$, and for every group element in vp^*, $\mathcal{A}$ has accompanied it with its algebraic representation, which are $\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}, \boldsymbol{\delta}, \boldsymbol{\varphi}, \boldsymbol{\omega}, \boldsymbol{\eta}$, for respectively $\bar{B}_{j^*}, \bar{A}_{j^*}, \bar{B}^*, \bar{A}^*, \bar{D}^*, \mathrm{ipk}_{j^*}{}', \tilde{\mathrm{ipk}}_{j^*}'$.
    * If $\mathcal{A}$ winning condition is not satisfied, then $\mathcal{B}$ aborts.
    * Otherwise $\mathcal{B}$ extracts a witness (t', r, e_j) therefore from π_{cert} and a witness $(e, \hat{t}, r_3, \{m_i^*\}_i^n)$ from π_{validity}.
    * With the extracted values $\mathcal{B}$ recovers ipk_{j^*}, and the corresponding certificate $\sigma^* = (A^*, e^*)$. If $\mathrm{ipk}_{j^*} \neq \mathrm{ipk}_{j_0^*}$, then $\mathcal{B}$ aborts.
    * If $\{m_i^*\} \in \mathrm{ATT}$, aborts. Otherwise $\mathcal{B}$ returns $(\{m_i^*\}_i^n, \sigma^* = (A^*, e^*))$ as a valid BBS forgery. Therefore we have the following:

$$\Pr\left[\mathrm{CredForge} \wedge \mathrm{ipk}_{j^*} = \mathrm{ipk}_{j_0^*}\right] = \frac{1}{n_j} \mathsf{Adv}^{\mathrm{suf-cma}}_{\mathcal{B},\mathrm{BBS}}(1^\lambda)$$

C.2 Proof of Theorem 2

Proof of Issuer-Hiding Anonymity.

Proof. (sketch) In a series of game hops, we leverage the zero-knowledge property of our of our PoK π_{cert} and π_{validity} to swap the underline values of the presentation from the input set used in $b = 0$, to the one used in $b = 1$, showing anonymity.

Game C1. *This is exactly the anonymity game with $b = 0$, where the input set $(j_0, u_0, \{m_i^0\}_{i\in[n]}, \sigma^0, \mathcal{D}, \mathcal{P}, \mathrm{pol}^*, \mathrm{ctx})$, where σ^0 is the credential of user u_0 on his attributes $\{m_i^0\}_{i\in[n]}$, issuer by the issuer j_0 who has has his public keys $(\mathrm{ipk}^0, \tilde{\mathrm{ipk}}^0)$ in $\mathcal{P}$ therefore is among the certified issuer. Let us denote by $vp = (\{m_i\}_{i\in\mathcal{D}}, \mathrm{ipk}'_*, \tilde{\mathrm{ipk}}'_*, \bar{A}_{j^*}, \bar{B}_{j^*}, \bar{A}, \bar{B}, \bar{D}, \pi_{\mathrm{cert}}, \pi_{\mathrm{validity}})$ the output of the show algorithm on this input set.*

Game C2. *In the game we simulate the PoK π_{cert} and π_{validity}, by programming the random oracle. This change is indistinguishable by the zero-knowledge property of the proofs π_{cert} and π_{validity}.*

Game C3. *In the game we switch the "content" of the verifiable presentation vp, and construct $(\{m_i\}_{i\in\mathcal{D}}, \mathrm{ipk}^{*\prime}, \tilde{\mathrm{ipk}}^{*\prime}, \bar{A}_{j^*}, \bar{B}_{j^*}, \bar{A}, \bar{B}, \bar{D}, \pi_{\mathrm{cert}}, \pi_{\mathrm{validity}})$ from $\{m_i^1\}_{i\in[n]}, \sigma^1$, the public keys $\mathrm{ipk}^1, \tilde{\mathrm{ipk}}^1$ of the issuer j_1, and his certificate σ_j^1. Since $\mathrm{ipk}^{*\prime}, \tilde{\mathrm{ipk}}^{*\prime}, \bar{A}_{j^*}, \bar{B}_{j^*}, \bar{A}, \bar{B}, \bar{D}$, are perfectly blind versions of the issuer keys, his certificate σ_j^1 and the user's credential σ_1; and $\bar{B}_{j^*}, \bar{B}, \bar{D}$ are determined unambiguously by the former elements, the distribution of the presentation proof does not change.*

Game C4. *This game revert from simulated to regular response, now using the input set $(j_1, u_1, \{m_i^1\}_{i\in[n]}, \sigma^1, \mathcal{D}, \mathcal{P}, \mathrm{pol}^*, \mathrm{ctx})$ for the witness. This makes* **??** *C3 identical to the anonymity game with $b = 1$.*

In conclusion, we showed to the adversary that the behavior of the anonymity game with $b = 0$ and $b = 1$ are indistinguishable. Therefore our construction satisfies the issuer-hiding anonymity requirement.