Daniel Bosk, Davide Frey, Mathieu Gestin, and Guillaume Piolle

Hidden Issuer Anonymous Credential

Abstract: Identity Management Systems (IMS) allow users to prove characteristics about themselves to multiple service providers. IMS evolved from impractical, site-by-site authentication, to versatile, privacy-enhancing Self Sovereign Identity (SSI) Frameworks. SSI frameworks often use Anonymous Credential schemes to provide user privacy, and more precisely unlinkability between uses of these credentials. However, these schemes imply the disclosure of the identity of the Issuer of a given credential to any service provider. This can lead to information leaks. We deal with this problem by introducing a new Anonymous Credential scheme that allows a user to hide the Issuer of a credential, while being able to convince the service providers that they can trust the credential, in the absence of a trusted setup. We prove this new scheme secure under the Computational Diffie Hellman assumption, and Decisional Diffie Hellman assumption, in the Random Oracle Model. We show that this scheme is efficient enough to be used with laptops, and to be integrated into SSI frameworks or any other IMS.

Keywords: Anonymous Credential, Unlinkability, Issuer Indistinguishability, Hidden Issuer, Privacy

DOI 10.56553/popets-2022-0123
Received 2022-02-28; revised 2022-06-15; accepted 2022-06-16.

1 Introduction

Self Sovereign Identity (SSI) [1] has emerged as a new paradigm for Identity Management Systems (IMS). It aims to provide users with a privacy-preserving manner to prove their identities. Allen [2] defines a self-sovereign identity as one satisfying ten principles.

In this paper, we focus on two of them: consent and minimization. Consent states that users must remain in control of their identities by agreeing to their usage by third parties. Minimization states that the system must only disclose minimal information about a user's identity, effectively protecting their privacy. SSI frameworks allow for versatile, user-centric, privacy-preserving IMSs. Many SSI implementations have appeared in recent years, from EL PASSO [3] to the more decentralized Sovrin implementation [4, 5]. Most of them protect privacy by relying on a cryptographic primitive known as Anonymous Credential (AC).

An AC scheme, introduced by Chaum [6] in 1985, allows a user to prove his/her identity elements without disclosing any information other than the one he/she intends to disclose. It relies on the presence of three groups of actors: users, identity issuers, and service providers. Users want to access services offered by one or more service providers. Identity issuers, or simply issuers, certify that a given user has some characteristics. Finally, service providers leverage these certified characteristics, or credentials to manage access to the services they offer. For instance, a user who wants to access a bar will need to prove that he has received his Covid-19 vaccines and that he is of legal age to buy alcohol. If the bar organizes a concert, the user will also have to present a ticket. As the primary role of the service provider in a credential scheme consists in verifying credentials, we refer to service providers as verifiers.

The separation between issuers and verifiers is a key factor for privacy preservation in AC schemes. Establishing a proof of identity most likely requires access to numerous information items. For example, a user that wants to prove to be over 18 years old will typically provide an ID card with a picture, which is then verified in person or by matching the ID to a video or a photograph. An AC scheme allows the user to disclose all extra information (picture, date of birth, name or all other elements on the id card) only to the issuer. The issuer then provides the user with a certification of an identity attribute [7] (being over 18). This takes the form of a digital signature that the user can employ as a credential with the verifier in order to access its services.

However, the separation between the issuer and the verifier is not complete. A verifier can only accept a credential if it trusts the corresponding issuer, and by using a key produced by the issuer itself. This raises

Daniel Bosk: KTH, E-mail: dbosk@kth.se
Davide Frey: Univ Rennes, CNRS, Inria, IRISA, E-mail: davide.frey@inria.fr
Mathieu Gestin: Univ Rennes, CNRS, Inria, IRISA, E-mail: mathieu.gestin@inria.fr
Guillaume Piolle: CentraleSupélec, Inria, Univ Rennes, CNRS, IRISA, E-mail: guillaume.piolle@centralesupelec.fr

important privacy concerns. In the previous Covid-19 example, a verifier may exploit the identity of the issuer (a vaccination center in this case) to infer where the user lives. Similarly, in the case of a certificate stating that a user is 18 years old, knowledge of the issuer may help the verifier infer where the user was born. These pieces of information constitute partial identifiers, which, when combined, may lead to the full identification of a user. This situation can be even worse if the verifier and the issuer collude. Let us assume that a covid-19 vaccination certificate states the place and date of vaccination. If the user is the only person vaccinated at that time on that date, verifier and issuer together can easily identify the user. These privacy concerns can undermine the very objective of SSIs, violating minimization and consent.

Until recently, no one had studied this problem. However, a recent paper [8], which appeared concurrently with the preparation of our own, proposes a scheme addressing the same goal using different building blocks and requiring a trusted setup.

In this paper, we take a different approach. We formalize the key features of hidden-issuer anonymous credentials into a novel cryptographic primitive and instantiate it into a solution that does not require a trusted setup. In doing so, we make four main contributions..

- We introduce the cryptographic *aggregator*, a new primitive that makes it possible to prove the set-membership of an element, without revealing it, and while only knowing a commitment to it. We formally define this new primitive and give a concrete instantiation that works in the absence of a trusted setup.
- We propose a novel credential scheme that provides issuer indistinguishability from the verifier's point of view with an instantiation that does not require a trusted setup. Our scheme uses aggregators to hide the issuer of a given credential among a set of issuers that are trusted by the verifier. This allows the verifier to trust the information in the credential even if it does not know its precise issuer. Our scheme supports non-transferable signatures and signatures-on-committed-message.
- We prove the security of our scheme. Specifically, we prove its soundness under the Computational Diffie-Hellman assumption, in the Random Oracle model. We also prove the indistinguishability of the issuer among the whole set of issuers trusted by one verifier under the Decisional Diffie-Hellman assumption.
- We provide an open-source implementation of our scheme in Rust and evaluate its performance.

2 Problem Statement

We aim to provide an AC that hides not only the identity of the user, but also that of the issuer, thereby reducing the associated privacy risks. In the rest of this section, we define the properties we seek for our Hidden-Issuer Anonymous-Credential scheme, before describing our adversary model.

Properties. We focus on two of the principles enunciated by Allen [2]: minimization and consent. Existing credential schemes satisfy these principles only partially because, as we discussed, knowledge of the issuer can reveal personal information about the user. Specifically, they provide minimization and consent (i) by sharing only the identity elements that were selected by the user, (ii) by providing unlinkability, i.e. by ensuring that multiple uses of the same credential cannot be linked together, and (iii) by providing a way for the issuer to sign committed message, i.e. the issuer is able to sign a message without learning the actual content of this message.

In order to hide the identity of the issuer from the verifier, we introduce two additional properties:

- **Issuer indistinguishability** states that, given two credentials certifying the same types of messages, a verifier should not be able to say if the issuers of these credential are distinct or not.
- **Trusted issuer** states that the verifier should be certain that the credential it accepts has been issued by an issuer it trusts. This would be trivial in a classic Anonymous Credential scheme, but becomes relevant once we hide the identity of the issuer.

The credential should satisfy these properties while being used an arbitrary number of times (multi-show) and with an arbitrary number of different verifiers (multi-verifier). It should also have an optional non-transferable-credential feature. To ensure usability, it should be efficiently computable with low-end modern computers for its three types of actors. Finally, the scheme should not require a trusted setup as this would either introduce a trusted third party or require complex distributed synchronization protocols [9, 10].

Need for a Decentralized Solution. At first glance, it might appear that the above properties could be provided by a central anonymizer—possibly chosen by a decentralized leader-election algorithm—to which issuers would delegate the signature of all credentials. However, this solution would defeat the very point of a decentralized IMS, adding the (strong) hypothesis that all issuers and verifiers trust the same anonymizer, creating a single point of failure. A verifier would not be able to choose specific trusted issuers and would need to trust *all* the issuers the anonymizer represents. A centralized anonymizer would also complicate the issuance workflow. Let us consider a bar that verifies tickets for an event it organizes. The bar can trust multiple ticket resellers but it can also sell tickets directly to its clients. With a central anonymizer, the bar would thus need to register to the anonymizer responsible for ticket sales. This would create unnecessary overhead, particularly if the bar sells tickets only occasionally. Moreover, if the central anonymizer is responsible for a specific population, e.g. a region or a country, its usage would still leak information about this region. A decentralized IMS using AC should instead (i) give verifiers the freedom to trust their preferred issuers, and (ii) give users the freedom to choose verifiers based on the issuers they trust.

Adversary Model. We consider two adversaries: 1) A malicious verifier colludes with a subset or all of its trusted issuers. It has access to their secret keys, and to all previously issued credentials. It tries to reveal the identity of the owner or of the issuer of a credential. 2) A malicious user tries to use credentials on messages not signed by any trusted issuers. He can request multiple signatures from the signing oracles of each trusted issuer, and obtain all their public keys.

Colluding parties can share any information with each other. We consider the security of our system in an isolated world. The only potential flaws considered here are the ones within the scheme. We provide a formal adversary model in Section 5.1.

3 Overview

We propose a novel Anonymous Credential scheme that does not disclose the identity of a credential's issuer. Our scheme introduces a randomization process, run by the user, that hides the issuer of a credential among the set of issuers trusted by the verifier. To make this possible, each verifier publishes the list of its trusted issuers in the form of a set, represented by an *aggregator*, a new primitive, central to an HIAC scheme. The aggregator allows a user to prove that a randomized issuer's public key belongs to a given set, while not knowing the associated secret key, and while being able to use this key in a subsequent signature-verification process. This aggregator is independent from the rest of the signature scheme. The only interface between the aggregator and the signature is the randomized issuer's public key.

When the user wants to use a credential received from an issuer, he randomizes the issuer's public key, while providing a proof that the issuer's key belongs to the verifier's aggregator. This randomize-and-prove process hides the identity of the issuer, while allowing the verifier to ensure that it belongs to its trusted list.

Our Hidden Issuer Anonymous Credential (HIAC) scheme combines a modified Pointcheval Sanders (PS) signature [11], and two Aggregator instances, one for each secret key used by PS. We use PS because it does not require a trusted setup, and it is trapdoor-free.

4 Building Blocks

Before detailing the operation of our Anonymous Credential scheme, we introduce some notation and the definitions of our main building blocks.

– Let $\mathbb{G}$ be a group. We note by $\langle g \rangle = \mathbb{G}$ the fact that g is the generator of $\mathbb{G}$, and we note $1_{\mathbb{G}}$ the neutral element of $\mathbb{G}$.
– Let $\mathbb{Z}$ be a set, we note by $a \leftarrow\!\!\$\, \mathbb{Z}$ the fact that a is chosen uniformly at random in the set $\mathbb{Z}$.

Signature Scheme. A digital signature [12] can be seen as a proof that a given message m was certified by an issuer. Here, we give a formal definition for a digital signature scheme:

Definition 1. A digital signature scheme Σ is defined as a tuple of four algorithms (**Setup, KeyGen, Sign, Verify**).

1. $(pp) \leftarrow \mathbf{Setup}(\lambda)$: On input of a security parameter λ, the algorithm outputs pp, a description of public parameters whose security level depends on λ;
2. $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathbf{KeyGen}(pp)$: on input of a setup parameter pp, the algorithm computes a secret/public key pair $(\mathsf{sk}, \mathsf{pk})$;
3. $(\sigma) \leftarrow \mathbf{Sign}(pp, m, \mathsf{sk})$: on input of a setup parameter pp, a message m, and a secret key sk, the algorithm outputs a signature σ; and
4. $\{0, 1\} \leftarrow \mathbf{Verify}(pp, m, \mathsf{pk}, \sigma)$: on input of a setup parameter pp, a message m, a public key pk, and a signature σ, the algorithm outputs 1 if σ is a

valid signature on the message m by the secret key associated with pk, and 0 otherwise.

Security Notion: A signature is secure if no adversary can create a signature in the name of some other issuer, and if no adversary can modify the previously signed message without invalidating the signature. The standard security notion for a signature scheme is the Existential Unforgeability under Chosen Message Attack (EUF-CMA) [13]. This notion states that the scheme is resistant against an adversary that can make an arbitrary number of queries to a signing oracle. An **Anonymous Credential** [6] is a digital signature that provides unlinkability between its uses, non-transferability, and the ability to sign committed message.

The unlinkability property states that when a user provides credentials multiple times to a verifier, the latter cannot tell whether they come from the same original signature. This property is often enabled by an additional algorithm, run by the user, called ***randomize***. Non-transferability makes it possible for a verifier to ensure that a signature was issued to the user that presents it. Finally, the ability to sign committed message makes it possible for an issuer to sign a commitment to a message m, without learning the value of the message it signs. The user is then able to open the commitment, thus obtaining a signature on message m.

Hash Function. The security of our signature scheme is based on cryptographic hash functions [12].

Definition 2. A hash function is a map H which takes an element of arbitrary size $A \in \mathbb{Z}$ and outputs an element $B \in \{1, \cdots, p\}$ of fixed size p. A cryptographic hash function has two additional properties:

1. (Pre-image resistance) given $a = \mathsf{H}(b)$, no probabilistic polynomial time (PPT) adversary can find b with non-negligible probability; and
2. (Collision resistance) no PPT adversary can find, with non-negligible probability, $x \in \mathbb{Z}$ and $y \in \mathbb{Z}$ such that $\mathsf{H}(x) = \mathsf{H}(y)$ and $x \neq y$.

Bilinear Pairing. Our scheme needs an efficient bilinear pairing.

Definition 3. Let $\mathbb{G}_1 = \langle g_1 \rangle$, $\mathbb{G}_2 = \langle g_2 \rangle$, $\mathbb{G}_T = \langle g_T \rangle$, three groups of prime order p. A bilinear pairing is a map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ that has the bilinearity property. This means that $e(g_1^a, g_2) = e(g_1, g_2)^a = e(g_1, g_2^a)$, for all $a \in \{0, \cdots, p-1\}$;

To be efficiently used in cryptography we also require this map to be non-degenerate and efficiently computable:

- (Non degeneracy) $\forall A \in \mathbb{G}_1, A \neq 1_{\mathbb{G}_1}, \exists B \in \mathbb{G}_2$ such that $e(A, B) \neq 1_{\mathbb{G}_T}$ and $\forall B \in \mathbb{G}_2, B \neq 1_{\mathbb{G}_2}, \exists A \in \mathbb{G}_1$ such that $e(A, B) \neq 1_{\mathbb{G}_T}$;
- (Efficient computability) there is an efficient algorithm to compute e.

Furthermore, we can classify bilinear pairings in three types [14]. In our construction, we will use a type-3 bilinear pairing. Type-3 bilinear pairings are pairings where $\mathbb{G}_1 \neq \mathbb{G}_2 \neq \mathbb{G}_T$ and there is no efficiently computable morphism $\psi_1 : \mathbb{G}_2 \to \mathbb{G}_1$ or $\psi_2 : \mathbb{G}_1 \to \mathbb{G}_2$.

Zero-Knowledge Proof. The actors of our signature scheme need to prove knowledge of values, while keeping them secret. A Zero-Knowledge Proof (ZKP) system is a tool that matches this requirement by allowing a prover to prove to a verifier that it knows a secret statement, without revealing it.

In this paper, we will use the convenient notation of ZKP protocols introduced by Camenisch and Stadler [15]. The aim of the notation is to explain what is proven, and what is hidden, in a compact way. For example, the notation for a ZKP about the knowledge of the exponent α in the expression $y = g^\alpha$ is $ZKPoK\{(\alpha) : y = g^\alpha\}$. Where the elements between parentheses (α) are the elements known only by the prover, and the equations on the right explain how the prover commits to those elements. The letters on the left define what proof is being conducted. Here, ZKPoK means Zero Knowledge Proof of Knowledge, which explains that the prover will prove knowledge of an element. We will also use ZKPoE, which is a Zero Knowledge Proof of Equality. The goal of a ZKPoE is to prove that different elements have the same exponent. Chaum and Pedersen presented a version with only two elements in 1993 [16], but this construction can be generalized to any number of elements. We will use the prefix "NI-" to refer to non-interactive proofs. An instantiation of NI-ZKPoK in the random oracle model was described by Fiat and Shamir [17], and later modified and proved secure by Bernhard, Pereira, and Warinschi [18].

Assumptions. Our signature scheme relies on several complexity assumptions. The first is the Discrete Logarithm (DL) assumption [19], the second is the Computational Diffie Hellman Assumption (CDH) [19], and the third is the Decisional Diffie Hellman (DDH) assumption [19]. We also assume that there exists a type-3 bilinear pairing, e, defined on elliptic curves over a finite field and a cryptographic hash function, H.

5 Formal Definitions

5.1 Hidden Issuer Anonymous Credential

A Hidden Issuer Anonymous Credential (HIAC) is an Anonymous Credential with the issuer-indistinguishability and trusted-issuer properties.

Definition 4. A Hidden Issuer Anonymous Credential scheme Σ is defined as a tuple of six algorithms (**Setup, IssuerKeygen, Sign, VerifierSetup, Randomize, VerifyRandomized**) defined as follows:

1. $(pp) \leftarrow$ **Setup**(λ): On input of a security parameter λ, the algorithm outputs pp, a description of public parameters whose security level depend on λ;
2. $(isk, ipk) \leftarrow$ **IssuerKeyGen**(pp): on input of pp, an issuer computes a secret/public key pair (isk, ipk);
3. $(\sigma) \leftarrow$ **Sign**(pp, m, isk): on input of pp, of a message m, and of its issuer secret key isk, an issuer outputs a signature σ;
4. $(\mathcal{S}, \mathsf{aux}_{\mathcal{S}}) \leftarrow$ **VerifierSetup**$(pp, \mathcal{S}')$: on input of pp, and of a set of k' issuers $\mathcal{S}'$, a verifier outputs $\mathcal{S}$ a selection of $k \leq k'$ trusted issuers in $\mathcal{S}'$. The algorithm also outputs auxiliary information $\mathsf{aux}_{\mathcal{S}} = (vpk, vsk)$, about $\mathcal{S}$; where vpk is public and vsk is secret.
5. $(ipk', \pi_{ipk'}, \sigma') \leftarrow$ **Randomize**$(pp,\ ipk,\ vpk,\ \mathcal{S},\ \sigma)$: on input of pp, of a signature σ, of a public key ipk of the issuer of σ, of a set of issuers $\mathcal{S}$, and of public auxiliary information vpk, a user:
   (a) Checks that vpk is built using elements from $\mathcal{S}$;
   (b) Produces ipk', a randomized version of ipk;
   (c) Produces a proof $\pi_{ipk'}$ that ipk' is an element of $\mathcal{S}$, using vpk;
   (d) Produces a randomized signature σ' using σ, ipk', and $\pi_{ipk'}$.

   The algorithm outputs ipk', $\pi_{ipk'}$, and σ'.
6. $\{0,1\} \leftarrow$ **VerifyRandomized**$(pp,\ vsk,\ ipk',\ \pi_{ipk'},\ \sigma',\ m)$: on input of pp, of verifier secret auxiliary information vsk, of a randomized issuer key ipk', of a proof $\pi_{ipk'}$, of a randomized signature σ', and of a message m, a verifier outputs 1 if the proof $\pi_{ipk'}$ is valid, and if σ' is a valid signature on message m, signed with the secret key associated with ipk'. The algorithm outputs 0 otherwise.

Security notions: We define three security notions: *correctness*, *issuer indistinguishability*, and *trusted issuer*. *Issuer indistinguishability* extends the unlinkability property of classical Anonymous Credential schemes.

Definition 5. (Correctness) A HIAC scheme is correct, if for any honestly built tuple $(pp, vsk,\ ipk', \pi_{ipk'}, \sigma', m)$, **VerifyRandomized**$(pp, vsk,\ ipk', \pi_{ipk'}, \sigma', m)$ always outputs 1.

Definition 6. (Issuer Indistinguishability) We define the **IssuerIndistinguishGame**$(\mathcal{A}, \mathcal{C})$, for a challenger $\mathcal{C}$ that represents a user, and the actions preformed by the issuers, and an adversary $\mathcal{A}$ that represents a malicious verifier colluding with *all* its trusted issuers[1]:

- **Setup**: First, $\mathcal{C}$ runs the HIAC **Setup** algorithm, and two rounds of the **Keygen** algorithm to obtain two issuer key pairs $(\{ipk_1, ipk_2\}, \{isk_1, isk_2\})$. Second, $\mathcal{A}$ runs **VerifierSetup** to build a set $\mathcal{S}$ with the two issuers, and the associated auxiliary information (vpk, vsk). Third, $\mathcal{C}$ computes $\sigma_1 = \mathbf{Sign}(pp, isk_1, m)$ and $\sigma_2 = \mathbf{Sign}(pp, isk_2, m)$ on the same message $m \leftarrow\!\$\ \mathbb{Z}_p$. $\mathcal{A}$ is given $(pp, \{ipk_1, ipk_2\}, \{isk_1, isk_2\}, \sigma_1, \sigma_2, m, \mathcal{S}, vpk, vsk)$. Finally, $\mathcal{C}$ chooses a random number $I \leftarrow\!\$\ \{0,1\}$.
- **Queries**: $\mathcal{A}$ adaptively requests a finite number of randomized signatures. $\mathcal{C}$ answers each query by returning $(ipk'_I, \pi_{ipk'_I}, \sigma'_I) \leftarrow$ **Randomize**$(pp,\ ipk_I,\ vpk,\ \mathcal{S},\ \sigma_I)$.
- **Output**: $\mathcal{A}$ eventually outputs $I' \in \{0,1\}$. The game outputs 1 if $I' = I$. The game outputs 0 otherwise.

A HIAC scheme is said to be Issuer-Indistinguishable iff, for a negligible ϵ, the probability for a PPT Adversary $\mathcal{A}$ to win the **IssuerIndistinguishGame** against a challenger $\mathcal{C}$ is:

$$\Pr[\mathbf{IssuerIndistinguishGame}(\mathcal{A}, \mathcal{C}) = 1] \leq \frac{1}{2} + \epsilon$$

Definition 7. (Trusted issuer) We define the **TrustedIssuerGame**$(\mathcal{A}, \mathcal{C})$, for a challenger $\mathcal{C}$ that represents a verifier and its trusted issuers, and an adversary $\mathcal{A}$ that represents a malicious user:

- **Setup**: $\mathcal{C}$ runs HIAC's **Setup**, **IssuerKeygen**, and **VerifierSetup** algorithms to obtain public parameters pp, k issuers' key pairs $\{(isk_i, ipk_i)\}_{i=1}^{k}$ and one verifier's information pair $(\mathcal{S}, \mathsf{aux}_{\mathcal{S}})$. $\mathcal{A}$ is given $\{(ipk_i)\}_{i=1}^{k}$, $\mathcal{S}$, and public information vpk.

[1] The formal definition of the adversary model only takes into account two trusted issuers. However, the privacy of the users relies on the size of the underlying anonymity set. Therefore, the size of a verifier's trusted issuer set should be large enough to hide the users.

- **Queries**: $\mathcal{A}$ adaptively requests signatures on at most n messages $m_1, ..., m_n$ to the k issuers. $\mathcal{C}$ answers each query by returning for each issuer $\{\sigma_{(i,j)} \leftarrow$ **Sign**$(isk_j, m_i)\}_{i=1}^{n}$.
- **Output**: $\mathcal{A}$ eventually outputs a randomized message-signature pair $(m^*, (ipk^*, \pi^*_{ipk}, \sigma^*))$. The game outputs 1 if **VerifyRandomized**$(pp$, vsk, ipk^*, π^*_{ipk}, $m^*) = 1 \wedge ipk^* \neq \mathcal{S}$, and 0 otherwise.

An HIAC scheme is said to have the trusted-issuer property if, for a negligible value, ϵ, the probability for a Probabilistic Polynomial Time (PPT) Adversary $\mathcal{A}$ to win **TrustedIssuerGame** against a challenger $\mathcal{C}$ is:

$$\Pr[\textbf{TrustedIssuerGame}(\mathcal{A}, \mathcal{C}) = 1] \leq \epsilon$$

Definition 8. (Existential Unforgeability Against Chosen Message Attack) We define the **EUFCMAGame** ($\mathcal{A}$, $\mathcal{C}$), for a challenger $\mathcal{C}$ that represents a verifier and its trusted issuers, and an adversary $\mathcal{A}$ that represents a malicious user:

- **Setup**: $\mathcal{C}$ runs the HIAC's **Setup**, **IssuerKeygen**, and **VerifierSetup** algorithms to obtain the public parameters pp, k issuers' key pairs $\{(isk_i, ipk_i)\}_{i=1}^{k}$ and one verifier's information pair $(\mathcal{S}, \mathsf{aux}_{\mathcal{S}})$. $\mathcal{A}$ is given $\{(ipk_i)\}_{i=1}^{k}$, $\mathcal{S}$, and vpk.
- **Queries**: $\mathcal{A}$ adaptively requests signatures on at most n messages $m_1, ..., m_n$ from the k issuers. $\mathcal{C}$ answers each query by returning, for each issuer, $\{\{\sigma_{(i,j)} \leftarrow$ **Sign**$(isk_j, m_i)\}_{i=1}^{n}\}_{j=1}^{k}$.
- **Output**: $\mathcal{A}$ eventually outputs a randomized message-signature pair $(m^*, (ipk^*, \pi^*_{ipk}, \sigma^*))$. The game outputs 1 if **VerifyRandomized**$(pp$, vsk, ipk^*, π^*_{ipk}, $m^*) = 1 \wedge m^* \neq m_i \, \forall i$, and 0 otherwise.

An HIAC scheme is said to have the EUF-CMA property if for a negligible value, ϵ, and M the space of possible messages, the probability for a PPT Adversary $\mathcal{A}$ to win the **EUFCMAGame** against a challenger $\mathcal{C}$ is:

$$\Pr[\textbf{EUFCMAGame}(\mathcal{A}, \mathcal{C}) = 1] \leq \frac{1}{M} + \epsilon$$

Definition 9. A Hidden Issuer Anonymous credential scheme is secure iff it achieves *correctness*, *EUF-CMA*, *issuer indistinguishability*, and *trusted issuer*.

5.2 Aggregator

To instantiate an HIAC scheme, we need an object which allows a verifier to commit to a set of values, and which offers a way for a user to prove that some value is indeed in the commitment set, without revealing this value, and while only knowing a commitment to this value. State-of-the-art primitives do not offer such a possibility. We therefore introduce a novel cryptographic object: the *aggregator*. In our HIAC scheme, the verifier builds the aggregator and uses it as a commitment to a set of trusted issuers. But we conjecture that other designs for an HIAC scheme would also need an aggregator or a similar object. An aggregator provides one witness for each trusted issuer. Each such witness can be used to prove that a specific issuer's public key (or a commitment to an issuer's secret key) is accumulated in the aggregator. The verifier is the only actor able to verify this set-membership proof. The user can further make this proof element indistinguishable by adding random elements to the witness and the issuer key. An aggregator is defined by the five following algorithms:

Definition 10. – $(pp) \leftarrow$ **Setup**(λ): on input of setup parameter λ, the algorithm outputs pp, a description of public parameters whose security depends on λ;

- $(\mathsf{Agg}, \mathsf{aux}, \mathsf{sk}) \leftarrow$**Gen**$(pp, \mathcal{S})$: on input of pp and of a set $\mathcal{S}$, the algorithm outputs aggregator Agg, auxiliary information aux, and a secret verification key sk;
- $\{0,1\} \leftarrow$ **IntegrityVerification**$(pp, \mathcal{S}, \mathsf{Agg}, \mathsf{aux})$: on input of pp, a set $\mathcal{S}$, aggregator Agg, and auxiliary parameters aux, the algorithm outputs 1 if Agg is valid, and if it aggregates all the elements in $\mathcal{S}$. The algorithm outputs 0 otherwise;
- $(C', \pi_s) \leftarrow$**WitCreate**$(pp$, Agg, aux, $C)$: on input of pp, aggregator Agg, auxiliary parameters aux, and a cryptographic commitment C to an element $s \in \mathcal{S}$, the algorithm outputs C' a new commitment to s, and a proof π_s which proves that C' is a commitment to an element aggregated in Agg.
- $\{0,1\} \leftarrow$ **Verify**$(pp, \mathsf{Agg}, \mathsf{aux}, \mathsf{sk}, C', \pi_s)$: on input of pp, aggregator Agg, auxiliary parameters aux, a cryptographic commitment C' to an element $s \in \mathcal{S}$, a secret verification key sk, and a witness W_s, the algorithm outputs 1 if π_s is valid and 0 otherwise.

Security notions for a secure aggregator can be derived from the needs expressed in the HIAC formal definition. The first, *Collision-freedom*, states that the probability for an adversary to find a proof of set-membership for a non accumulated element is negligible. The trusted issuer property of the formal HIAC definition is derived from the collision-freedom property of its aggregator.

Definition 11. (Collision Freedom) We define the **CollisionFreedomGame**$(\mathcal{A}, \mathcal{C})$, for a challenger $\mathcal{C}$ that represents a verifier and its trusted issuers, and an adversary $\mathcal{A}$ that represents a malicious user:

- **Setup**: $\mathcal{C}$ runs the aggregator **Setup** algorithm, chooses a set $\mathcal{S}$, runs the **Gen** algorithm to build a

commitment to $\mathcal{S}$, and the associated verification key. $\mathcal{A}$ is given the aggregator's public information.
- **Output**: $\mathcal{A}$ outputs C^* a commitment to s^* and π_s^*. The game outputs 1 if $s^* \notin \mathcal{S}$, and **Verify**$(pp, \mathsf{Agg}, \mathsf{aux}, \mathsf{sk}, C^*, \pi_s^*) = 1$, or 0 otherwise.

An Aggregator is said to be element-indistinguishable if for a negligible ϵ, the probability for a PPT Adversary $\mathcal{A}$ can to win the **CollisionfreedomGame** is:

$$\Pr[\textbf{CollisionfreedomGame}(\mathcal{A}, \mathcal{C}) = 1] \leq \epsilon$$

The second security notion, *Element Indistinguishabity* states that, while the verifier running the **Verify** algorithm knows that commitment C' is associated with one of the elements of $\mathcal{S}$, it cannot learn the actual element committed to. The issuer-indistinguishability property of the HIAC definition is partially derived from the element-indistinguishability property of its aggregator.

Definition 12. (Element indistinguishability) We define the **ElementIndistinguishGame**$(\mathcal{A}, \mathcal{C})$, for a challenger $\mathcal{C}$ that represents a user, and the actions preformed by the issuers, and an adversary $\mathcal{A}$ that represents a malicious verifier:
- **Setup**: $\mathcal{C}$ runs the aggregator **Setup** algorithm, and chooses a set $\mathcal{S}$ with at least 2 elements. Then, $\mathcal{A}$ runs the **Gen** algorithm to build a commitment to the set $\mathcal{S}$, and the associated verification key. Finally, $\mathcal{C}$ chooses $s \in \mathcal{S}$. $\mathcal{A}$ is given the set of secret values aggregated in the aggregator, the commitment to the issuer set, and the secret verification key.
- **Queries**: $\mathcal{A}$ adaptively requests q randomized commitments to elements of $\mathcal{S}$, and the associated proofs. $\mathcal{C}$ answers each query by returning $(C', \pi_s) \leftarrow$ **WitCreate**$(pp, \mathsf{Agg}, \mathsf{aux}, C)$ a commitment to $s \in \mathcal{S}$.
- **Output**: $\mathcal{A}$ eventually outputs $s' \in \mathcal{S}$. The game outputs 1 if $s' = s$. The game outputs 0 otherwise.

An Aggregator is said to be element-indistinguishable if for a negligible value, ϵ, and k trusted issuers, the probability for a PPT Adversary $\mathcal{A}$ to win the **ElementIndistinguishGame** against a challenger $\mathcal{C}$ is:

$$\Pr[\textbf{ElementIndistinguishGame}(\mathcal{A}, \mathcal{C}) = 1] \leq \frac{1}{k} + \epsilon$$

The last security notion is *correctness*:

Definition 13. An aggregator scheme is correct iff, for any honestly built tuple $(pp, \mathsf{Agg}, \mathsf{aux}, \mathsf{sk}, C', \pi_s)$, the **Verify** algorithm always outputs 1.

Definition 14. An Aggregator is secure iff it achieves *correctness*, *collision-freedom*, and *element indistinguishability.*

If this aggregator is embedded correctly into a Hidden Issuer Anonymous Credential scheme, then the *Trusted Issuer* property relies entirely on the aggregator's *Collision Freedom* property.

6 Instantiation

Next, we instantiate our aggregator object and our HIAC scheme. We start by describing a non-interactive version of our scheme. Then, we present our final, interactive version. Appendix B shows that we can add non transferability and signature on committed message to the scheme presented in this section.

Our setup is trust free. None of the actors needs to trust the constructions made by the other actors. The integrity of the elements published by each actor can be verified algorithmically. In this section, we assume all issuers of the system issue the same types of messages. We explain how to verify this assumption in Section 7.

6.1 Non-Interactive HIAC

6.1.1 Aggregator

Our aggregator consists of five algorithms: **Setup**, **Gen**, **IntegrityVerification**, **WitCreate**, and **Verify**. Unlike in the formal definition of Section 5.2, we pass the random values r_1 and r_2 to **WitCreate**. This makes it possible to map elements of the set-membership proof to the actual randomized signature in the instantiation of our HIAC scheme. As we mentioned, the aggregator is independent from the HIAC scheme, and can be used with other signature schemes based on bilinear pairings to provide issuer indistinguishability.

Protocol 1. 1. $(pp) \leftarrow$**Setup**(λ): on input of a security parameter λ, the algorithm chooses three prime-order p groups with the associated generators $\mathbb{G}_1 = \langle g_1 \rangle$, $\mathbb{G}_2 = \langle g_2 \rangle$, and $\mathbb{G}_T = \langle g_T \rangle$. The algorithm then chooses a type 3 bilinear pairing e such that $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$. The algorithm outputs public parameters $pp = (p, \mathbb{G}_1, g_1, \mathbb{G}_2, g_2, \mathbb{G}_t, g_T, e)$.

2. $(\mathsf{Agg}, \mathsf{aux}, \mathsf{sk}, T_1) \leftarrow$**Gen**$(pp, \mathcal{S})$: on input of pp and of a set $\mathcal{S} = \{g_1^{\frac{1}{x_1}}, \cdots, g_1^{\frac{1}{x_k}}\}, x_i \in \mathbb{Z}_p, \forall i \in \mathbb{Z}_p^*$, $k \geq 2$, the algorithm computes a secret key $\mathsf{sk} \leftarrow\$ \mathbb{Z}_p$, an aggregator $\mathsf{Agg} = (\mathcal{S}, W)$, with witness set $W = \{(W)_{(j)} = g_1^{\mathsf{sk}\frac{1}{x_j}}\}_{j=1}^{k}$. The algorithm also computes some auxiliary information $\mathsf{aux} = \beta$, with $\beta = \mathrm{NI}-\mathrm{ZKPoE}\{(\mathsf{sk}) : \bigwedge_{i=1}^{k} (W)_{(i)} = (\mathcal{S})_{(i)}^{\mathsf{sk}}\}$. The algorithm finally outputs $(\mathsf{Agg}, \mathsf{aux}, \mathsf{sk})$;

3. $\{0,1\} \leftarrow$ **IntegrityVerification**$(pp, \mathcal{S}_{\mathsf{Comm}}, \mathsf{Agg}, \mathsf{aux})$: on input of pp, of a set $\mathcal{S}_{\mathsf{Comm}} = \{g_2^{x_1}, \cdots, g_2^{x_k}\}, x_i \in \mathbb{Z}_p, \forall i \in \{1, \cdots, k\}$, of an aggregator $\mathsf{Agg} = (\mathcal{S}, W)$, and of the auxiliary parameter β, the algorithm outputs 1 if β is valid and 0 otherwise;
4. $(C', \pi_s) \leftarrow$ **WitCreate**$(pp, \mathsf{Agg}, \mathsf{aux}, C, r_1, r_2)$: on input of pp, of an aggregator $\mathsf{Agg} = (\mathcal{S}, W)$, of auxiliary parameters aux, of a cryptographic commitment $C = g_2^{x_l}, l \in \{1, \cdots, k\}$, and of two randomly chosen numbers r_1, r_2, the algorithm computes $C' = C^{r_1}$, and a proof $\pi_s = ((W)'_{(l)}, h)$, with $(W)'_{(l)} = (W)^{r_2}_{(l)}$ and $h = g_2^{r_1 r_2}$; the algorithm outputs (C', π_s).
5. $\{0,1\} \leftarrow$ **Verify**$(pp, \mathsf{sk}, C', \pi_s)$: on input of pp, of the secret aggregator's verification key sk, of a randomized commitment C', and of a proof $\pi_s = ((W)'_{(l)}, h)$, the algorithm outputs 1 if: $e((W)'_{(l)}, C') \overset{?}{=} e(g_1^{\mathsf{sk}}, h)$; and 0 otherwise.

Security Analysis

Theorem 1. The aggregator presented by Protocol 1 is correct.

Proof sketch 1. The proof of this theorem, in Appendix E.2, uses a correctly built element $e((W)'_{(i)}, C')$ to prove that it is equivalent to $e(g_T^{sk}, h)$.

Theorem 2. (Collision Freedom) The aggregator presented by Protocol 1 is collision free, under the CDH assumption (Definition 11).[2]

Proof sketch 2. The proof of Theorem 2, in Appendix E.3, shows that under the CDH assumption the secret value sk used in $e(g_T^{\mathsf{sk}}, h)$ forces the adversary to use exactly one witness to build its $(W)'_l$. Then it shows that C^* contains exactly one x_i.[3]

Theorem 3. The scheme presented here is *Element Indistinguishable* (Definition 12).

Proof sketch 3. The proof of the Theorem 3, in Appendix E.4, proceeds by analyzing the elements the adversary is given. Either there is no linear relationship between these elements (which, under the DDH assumption, means that no adversary can decide which element is being proven to be in $\mathcal{S}$) or the elements are symmetrical (and thus they do not depend on the variables, or they are distributed uniformly at random in the group).

6.1.2 Signature Scheme

Our Hidden Issuer Anonymous Credential (HIAC) scheme relies on a modified Pointcheval Sanders (PS) scheme [11] and uses six algorithms. The first three come from a classical signature scheme, namely **Setup** which produces the shared material, **IssuerKeygen** which allows an issuer to build a key pair, and **Sign** which allows an issuer to sign a credential. The three others provide the issuer-indistinguishability and trusted-issuer properties. The verifier runs the **VerifierSetup** algorithm to select a set of trusted issuers and generate the aggregator. In this non-interactive version of HIAC, the verifier needs to do this before each transaction to prevent malicious issuers from extracting the verifier's private key from the aggregator. Once it has the aggregator, the user runs the **Randomize** algorithm to randomize the credential and the associated issuer's key, and to create the associated proof of set-membership. Finally, the verifier runs **VerifyRandomized** to verify the proof and the authenticity of the randomized credential.

Protocol 2. 1. $(pp) \leftarrow$ **Setup**(λ): on input of a security parameter λ , the algorithm chooses three prime-order p groups with the associated generators $\mathbb{G}_1 = \langle g_1 \rangle$, $\mathbb{G}_2 = \langle g_2 \rangle$, and $\mathbb{G}_T = \langle g_T \rangle$. The algorithm then chooses a type 3 bilinear pairing e such that $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. Finally, it chooses a cryptographic hash function $\mathsf{H} : \mathbb{Z} \to \mathbb{Z}_p$. The algorithm outputs $pp = (p, \mathbb{G}_1, g_1, \mathbb{G}_2, g_2, \mathbb{G}_t, g_T, e, \mathsf{H})$.

2. $(isk_I, ipk_I) \leftarrow$ **IssuerKeygen**(pp): on input of pp, an issuer I computes a secret key $isk_I = (x_I, y_I)$, with $x_I, y_I \leftarrow\!\$\ \mathbb{Z}_p^*$ and the associated public key $ipk_I = (X^{(I)}, \bar{X}_1^{(I)}, Y_1^{(I)}, \bar{Y}_1^{(I)}, Y_2^{(I)}) = (g_2^{x_I}, g_1^{\frac{1}{x_I}}, g_1^{y_I}, g_1^{\frac{1}{y_I}}, g_2^{y_I})$.[4] The algorithm outputs (isk_I, ipk_I).
3. $(\sigma, R_y) \leftarrow$ **Sign**(pp, m, isk_I, u, ϕ): on input of pp, of a message m, of an issuer's secret key $isk_I = (x_I, y_I)$, of a value given by the user $u = (Y_1^{(I)})^{R_y} \cdot g_1^R$, where R_y and R are secret random values, and of ϕ, a

[2] The *collision-freedom* proof uses the fact that the value g_1^{sk} is known only to the verifier. But, if a given Aggregator was used multiple times, it would be possible for a malicious Issuer I to compute and publish the value $g_1^{\mathsf{sk}} = (W)^{x_I}_{(I)}$. For this reason, the non-interactive HIAC scheme computes a new aggregator at every transaction (cfr. Section 6.1.2).

[3] In anticipation of the Hidden Issuer Anonymous Credential (HIAC) protocol, we will prove Theorem 2 with an extended version of the adversary model, in which the adversary has access to the extra values $g_1^{x_i}, \forall i \in \{1, \cdots, k\}$.

[4] In the PS version of the signature, $g_1^{\frac{1}{x_I}}$ and $g_1^{\frac{1}{y_I}}$ are not disclosed. We show in the proof section that, thanks to the CDH assumption, this modification does not impact the security of the signature.

ZKPoK of u, ϕ :NI-ZKPoK$\{(R_y R) : u = (Y_1^{(I)})^{R_y} \cdot g_1^R\}$, the issuer initially outputs a signature $\sigma^\Delta = (h_1, h_2, \sigma_1^\Delta, \sigma_2^\Delta)$, where $h_1 = g_1^{r(I,1)}$, $h_2 = g_1^{r(I,2)}$, $\sigma_1^\Delta = (g_1^{x_I} \cdot u^{\mathsf{H}(m)})^{r(I,1)}$, $\sigma_2^\Delta = (g_1^{x_I} \cdot u^{\mathsf{H}(\mathsf{H}(m))})^{r(I,2)}$, with $r_{(I,1)}, r_{(I,2)} \leftarrow\!\!\$ \ \mathbb{Z}_p^*$. Then the user transforms the signature σ^Δ into $\sigma = (h_1, h_2, \sigma_1 = \sigma_1^\Delta \cdot h_1^{-R\mathsf{H}(m)}$, $\sigma_2 = \sigma_2^\Delta \cdot h_2^{-R\mathsf{H}(\mathsf{H}(m))})$. The user stores σ and R_y.

4. $(vpk, vsk) \leftarrow$ **VerifierSetup**$(pp, \{ipk_i\}_{i=1}^k)$: on input of pp and of trusted issuers' public keys $\{ipk_i\}_{i=1}^k$, the verifier commits to the issuer's secret keys x by building a new aggregator $(\mathsf{Agg}_x, \mathsf{sk}_x, \mathsf{aux}_x)$ = Aggregator·**Gen**$(pp, \mathcal{S}_x)$. Symmetrically, the verifier also commits to the issuer's secret key y by building another aggregator $(\mathsf{Agg}_y, \mathsf{sk}_y, \mathsf{aux}_y)$ = Aggregator·**Gen**$(pp, \mathcal{S}_y)$. The algorithm outputs $vsk = (\mathsf{sk}_x, \mathsf{sk}_y)$ and $vpk = (\mathsf{Agg}_x, \mathsf{aux}_x, \mathsf{Agg}_y, \mathsf{aux}_y)$.
5. $(X^{(I)'}, Y_2^{(I)'}, \pi_x, \pi_y, \sigma') \leftarrow$ **Randomize**$(pp, ipk, vpk, \mathcal{S}, \sigma, R_y)$: on input of pp, of a signature $\sigma = (h_1, h_2, \sigma_1, \sigma_2)$, of a verifier public key $vpk = (\mathsf{Agg}_x, \mathsf{aux}_x, \mathsf{Agg}_y, \mathsf{aux}_y)$, and of a secret value R_y generated by the user in step 3, and associated with σ, this algorithm, run by a user:
   (a) Verifies if:
   $$\text{Aggregator}\cdot\textbf{IntegrityVerification}(pp, \{X^{(j)}\}_{j=1}^k, \mathsf{Agg}_x, \mathsf{aux}_x) \overset{?}{=} 1$$
   $$\text{Aggregator}\cdot\textbf{IntegrityVerification}(pp, \{Y_2^{(j)}\}_{j=1}^k, \mathsf{Agg}_y, \mathsf{aux}_y) \overset{?}{=} 1$$
   Otherwise, it aborts.
   (b) Chooses $r_u, r_u', r_u'', r_{(u,x)}, r_{(u,y)} \leftarrow\!\!\$ \ \mathbb{Z}_p^*$
   (c) Computes :
   $$(X^{(I)'}, \pi_x) = \text{Aggregator} \cdot \textbf{WitCreate}(pp, \mathsf{Agg}_x, \mathsf{aux}_x, r_u, r_{(u,x)})$$
   $$(Y^{(I)'}, \pi_y) = \text{Aggregator} \cdot \textbf{WitCreate}(pp, \mathsf{Agg}_y, \mathsf{aux}_y, r_u R_y, r_{(u,y)})$$
   (d) Computes $\sigma' = (h_1' = h_1^{r_u'}, h_2' = h_2^{r_u''}, \sigma_1' = \sigma_1^{r_u r_u'}, \sigma_2' = \sigma_2^{r_u r_u''})$

   The algorithm outputs $(X^{(I)'}, Y_2^{(I)'}, \pi_x, \pi_y, \sigma')$.
6. $\{0,1\} \leftarrow$ **VerifyRandomized**$(pp, \sigma', X^{(I)'}, Y_2^{(I)'}, \pi_x, \pi_y, m, vsk)$:[5] on input of pp, of commitments to an issuer's keys $X^{(I)'}, Y_2^{(I)'}$, of two proofs of set-membership π_x, π_y, of a randomized signature $\sigma' = (h_1', h_2', \sigma_1', \sigma_2')$, of a verifier secret key $vsk = (\mathsf{sk}_x, \mathsf{sk}_y)$, and of a message m, a verifier, outputs 1 if the proof of set-membership is honestly built, i.e. if: Aggregator·**Verify**$(pp, \mathsf{Agg}_x, \mathsf{aux}_\mathsf{x}, \mathsf{sk}_x, X^{(I)'}, \pi_x) \overset{?}{=} 1$, Aggregator · **Verify**$(pp, \mathsf{Agg}_y, \mathsf{aux}_\mathsf{y}, \mathsf{sk}_y, Y_2^{(I)'}, \pi_y) \overset{?}{=} 1$, and the signature is correct, i.e.:
   - $e(h_1', X^{(i)'} Y_2^{(i)'\mathsf{H}(m)}) \overset{?}{=} e(\sigma_1', g_2)$
   - $e(h_2', X^{(i)'} Y_2^{(i)'\mathsf{H}(\mathsf{H}(m))}) \overset{?}{=} e(\sigma_2', g_2)$
   - $h_1' \neq 1_{\mathbb{G}_1}, h_2' \neq 1_{\mathbb{G}_1}, \sigma_1' \neq 1_{\mathbb{G}_1}, \sigma_2' \neq 1_{\mathbb{G}_1}$;

   the verifier outputs 0 otherwise.

Security Analysis

The formal definition of HIAC gives us 4 security notions that our instantiation must achieve: EUF-CMA, unlinkability, issuer indistinguishability, trusted issuer. The trusted-issuer property comes from the collision freedom of the aggregator and is already proven.

Theorem 4. The **VerifyRandomized** algorithm is correct, i.e. any honestly built signature will be accepted by the **VerifyRandomized** algorithm.

Proof sketch 4. The proof, in Appendix E.5, shows that a correctly built signature is equivalent to $e(h_1', X^{(i)'} Y_2^{(i)'\mathsf{H}(m)})$.

Theorem 5. The signature presented here has the EUF-CMA property.

Proof sketch 5. The proof, in Appendix E.6, shows that, under the CDH assumption, the user cannot find a way to inject malicious material into $Y_2'^{(I)}$, σ_1 and σ_2. First, the malicious user would need to know $g_1^{r_{(I,a)} y_I}, a \in \{1,2\}$ to be able to modify the message a credential refers to. However, under the CDH assumption, this value cannot be found by the adversary. Second, the malicious user would need to tweak the commitment to $Y_2^{(I)}$. However, tweaking these values, so that they validate both of the signature's verification equations at the same time would imply inverting a hash function. This shows that the signature has the EUF-CMA property, under the CDH assumption, in the Random Oracle Model.

The random elements added by the user in **Randomize** ensure that the verifier cannot compare the commitment's elements with actual elements of its own key, and the elements of the signature with the formerly issued credentials. This provides issuer indistinguishability.

[5] The user can verify the integrity of the signature – without using **VerifyRandomized** – by verifying that $e(\sigma_1, g_2) \overset{?}{=} e(h_1, X^{(I)}(Y_2^{(I)})^{R_y \mathsf{H}(m)})$, $e(\sigma_2, g_2) \overset{?}{=} e(h_2, X^{(I)}(Y_2^{(I)})^{R_y \mathsf{H}(\mathsf{H}(m))})$, and $h_1 \neq 1_{\mathbb{G}_1}, h_2 \neq 1_{\mathbb{G}_1}$

Theorem 6. The Hidden Issuer Anonymous Credential scheme presented here is *issuer indistinguishable.*
Proof sketch 6. The proof, in Appendix E.7 extends the one for Theorem 3: it compares two randomized signatures issued by two different issuers and shows that the differences between them are hidden by the random elements added by the user in the **Randomize** algorithm. Under the DDH assumption, the malicious verifier has no way to compare the signature it is given with any other signature as all its elements are hidden by a random number that is unknown to the verifier.

6.2 Interactive HIAC

The interactive version of our scheme removes the need for the verifier to run **VerifierSetup** and for the user to run Aggregator·**IntegrityVerification** as part of **Randomize** at each transaction. To achieve this, it relies on an interactive version of the aggregator instantiation.

6.2.1 Aggregator

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 $(W_{(l)})^{x_l} = g_1^{\mathsf{sk}}$. This implies that the user needs to interact with the verifier to randomize a witness, hence the interactive nature of the protocol.

- $(\mathsf{Agg}, \mathsf{aux}, \mathsf{sk}, T_1) \leftarrow$**Gen'**$(pp, \mathcal{S})$: on input of pp and of a set $\mathcal{S} = \{g_1^{\frac{1}{x_1}}, \cdots, g_1^{\frac{1}{x_k}}\}, x_i \in \mathbb{Z}_p, \forall i \in \mathbb{Z}_p^*$, $k \geq 2$, the algorithm computes a secret key $\mathsf{sk} \leftarrow\$ \mathbb{Z}_p$, an aggregator $\mathsf{Agg} = (\mathcal{S}, W)$, with $W = \{(W)_{(j)} = g_1^{\mathsf{sk}(1+\frac{1}{x_j})}\}_{j=1}^{k}$. The algorithm also computes some auxiliary information $\mathsf{aux} = \beta$, with:$\beta = \mathrm{NI-ZKPoE}\{(\mathsf{sk}) : \bigwedge_{i=1}^{k} (W)_{(i)} = ((\mathcal{S})_{(i)} \cdot g_1)^{\mathsf{sk}}\}$ The algorithm finally outputs $(\mathsf{Agg}, \mathsf{aux}, \mathsf{sk})$;
- $(C', \pi_s) \leftarrow$**WitCreate'**$(pp, \mathsf{Agg}, \mathsf{aux}, C, r_1, r_2)$: on input of pp, of an aggregator $\mathsf{Agg} = (\mathcal{S}, W)$, of auxiliary parameters aux, of a commitment $C = g_2^{x_l}, l \in \{1, \cdots, k\}$, and of two random numbers r_1, r_2, the algorithm computes $C' = C^{r_1}$, and a proof $\pi_s = ((W)'_{(l)}, h)$, with $(W)'_{(l)} =$**CommitRevealExchange**$((W)_{(l)})$ and $h = g_2^{r_1 r_2}$; the algorithm outputs (C', π_s). The result of **CommitRevealExchange** is sent to the verifier.

The algorithms use the **CommitRevealExchange** function in Figure C. On input of a witness, the function allows user and verifier to compute a randomized witness interactively. We stress that this does not limit applicability as the presentation of a user's credential to a verifier can generally be turned into an interactive operation. We discuss the efficiency trade-off between interactive and non-interactive variants in Section 8.1

6.2.2 Signature Scheme

The interactive aggregator can be directly used in the signature instantiation by replacing Aggregator.**Gen** and Aggregator.**WitCreate** by Aggregator.**Gen'** and Aggregator.**WitCreate'**. Thanks to these modifications, the verifier only needs to run **VerifierSetup** when its set of trusted issuers changes. Similarly, the user only needs to run Aggregator·**IntegrityVerification** as part of the **Randomize** algorithm when interacting with a given verifier for the first time or when the verifier's aggregators or keys change.

Security Analysis

All we need to show is that the interactive aggregator respects the same properties as the non-interactive one.

Lemma 1. The interactive version of the aggregator is collision-free, correct, and element indistinguishable.
Proof sketch 7. The proof, in appendix E.8, uses the original aggregator proofs presented in Appendix E.3 and Appendix E.4, and shows that the protocol presented in Appendix C does not affect their validity.

7 Deployment

The protocols described in the previous sections fulfill the adversary model, and can be instantiated in real-world use cases. But an Identity Management System does not depend only on its signature scheme as trust between real-life actors cannot be enforced only by algorithms. In this section, we discuss how organizational solutions could improve the security of our scheme by ensuring that its assumptions are satisfied. It is important to note that these solutions are not trivial to implement, and that the privacy impact of an insecure implementation could be significant. A user that trusts the system to protect his/her privacy will tend to leak more information when the system's assumptions are not satisfied than a user who is aware of being in a ”known-issuer” setup.

Credential and Aggregator Management. The formal definition of the issuer-indistinguishability property (Definition 6) assumes that all the verifier's trusted issuers issue credentials on the same types of messages. But in many cases, including in our concert-bar example, the verifier needs to verify multiple credential types. This can easily be solved by having multiple credentials[6] (e.g. one for the ticket, one for the Covid-19 vaccine, and one for the legal age). The user can obtain these credentials either from the same issuer or from different issuers. In either case, each credential will be associated with a specific anonymity set when matched against a verifier's aggregator. To this end, the verifier—the bar in our example—should build a specific aggregator for each credential type. Note that a given issuer may well appear into multiple such aggregators if it issues credentials of multiple types.

Issuer Selection. Clearly, issuer indistinguishability only holds within a specific anonymity set. If the set is too small, or if the set contains issuers that do not issue credentials of the right type, the verifier may be able to associate a credential with a specific issuer.[7] To limit this kinds of attacks, the user's SSI client can integrate a conformity checker. The user should be able to input a desired security policy. The conformity checker can then verify that a verifier's aggregator contains a large enough number of issuers of the right type (based on notions such as l-diversity or t-closeness [20]) that satisfy the policy. The security policy can, for example, specify the geographical distribution of issuers or the organization running the issuing servers.

The checker should be able to access the relevant information for verifying the policy. Most SSI frameworks use a ledger that allows actors to register Decentralized IDentifier (DID) Documents [21]. These documents describe some characteristics of the actors of the framework, and are accessible by every one of them. To this end, the verifier provides the user with the public keys of the issuers aggregated in its aggregator. The conformity checker can verify that these keys correspond to those referenced in the aggregator. It can use them to retrieve the DIDs of the issuers from the ledger and use those to verify the security policy. If the checker detects that a verifier's aggregator does not satisfy the policy, it aborts the transaction and informs the user with a message similar to the one displayed by browsers in the event of an expired HTTPS certificate.[8] If an advanced user understands the associated risks, he can resume the transaction. Furthermore, the checker can detect more elaborated attacks. For example, a verifier could link multiple presentations from the same user via a side channel. In this case, the verifier could adapt its trusted-issuer set, by presenting a set from which it removes one new issuer at each new presentation. Eventually, the removed issuer will be the user's issuer. To detect and counter this attack, the checker can consider the trusted-issuer sets of all the transactions between its owner and the verifier in a specified time window, and check the validity of their intersection rather than the validity of the latest of them. The checker can then warn the user and block the transaction as soon as this intersection becomes too small.

This scheme does not require a central authority, but if one exists, the checker can report misbehaviors to it. For example, legislative power can produce norms involving sanctions if not respected by verifiers. Other central authorities can emerge in decentralized settings. For example, the SSI project Sovrin [4] is decentralized but uses a permissioned blockchain. In this case a central authority consisting of the permissioned nodes of the blockchain can, for instance, revoke the verifying rights of a misbehaving verifier by publishing a revocation list. A variety of other solutions are possible but their discussion is outside the scope of this paper.

Issuer Acting as a Verifier. It is interesting to consider the case in which the verifier is also the issuer of a user's credential. In this case, issuer and verifier are automatically colluding. This makes the identity-inference threats highlighted in the introduction a lot more likely. In this case, Classical AC schemes only provide credential-multi-usage unlinkability. The issuer/verifier can identify whether it was the one that issued a given user's credential. This places the user in a

[6] A technique to link credentials is presented in Appendix B.

[7] We need to emphasize the fact that not only does the verifier need to build a consistent set of trusted issuers, but these issuers must also agree on credential templates and curve parameters.

[8] The design and implementation of messages informing the user about privacy risks is out of the scope of this paper. These messages will have a important role, as they will allow users to understand, and mitigate, privacy risks.

relatively small anonymity set corresponding to all the users that received a credential from the issuer/verifier.

With our solution, the issuer/verifier cannot know whether a given credential originates from itself. This enlarges the user's anonymity set to all the users that received credentials from any of the issuers trusted by the issuer/verifier. Our scheme therefore increases user privacy even in this particularly challenging case.

8 Efficiency

We evaluate the efficiency of our implementation and compare it with that of other signature schemes. We first present a runtime-complexity analysis and compare our scheme with the Pointcheval-Sanders (PS) signature [11]. Then, we compare the asymptotic complexity of our scheme with that of PS and that of the Issuer-Hidden-Attribute-Based-Credential scheme (IHABC) [8]. Finally, we estimate the communication cost of our scheme.

8.1 Runtime Comparison

We start by comparing the runtime cost of our scheme with that of PS, highlighting the overhead resulting from the issuer-indistinguishability property. As both schemes are implemented using the same tools, this is the most accurate manner to identify the overhead of our scheme, regardless of implementation details. We start by comparing the non-interactive and interactive variants. Then we analyze the cost of each operation: issuer-key generation, verifier-key generation, signature issuance, and signature presentation.

We base our comparisons on an implementation in Rust (at: https://gitlab.inria.fr/mgestin/rust_hidden_issuer_signature). We use the BLS12-381 curve [22], a widely used curve for pairing-based cryptography with efficient computation time. BLS is a type-3-pairing-friendly curve with 256-bit-prime group size. This group size gives us a 128-bit AES security level. We ran our experiments on an i7-1185G7, 3.0 GHz CPU, with no multi-processor optimizations.

Non-Interactive vs Interactive Version. Section 6 presented two versions of our scheme: a non-interactive and an interactive one. The non-interactive version requires repeating **VerifierSetup** and **IntegrityVerification** at each credential-presentation transaction, which involves sending two aggregators whose size depends on the number of issuers. As a result, the interactive version becomes more and more interesting as the number of issuers in these aggregators increases. Figure 1 depicts, for a given value of available upload bandwidth, the number of issuers beyond which the interactive version is more efficient than the non-interactive one. For non-first-time interactions (in blue) the interactive protocol turns out to be more efficient as soon as there are more than a few issuers in the aggregator (7 issuers for 40 kbps of upload bandwidth). For first-time interactions (in red), the interactive protocol remains more efficient as long as the upload bandwidth is reasonable albeit for a slightly larger number of issuers (beyond 13 issuers for 40 kbps).

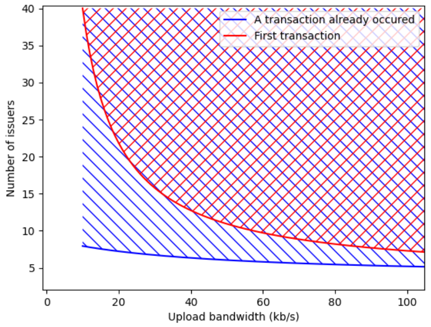

Fig. 1. Number of issuers beyond which the interactive scheme is more efficient than the non-interactive one based on the available bandwidth. The hatched areas represent the area where the interactive protocol is more efficient than the non-interactive one. The red curve and hatching from bottom left to top right consider a first-time interaction: both variants needs to verify the verifier's keys. The blue curve and hatching from top left to bottom right considers a subsequent transaction with the same verifier without a change in the aggregator or key: the interactive scheme does not need to verify the verifier's keys and aggregators as they are already present on the user's device.

In scenarios where the presentation of credentials cannot be done interactively, it is necessary to run **VerifierSetup** and **IntegrityVerification** at each transaction. But even then, our experiments show that the running time of each of these two operations remains under 250 ms with 100 trusted issuers. Moreover, in SSI systems the presentation of credentials is generally an interactive process, so using the interactive protocol appears natural. For this reasons, we concentrate on the interactive protocol in the following.

Issuer-Key Generation. The generation of an issuer key (**IssuerKeyGen**) takes approximately twice as long in our scheme (average: 4.17 ms) as in PS (average: 2.17 ms). This is because our scheme needs to

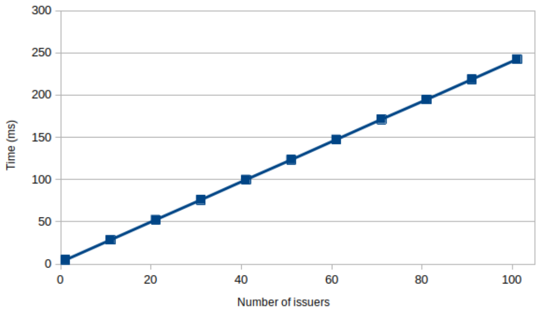

Fig. 2. Average time to build one verifier commitment (**VerifierSetup**) with the HIAC scheme as a function of the number of trusted issuers added to the aggregator.

compute 3 more elements: two to build a verifier key, and one to issue a signature. However, key generation is run only once in a long period of time. Therefore this difference is not a major drawback for our scheme.

Verifier-Key Generation. The verifier-key-generation algorithm is unique to the HIAC scheme. It is run by the verifier after all trusted issuers have created their keys and its performance depends on the number of issuers that are being added to the aggregator. Figure 2 shows that this relationship is linear. Nonetheless, in the interactive version of our scheme, this algorithm only needs to be run when the verifier's secret keys, sk, change. Moreover, since the dependence is on the number of added issuers, this update can actually be very efficient in a number of use cases.

Signature Issuance. The HIAC signature-issuance protocol is equivalent to that of PS. However, our scheme requires issuing the equivalent of two PS signatures, thus taking approximately twice as long to run (average: 9.7 ms) as in PS (average: 4.67 ms) on the issuer's side. In addition, our scheme requires the user to compute 6 exponentiations during the issuance process. This takes 4.91 ms on average on the user's side. In most of the use cases, a user only requests a few credentials, and thus this additional computation time remains negligible.

Signature Presentation. During the presentation of a signature, on the user side, both schemes run a randomization process. In the PS scheme, this randomization has a constant time complexity. With our scheme, there are two different cases. The first time the user interacts with the verifier, he must verify the integrity of the verifier's key. This verification has a linear time complexity. Each new transaction with the same verifier will rely on the first integrity verification, so new transactions only experience a constant time complexity for the user. Both scenarios depend on the available bandwidth to run the interactive protocol. Figure 3 compares these different cases with the PS randomization process.

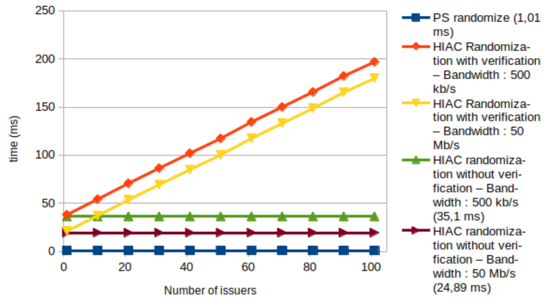

Fig. 3. Comparison of the running time to randomize a PS signature, and a HIAC Signature as a function of the number of issuers.

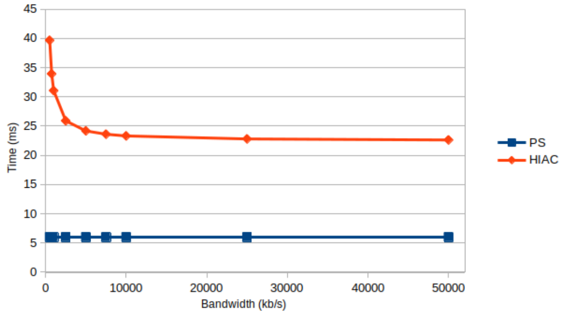

Fig. 4. Time to verify one credential with the PS scheme and the HIAC scheme, as a function of the Bandwidth.

On the verifier side, the verifier needs to compute the **VerifyRandomized** algorithm. The running time of this algorithm depends on network bandwidth. The comparison of the verification times of the PS scheme and of the HIAC scheme is presented in Figure 4. With a bandwidth of at least $2,5$ Mb/s, our verification algorithm is four times slower than that of PS.

General Remarks. The above analysis highlights that while the running time of most of the algorithms is longer than that of an equivalent Anonymous Credential scheme, interactive algorithms still exhibit efficient running times. In particular, the issuance and presentation of a credential only take tenths of milliseconds.

The only concern on the user side lies in the verification of the verifier's key and aggregator, which is linear in the number of trusted issuers, and which must be conducted with each unknown verifier. However, a verification of a subset of the verifier's aggregator may be sufficient if it is enough to fulfill the requirements of the conformity checker discussed in Section 7.

From the point of view of issuers and verifiers, the overhead with respect to the PS scheme is lower (at most 5 times less efficient with a 2500 kb/s bandwidth). Thus the practicality of our scheme will depend on usage. If the verifier (respectively, issuer) needs to verify (respectively, issue) as many credentials per second as possible, the overhead becomes significant, but we expect our scheme to be used mostly in interactions that involve humans, which are less sensitive to slightly higher latency. Furthermore, in an SSI, the overall throughput in terms of transactions per second increases with the number of verifiers, unlike in centralized or blockchain-based services. In particular, with a bandwidth of 2500 kb/s, each new verifier added to the network adds a capacity of 38 transaction per second to the infrastructure. A service provider can thus easily add a server to augment its verification capacity.

8.2 Asymptotic Comparison

Table 1[9] presents an asymptotic comparison of our signature scheme, with two other schemes. As mentioned, PS constitutes the basis which our scheme builds upon. The differences between our scheme and this building block highlight the real cost of issuer-indistinguishability. The recent Issuer-Hidden Attribute-Based Credential (IHABC) [8] achieves issuer-indistinguishability in a trusted-setup setting, using bilinear pairings, and Groth [23] signatures.

Table 1 compares the number of exponentiations and pairings required by our scheme, PS, and IHABC, for the various operations. Table 7 in Appendix D shows the same data after converting the running time $\mathbb{G}_2$ operations and pairings into that of $\mathbb{G}_1$ operations.

The comparison with PS confirms the observations we made in Section 8.1 with respect to our implementation. The one with IHABC depends on whether we analyze the user, the verifer, or the issuer. We analyze the data with reference to Appendix D. On the user's side, the most frequent operations is **Randomize**. Albeit short, its running time is 15% faster for our scheme for 10 trusted issuers. Even if less frequent, the **IntegrityVerification** operation runs 3 times faster in our scheme than in IHABC.

On the verifier's side, the most frequent operation is **VerifyRandomized**. The IHABC implementation of this algorithm is 20% more efficient. In the interactive version we are considering, **VerifierSetup** is executed relatively rarely but it runs 4 times faster in our scheme than in IHABC. Finally, on the issuer's side, **Sign** runs twice as fast in IHABC as in our scheme, while **IssuerKeyGen** runs four times faster in IHABC. However, both operations are very quick to execute, so the difference remains very low in absolute value. Moreover, **IssuerKeyGen** is run only when the issuer starts or changes its keys. To conclude, our scheme offers better performance for the user, and comparable or slightly worse performance for verifiers and issuers, but without requiring a trusted setup.

8.3 Communication Cost

Finally, we study the communication cost of our approach. Table 2 presents the results based on the asymptotic evaluation of the size of the data structures presented in Appendix D.2, expressed using the BLS12-381 group elements' size. The most expensive communication is related to the integrity-verification operation, in which the verifier needs to transfer its aggregators for verification by the user's device. Yet, the table shows that with 100 verifiers, this cost totals to only 25 kB. All other operations exhibit very low communication costs of less than 1 kB per transaction.

9 Qualitative Comparison

We now consider the security properties of HIAC in comparison with those of other schemes as summarized in Table 3. The first property is issuer indistinguishability (hidden issuer). Apart from our scheme, only one other achieves it, Issuer Hidden Attribute Based Credential [8], but using a trusted setup. The second property is the non-transferability of credentials. Our scheme achieves it with the transformation presented in Appendix B. Camenisch and Lysyanskaya (CL01) [24] also propose a signature scheme with such a property. The third property to analyze is the ability to sign a committed message (One-show). Our scheme achieves it in the same way as the PS scheme. Numerous signature schemes, like CL01 or CL04, also have this property.

The fourth property relates to the fact that most AC signature schemes offer a way to prove the possession of

[9] The BLS12-381 implementation we use gives us: 1 exponentiation in $\mathbb{G}_2 \approx 2$ exponentiation in $\mathbb{G}_1$ and 1 pairing operation ≈ 1.12 exponentiation in $\mathbb{G}_2$.

	HIAC	IHABC	PS
IssuerKeyGen	$3\mathbb{G}_1 + 2\mathbb{G}_2$	$1\mathbb{G}_2$	$2\mathbb{G}_2$
Sign (User Side)	$6\mathbb{G}_1$	$\emptyset$	$\emptyset$
Sign (Issuer Side)	$10\mathbb{G}_1$	$4\mathbb{G}_1 + 1\mathbb{G}_2$	$3\mathbb{G}_1$
VerifierSetup	$4k\mathbb{G}_1$	$(1+k)\mathbb{G}_1 + 4k\mathbb{G}_2$	$\emptyset$
IntegrityVerification	$4k\mathbb{G}_1$	$6k$ Pairing	$\emptyset$
Randomize	$26\mathbb{G}_1 + 2\mathbb{G}_2$	$10\mathbb{G}_1 + 6\mathbb{G}_2 + 6$ Pairing	$2\mathbb{G}_1$
VerifyRandomize	$20\mathbb{G}_1 + 2\mathbb{G}_2 + 8$ **Pairing**	$4\mathbb{G}_1 + 2\mathbb{G}_2 + 12$ Pairing	$1\mathbb{G}_2 + 2$ Pairing

Table 1. Comparison of the number of exponentiations in $\mathbb{G}_1$, $\mathbb{G}_2$ and number of pairings required to run our scheme (HIAC), the Issuer Hidden Attribute Based Credential [8] (IHABC) without ZKP of knowledge signature, and the Pointcheval Sanders Signature Scheme [11] (PS), with k being the number of trusted issuers.

	Issuer	User	Verifier
Sign	1536	768	$\emptyset$
VerifierSetup	1536	$\emptyset$	$\emptyset$
IntegrityVerification	$\emptyset$	$\emptyset$	$2k \times 256 + 4k \times 384$
Credential Presentation	$\emptyset$	7680	1536

Table 2. Communication costs of each operation of the interactive HIAC scheme, for each issuer, user, and verifier, expressed in bits, with k being the number of trusted issuers. Only upload costs are considerated, round trip times are not taken into account. The Credential Presentation operation takes into account the transmission of the user's randomized credential and the commitment reveal exchange.

	HIAC	CL01	CL04	PS	IHABC
Hidden issuer	✓				✓
Non transferable	✓	✓			
One-show	✓	✓	✓		
ZKP of signature		✓	✓	✓	✓
Revocation			✓		✓

Table 3. Qualitative comparison of different Anonymous Credential scheme. (HIAC: Hidden Issuer Anonymous Credential; CL01: hidden size group based Anonymous Credential scheme [24] ; CL04: bilinear pairing based Anonymous Credential scheme [25]; PS: Short randomizable Anonymous Credential scheme [11]; IHABC: Issuer Hidden Attribute Based Credential [8])

a credential without revealing the associated message. Our scheme could implement this by adapting the zero-knowledge proof of a signature used by PS. However, the prover would need to prove a relationship between a message and its hash. Whereas this can be proven in zero knowledge, any implementation would be inefficient, because hash functions are based on boolean circuits, for which zero knowledge proofs are known to be inefficient. To achieve this property efficiently, we could replace the hash function with a low-degree polynomial fulfilling some conditions. We leave this as future work. Another way to achieve this property would be to use Groth's signature like IHABC, but this would come at the cost of a trusted setup as discussed in Section 10.

The fifth and last property often implemented by AC schemes is credential revocation. Designing a revocation mechanism while keeping issuer indistinguishability is non trivial. The most efficient way to achieve it in our scheme would be for the issuers to regularly publish a revocation accumulator, accumulating all revoked credentials. Users would later leverage this accumulator to prove that their credentials do not appear in the revocation list, as suggested by Camenisch et al. [26]. However, this can be rather complicated to achieve in a hidden-issuer context, and should be the subject of further research.

10 Related Work

Anonymous Credentials (AC) have been a well studied topic since they were introduced by Chaum [6] in 1985. This first work defines them as signatures certifying that an element was issued by a given entity, while ensuring unlinkability between each use of this signature by a user. The first efficient implementation is due to Camenisch and Lysyanskaya in 2001 [24]. This work offers a cryptographic basis to build an efficiently computable signature, with the properties listed by Chaum. It also adds properties, in particular non transferability. Several papers enhanced this first implementation of Chaum's principles with more efficient schemes, including one from the same authors, in 2002 [27]. The first two ACs we cited above are based on RSA groups. Later evolution made it possible to work on elliptic-curve-based groups with bilinear pairings. These include the 2004 Camenisch and Lysyanskaya article [25], and the 2016 Pointcheval Sanders (PS) Short Randomizable

Signature [11]. The latter raised our interest because, it contains few elements, and it does not contain trapdoors. Even though the AC schemes presented here require a third-party issuer, it is important to notice that, in some particular cases, such issuers are not needed. Decentralized anonymous credentials [28] propose an efficient scheme for cases such as the mitigation of sybil attacks or Direct Anonymous Attestation.

Other interesting signatures have been developed in the meantime, in particular, ring signatures [29], and ad hoc group signatures [30]. These signatures do not disclose the identity of the issuer of a credential. This property comes from the fact that the issuer is able to hide among a group of other potential issuers. To create such a signature, the issuer chooses a set of issuers. It signs a message using these issuers' public keys and its own secret key. It is then computationally impossible to find the original issuer. However, only the issuer can create such a ring, which makes ring signatures impractical for the user, who wants to be able to use his credential with numerous different verifiers, who do not trust the same set of issuers. An answer to this problem can be found in oblivious signatures [31], and more precisely in the Universal ring Signature (US) scheme [32]. The latter makes it possible for a user to recreate a ring signature from a simple signature, without knowledge of any secret key. Although this signature seems to meet our issuer indistinguishability requirement, it presents major drawbacks with respect to HIAC. First, it does not provide unlinkability: the signature in the US scheme is given to the verifier without randomization. Secondly, the US scheme does not offer collusion resistance: a verifier colluding with the issuer of a given credential can directly find the real issuer inside the produced ring, by making a comparison between the elements in the ring and the previously signed elements. Finally, the US scheme always exhibits linear complexity both in verification and in the ring-generation process that effectively hides the identity of the issuer. On the other hand, HIAC always runs verification in constant time, and only exhibits linear-time complexity for hiding the public key of the issuer when a user interacts with a given verifier for the first time.

During the writing of this paper, a new AC scheme was proposed by Bobolz et al. [8]. Their paper also offers an AC scheme with issuer indistinguishability. As shown in Table 3, their scheme provides anonymity and credential revocation by means of a revocation authority. However, it uses an aggregator-like construct that is less efficient than our own (**VerifierSetup** and **IntegrityVerification** lines of Table 1) and it employs Groth's signature scheme [23], which requires a trusted setup—the value Y in the signature.

The use of Groth's signature provides nonetheless an advantage over the implementation proposed in this paper, it enables efficient Zero Knowledge Proofs of Knowledge of signatures. Because both our solution and theirs use bilinear pairings, and because our aggregator is more efficient, but their signature offers more properties, it seems interesting to explore the perspective of a HIAC scheme built using their signature scheme and our aggregator.

Our new aggregator primitive makes it possible to prove the set-membership of a value, without revealing it, and while knowing only a commitment to it. The state of the art in the domain of set-membership uses two different approaches. The first one is based on cryptographic accumulators [33–37]. These objects enable the aggregation of multiple elements in one object and the proof that a given element was indeed accumulated. The second way is more direct, without accumulators [38, 39]. However, all these approaches require the prover to know the value it proves set-membership of. Our aggregator removes this need: the prover only needs to know a commitment to the value.

11 Conclusion

We provided a formal definition of an Anonymous Credential scheme that hides the issuer of a credential inside the set of issuers trusted by a verifier. We gave an instantiation of this scheme, based on a new primitive called aggregator, and a modified Pointcheval Sanders signature scheme. This new Hidden Issuer Anonymous Credential (HIAC) enhances the minimization principle of SSIs, and it improves the collusion resistance of Anonymous Credential schemes. It achieves EUF-CMA, unlinkability, issuer indistinguishability, and the trusted-issuer property; and it does not require a trusted-setup. The aggregator primitive can be used in other issuer-indistinguishable signature schemes, and its instantiation is interoperable with state-of-the-art signatures.

Our work opens new research topics. First, we want to explore the use of low-degree polynomials to make it practical to prove the possession of a credential without revealing the associated message. Second, an interesting but complex task that must be addressed before applying our scheme in SSIs consists in identifying how to best address the several deployment challenges we discussed in Section 7.

Acknowledgments

We wish to thank the anonymous reviewers and our shepherd, Christina Garman, for their insightful comments and remarks that led to significant improvements to our paper. This work was partially funded by the SOTERIA project. SOTERIA has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101018342. This content reflects only the author's view. The European Agency is not responsible for any use that may be made of the information it contains.

References

[1] Alexander Mühle, Andreas Grüner, Tatiana Gayvoronskaya, and Christoph Meinel. A survey on essential components of a self-sovereign identity. *Computer Science Review*, 30:80–86, 2018.

[2] C. Allen. The path to self sovereign identity, April 2016. [Online], (http://www.lifewithalacrity.com/2016/04/the-path-to-self-sovereign-identity.html).

[3] Zhiyi Zhang, Michał Król, Alberto Sonnino, Lixia Zhang, and Etienne Rivière. El passo: Efficient and lightweight privacy-preserving single sign on. In *Proceedings on Privacy Enhancing Technologies*, volume 2, pages 70–87, 2021.

[4] Sovrin: A protocol and token for self-sovereign identity and decentralized trust. Technical report, 01 2018.

[5] D. Reed A. Tobin. The inevitable rise of self sovereign identity. Technical report, 09 2017.

[6] David Chaum. Security without identification: Transaction systems to make big brother obsolete. *Commun. ACM*, 28(10):1030–1044, October 1985.

[7] Md. Sadek Ferdous, Gethin Norman, and Ron Poet. Mathematical modelling of identity, identity management and other related topics. In *Proceedings of the 7th International Conference on Security of Information and Networks*, SIN '14, page 9–16, New York, NY, USA, 2014. Association for Computing Machinery.

[8] Jan Bobolz, Fabian Eidens, Stephan Krenn, Sebastian Ramacher, and Kai Samelin. Issuer-hiding attribute-based credentials. In Mauro Conti, Marc Stevens, and Stephan Krenn, editors, *Cryptology and Network Security*, pages 158–178, Cham, 2021. Springer International Publishing.

[9] Ewa Syta, Philipp Jovanovic, Eleftherios Kokoris Kogias, Nicolas Gailly, Linus Gasser, Ismail Khoffi, Michael J. Fischer, and Bryan Ford. Scalable bias-resistant distributed randomness. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 444–460, 2017.

[10] Serguei Popov. On a decentralized trustless pseudo-random number generation algorithm. *Journal of Mathematical Cryptology*, 11(1):37–43, 2017.

[11] David Pointcheval and Olivier Sanders. Short randomizable signatures. In Kazue Sako, editor, *Topics in Cryptology - CT-RSA 2016*, pages 111–126, Cham, 2016. Springer International Publishing.

[12] S. Galbraith. *Mathematics of Public Key Cryptography. Version 2.0*. October 2018.

[13] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen message attack. In David S. Johnson, Takao Nishizeki, Akihiro Nozaki, and Herbert S. Wilf, editors, *Discrete Algorithms and Complexity*, pages 287 – 310. Academic Press, 1987.

[14] Steven D. Galbraith, Kenneth G. Paterson, and Nigel P. Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, 156(16):3113 – 3121, 2008. Applications of Algebra to Cryptography.

[15] Jan Camenisch and Markus Stadler. Efficient group signature schemes for large groups. In Burton S. Kaliski, editor, *Advances in Cryptology — CRYPTO '97*, pages 410–424, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg.

[16] David Chaum and Torben Pryds Pedersen. Wallet databases with observers. In Ernest F. Brickell, editor, *Advances in Cryptology — CRYPTO' 92*, pages 89–105, Berlin, Heidelberg, 1993. Springer Berlin Heidelberg.

[17] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology — CRYPTO' 86*, pages 186–194, Berlin, Heidelberg, 1987. Springer Berlin Heidelberg.

[18] David Bernhard, Olivier Pereira, and Bogdan Warinschi. How not to prove yourself: Pitfalls of the fiat-shamir heuristic and applications to helios. In Xiaoyun Wang and Kazue Sako, editors, *Advances in Cryptology – ASIACRYPT 2012*, pages 626–643, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[19] Benger, Bernhard, Catalano, Charlemagne, Conti, Cubaleska, Fernando, Fiore, Galbraith, Galindo, Hermans, Iovino Vincenzo, Jager, Kohlweiss, Libert, Lindner, Loehr, Lynch, Moloney, Ouafi, Pinkas, Polach, Di Raimondo, Rückert, Schneider, Singh, Smart, Stam, Vercauteren, Villar Santos, and Williams. Final report on main computational assumptions in cryptography. Technical report, 01 2013.

[20] Charu C. Aggarwal and Philip S. Yu. A general survey of privacy-preserving data mining models and algorithms. In *Privacy-Preserving Data Mining*, 2008.

[21] W3C Decentralized Identifier Working Group. Decentralized identifiers (dids) v1.0 vore architecture, data model, and representation. Technical report, August 2021. Online. Accessed 24 February 2022.

[22] Paulo S. L. M. Barreto, Ben Lynn, and Michael Scott. Constructing elliptic curves with prescribed embedding degrees. In Stelvio Cimato, Giuseppe Persiano, and Clemente Galdi, editors, *Security in Communication Networks*, pages 257–267, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[23] Jens Groth. Efficient fully structure-preserving signatures for large messages. In Tetsu Iwata and Jung Hee Cheon, editors, *Advances in Cryptology – ASIACRYPT 2015*, pages 239–259, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[24] Jan Camenisch and Anna Lysyanskaya. A signature scheme with efficient protocols. In Stelvio Cimato, Giuseppe Persiano, and Clemente Galdi, editors, *Security in Communication Networks*, pages 268–289, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[25] Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In Matt Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, pages 56–72, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[26] Jan Camenisch, Markulf Kohlweiss, and Claudio Soriente. An accumulator based on bilinear maps and efficient revocation for anonymous credentials. In Stanisław Jarecki and Gene Tsudik, editors, *Public Key Cryptography – PKC 2009*, pages 481–500, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[27] Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In Birgit Pfitzmann, editor, *Advances in Cryptology — EUROCRYPT 2001*, pages 93–118, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[28] Christina Garman, Matthew Green, and Ian Miers. Decentralized anonymous credentials. Cryptology ePrint Archive, Report 2013/622, 2013. https://ia.cr/2013/622.

[29] Ronald L. Rivest, Adi Shamir, and Yael Tauman. How to leak a secret. In Colin Boyd, editor, *Advances in Cryptology — ASIACRYPT 2001*, pages 552–565, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[30] Yevgeniy Dodis, Aggelos Kiayias, Antonio Nicolosi, and Victor Shoup. Anonymous identification in ad hoc groups. In Christian Cachin and Jan L. Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004*, pages 609–626, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[31] Lidong Chen. Oblivious signatures. In Dieter Gollmann, editor, *Computer Security — ESORICS 94*, pages 161–172, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg.

[32] Raylin Tso. A new way to generate a ring: Universal ring signature. *Computers and Mathematics with Applications*, 65(9):1350 – 1359, 2013. Advanced Information Security.

[33] Josh Benaloh and Michael de Mare. One-way accumulators: A decentralized alternative to digital signatures. In Tor Helleseth, editor, *Advances in Cryptology — EUROCRYPT '93*, pages 274–285, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg.

[34] Niko Barić and Birgit Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In Walter Fumy, editor, *Advances in Cryptology — EUROCRYPT '97*, pages 480–494, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg.

[35] Jan Camenisch and Anna Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In Moti Yung, editor, *Advances in Cryptology — CRYPTO 2002*, pages 61–76, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[36] Lan Nguyen. Accumulators from bilinear pairings and applications. In Alfred Menezes, editor, *Topics in Cryptology – CT-RSA 2005*, pages 275–292, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[37] Daniel Benarroch, Matteo Campanelli, Dario Fiore, and Dimitris Kolonelos. Zero-knowledge proofs for set membership: Efficient, succinct, modular. *IACR Cryptol. ePrint Arch.*, 2019:1255, 2019.

[38] Eduardo Morais, Cees van Wijk, and Tommy Koens. Zero knowledge set membership. 2018.

[39] Jan Camenisch, Rafik Chaabouni, and abhi shelat. Efficient protocols for set membership and range proofs. In Josef Pieprzyk, editor, *Advances in Cryptology - ASIACRYPT 2008*, pages 234–252, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[40] Anna Lysyanskaya, Ronald L. Rivest, Amit Sahai, and Stefan Wolf. Pseudonym systems. In Howard Heys and Carlisle Adams, editors, *Selected Areas in Cryptography*, pages 184–199, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

[41] Torben Pryds Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Joan Feigenbaum, editor, *Advances in Cryptology — CRYPTO '91*, pages 129–140, Berlin, Heidelberg, 1992. Springer Berlin Heidelberg.

A Notations

We give a summary of the notations used in the aggregator scheme, and the hIAC scheme.

A.1 General Notations

A general notation table is given in Table 4.

A.2 Aggregator Notations

A summary of the notations used for the aggregator is given in Table 5.

A.3 Hidden Issuer Anonymous Credential Notations

A summary of the notations used in the HIAC scheme is given in Table 6.

B Possible Additional Properties

We presented the general hidden issuer Anonymous Credential scheme in 6.1.2. in this section we will present alternative properties we can achieve with the same signature, by modifying it slightly.

B.1 Non Transferable Signature

It can be useful for the verifier to ensure that a given credential was actually issued to the user who is using

Notation	Correspondence	Set
$\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$	Three groups of prime order p linked by a bilinear pairing e, such that $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$	
g_1, g_2, g_T	Generator of the groups $\mathbb{G}_1, \mathbb{G}_2$, and $\mathbb{G}_T$ respectively	
p	Prime order of the groups $\mathbb{G}_1, \mathbb{G}_2$, and $\mathbb{G}_T$	$\mathbb{P}$
e	Bilinear pairing	$e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$
H	Collision-free one-way hash function	$\mathsf{H} : \mathbb{Z} \rightarrow \{0, \cdots, k\}$

Table 4. General notations

Notation	Correspondence	Value	Set
x_i	Secret values aggregated in the aggregator		$\mathbb{Z}_p^*$
$\mathcal{S}$	Set of values aggregated in the aggregator	$\{g_1^{\frac{1}{x_1}}, \cdots, g_1^{\frac{1}{x_k}}\}$	$(\mathbb{Z}_p^*)^k$
C	Commitment to one element $s \in \mathcal{S}$	$g_2^{x_s}$	$\mathbb{G}_2$
sk	Secret set-membership verifcation key		$\mathbb{Z}_p^*$
W	Aggregator set-membership set of witnesses	$\{g_1^{\mathsf{sk}\frac{1}{x_j}}\}_{j=1}^k$	$\mathbb{G}_1$
$(W)_{(j)}$	j-th element of W	$g_1^{\mathsf{sk}\frac{1}{x_j}}$	$\mathbb{G}_1$
β	Aggregator integrity verification proof	NI-ZKPoE$\{(\mathsf{sk}) :$ $\bigwedge_{j=1}^k (W)_{(j)} = (W_p)_{(j)}^{\mathsf{sk}}\}$	
r_1, r_2	Two randomizing elements used by the user to hide the values he commits to		$\mathbb{Z}_p^*$
C'	Randomized commitment to one element $s \in \mathcal{S}$	$g_2^{x_s r_1}$	$\mathbb{G}_2$
π_s	Proof that C' belongs to the set $\mathcal{S}_{\mathsf{comm}}$	$((W)'_{(l)} = (W)_{(l)}^{r_2}, h = g_1^{r_1 r_2})$	$(\mathbb{G}_1, \mathbb{G}_2)$

Table 5. Aggregator notations

it, and not to someone else. Because of the unlinkability property, this cannot be straightforward. For example, the easiest way to obtain this property consists in adding an element in the message that is linked to its user. However, this would break the unlinkability property. We offer a way to make our signature non transferable without this drawback.

The idea consists in discouraging transfers by relying on an external escrow where the user stores some valuable item, or money. Sharing a credential would also imply sharing the valuable in the escrow. This is called *PKI-assured non-transferability* [40].

We propose to adapt this property to our scheme. The user will put something valuable inside an escrow, and lock it in place with a private key. The user then uses this private key instead of the random value R_y in step 4 (**Sign** algorithm). He then proves to the issuer, that the value he gives to it (which in our protocol was named $u = Y_1^{R_y} g_1^R$) is a commitment to a key that can open the escrow. The value of this escrow will depend on the requirements of the issuer. Afterward, the process is the same, except for the protocol **VerifyRandomized**. We add an interactive ZKP verification to the original algorithm:

$$\text{ZKPoK}\{(R_y r_u r_{(u,y)}) : h_y = g_2^{R_y r_u r_{(u,y)}}\}$$

This allows the verifier to ensure that the user knows the blinding elements, in particular the non-transferable element R_y. If all the issuers follow the requirements of the verifier, then the credentials are non transferable. This technique can also be used to link credentials between themselves. A verifier can verify that different credentials embed the same R_y's values. It proves that they were issued to the same user.

This modification can be added to the PS signature scheme to enable non-transferability property.

B.2 Signature on Commitments and One-Show Credential

An interesting feature to have in a credential scheme is optional one-show property. In this configuration, a credential can only be used once. It can be particularly useful in e-vote systems for example.

To enhance this property, the easiest way is for the issuer to sign committed message. If the issuer's key is used for this purpose only, the message can represent a nonce, that will identify the credential. Before showing the credential, the user will un-commit the message, while keeping the signature structure. Thus, the verifier

Notation	Correspondence	Value	Set
isk_i	i-th issuer secret key	(x_i, y_i)	$(\mathbb{Z}_p^*)^2$
ipk_i	i-th issuer public key	$(X^{(i)} = g_2^{x_i}, \bar{X}_1^{(i)} = g_1^{\frac{1}{x_i}}, Y_1^{(i)} = g_1^{y_i},$ $\bar{Y}_1^{(i)} = g_1^{\frac{1}{y_i}}, Y_2^{(i)} = g_2^{y_i})$	$(\mathbb{G}_2 \times \mathbb{G}_1$ $\times \mathbb{G}_1 \times \mathbb{G}_1$ $\times \mathbb{G}_2)$
m	Message certified by a credential		$\mathbb{Z}_p^*$
R_y	User's random secret element used to enhance issuer indistinguishability property of a credential. Used with only one credential.		$\mathbb{Z}_p^*$
R	User's secret random element used to hide R_y in the issuance protocol		$\mathbb{Z}_p^*$
u_1	User commitment to the i-th issuer key $Y_1^{(i)}$	$(Y_1^{(i)})^{R_y} g_1^R$	$\mathbb{G}_1$
u_2	User commitment to the i-th issuer key $Y_1^{(i)}$	$(Y_2^{(i)})^{R_y}$	$\mathbb{G}_2$
ϕ	Proof of knowledge of R_y and R in u_1	$\mathsf{NI\text{-}ZKPoK}\{(R_y, R) :$ $u_1 = (Y_1^{(i)})^{R_y} g_1^R\}$	
$(r_{(I,1)}, r_{(I,2)})$	I-th issuer's random secret elements used to enhance the security of the **VerifyRandomized** algorithm. Used only once.		$(\mathbb{Z}_p^*)^2$
σ^Δ	Outputted value by the issuer I after running the **Sign** algorithm. Depends on R.	$(h_1 = g_1^{r_{(I,1)}}, h_2 = g_1^{r_{(I,2)}},$ $\sigma_1^\Delta = g_1^{r_{(I,1)}(x_I + \mathsf{H}(m)(y_I + R))},$ $\sigma_2^\Delta = g_1^{r_{(I,2)}(x_I + \mathsf{H}(\mathsf{H}(m))(y_I + R))})$	$(\mathbb{G}_1)^4$
σ	User final credential. Does not depend on R.	$(h_1 = g_1^{r_{(I,1)}}, h_2 = g_1^{r_{(I,2)}},$ $\sigma_1 = g_1^{r_{(I,1)}(x_I + \mathsf{H}(m) y_I)},$ $\sigma_2^\Delta = g_1^{r_{(I,2)}(x_I + \mathsf{H}(\mathsf{H}(m)) y_I)})$	$(\mathbb{G}_1)^4$
Agg_x	Verifier's aggregator referring to the x values of the issuer's secret keys		
Agg_y	Verifier's aggregator referring to the y values of the issuer's secret keys		
aux_x	Verifier's aggregator's auxiliary values associated to Agg_x		
aux_y	Verifier's aggregator's auxiliary values associated to Agg_y		

Table 6. Hidden issuer Anonymous Credential scheme notations

will see the nonce, and will be able to add it to a shared ledger. Any message with a nonce already added to the ledger is thus revoked. With this protocol, the user gets one-show property, without loosing any other security properties.

To adapt this principle to our scheme, we use a modified PS signature on committed message protocol. In fact, to achieve this property we only modify the **Sign** algorithm:

Sign(pp, isk_I, u, u', ϕ): on input of a setup parameter pp,, an issuer's secret key $isk_I = (x_I, y_I)$, two values given by the user $u = (Y_1^{(I)})^{\mathsf{H}(m) R_y} \cdot g_1^R$ and $u' = (Y_1^{(I)})^{\mathsf{H}(\mathsf{H}(m)) R_y} \cdot g_1^R$, where R_y and R are secret random values, and m is the message, and ϕ a ZKPoK of u, $\phi : NI - ZKPoK\{(\mathsf{H}(m)R_y, R) : u = Y_1^{(I)\mathsf{H}(m)R_y} \cdot g_1^R \wedge u' = Y_1^{(I)\mathsf{H}(\mathsf{H}(m))R_y} \cdot g_1^R\}$. Initially, the issuer outputs a signature $\sigma^\Delta = (h_1, h_2, \sigma_1^\Delta, \sigma_2^\Delta)$, where $h_1 = g_1^{r_{(I,1)}}$, $h_2 = g_1^{r_{(I,2)}}$, $\sigma_1^\Delta = (g_1^{x_I} \cdot u)^{r_{(I,1)}}$, $\sigma_2^\Delta = (g_1^{x_I} \cdot u')^{r_{(I,2)}}$, with $r_{(I,1)}, r_{(I,2)} \leftarrow\$ \mathbb{Z}_p^*$. The user then uncommits the signature σ^Δ into $\sigma = (h_1, h_2, \sigma_1 = \sigma_1^\Delta \cdot h_1^{-R}, \sigma_2 = \sigma_2^\Delta \cdot h_2^{-R})$. The user stores σ and R_y.

With this transformation, the other parts of the protocol are not modified, and the issuer does not learn the message signed.

C Commitment Reveal Exchange Implementation

Figure 5 presents an example of a three move protocol that allows a user to commit to a verifier's aggregator, while none of the two actors learn unnecessary information.

Commitment reveal exchange

$\mathbf{User}(pp, \sigma, \mathsf{pk})$		$\mathbf{Verifier}(pp, \mathsf{sk})$
$r_a, r_b, r_c \leftarrow\!\$\ \mathbb{Z}_p^*$; $(\mathsf{Comm}'(s), \pi_s, r_2) = \mathbf{Randomize}(\sigma, \mathsf{pk})$		$r_d \leftarrow\!\$\ \mathbb{Z}_p^*$
$(W)''_I = (W)_I^{r_2} g_1^{r_a}$; $C_1 = g_1^{r_2 r_b}$; $C_2 = g_1^{r_a r_c}$		
$\phi_1 = NI - ZKPoK\{(r_2 r_b, r_a r_c) : C_1 = g_1^{r_2 r_b} \wedge C_2 = g_1^{r_a r_c}\}$	$\xrightarrow{(\sigma', (W)''_I, C_1, C_2, \phi_1)}$	$\mathbf{VerifyZKP}(\phi_1)$
		$C'_1 = C_1^{\mathsf{sk} r_d}; C'_2 = C_2^{r_d}$
		$\phi_2 = NI - ZKPoK\{(\mathsf{sk}, r_d) :$
$\mathbf{VerifyZKP}(\phi_2)$	$\xleftarrow{(C'_1, C'_2, \phi_2)}$	$C'_1 = C_1^{\mathsf{sk} r_d} \wedge C'_2 = C_2^{r_d}\}$
$C''_1 = (C'_1)^{\frac{1}{r_b}} (C'_2)^{\frac{1}{r_c}}$	$\xrightarrow{C''_1}$	$(W)'_I = (W)''_I (C''_1)^{-\frac{1}{r_d}}$

Fig. 5. Commitment reveal exchange

D Asymptotic Data Expressed in BL12-381 Setup

In this section, we express the asymptotic data we produced in section Section 8 with actual the actual implementation cost of the BLS12-381 setup. Then we

D.1 Asymptotic Efficiency

Table 7 expresses the asymptotic efficiency of our scheme, and the IHABC scheme in $\mathbb{G}_1$ equivalent operation, based on the approximation : 1 exponentiation in $\mathbb{G}_2 \approx 2$ exponentiation in $\mathbb{G}_1$ and 1 paring operation ≈ 1.12 exponentiation in $\mathbb{G}_2$.

	HIAC	IHABC
IssuerKeyGen	**9**	**2**
Sign (Issuer Side)	**10**	**6**
VerifierSetup	**40**	**90**
IntegrityVerification	**40**	**134.4**
Randomize	**30**	**35.44**
VerifyRandomize	**41.92**	**34.88**

Table 7. Comparison of the number of operations to run our scheme (HIAC), the Issuer Hidden Attribute Based Credential [8] (IHABC) without ZKP of signature. Operations in $\mathbb{G}_2$ and pairings operations are converted to $\mathbb{G}_1$. With 10 trusted issuers.

D.2 Size of the Data Structures

The size of the data structures of our scheme are given in Table 8. With the BLS12-381 configuration[10], the elements of our signature scheme are relatively small. Except from the verifier key, they are comparable to a high or very-high strength RSA key (2048 - 4096 bits). The only large element to transfer is the verifier key, which contains multiple issuer keys, and the associated aggregator. However, if this key is static, users will save it in their device, and will only need to request it to a the verifier once. The size of the elements of the IHABC scheme are equivalent in most of the cases, except for the issuer public key, which is smaller in their case. Table 9 completes the picture by expressing the size of the elements of each signature scheme in bits, using the fact that, in BLS12-381 configuration, the size of the element are : $\mathbb{Z}_p$: 256 bits, $\mathbb{G}_1$: 384 bits, and $\mathbb{G}_2$: 768 bits. Table 10 uses the data from previous tables to estimate the communication costs of operations. This table is the source for Table 2 in the paper.

E Proofs

E.1 Assumptions

Our signature scheme relies on several complexity assumptions:

Definition 15. [Discrete Logarithm (DL) assumption] Given $h \in \mathbb{G}$, with $\mathbb{G}$ a group generated by g, no PPT ad-

[10] In the BLS12-381 configuration, the size of the element are : $\mathbb{Z}_p$: 256 bits, $\mathbb{G}_1$: 384 bits, and $\mathbb{G}_2$: 768 bits.

	HIAC	IHABC	PS
Issuer public key	$\mathbf{3\mathbb{G}_1 + 2\mathbb{G}_2}$	$\mathbf{1\mathbb{G}_2}$	$\mathbf{2\mathbb{G}_2}$
Issuer secret key	$\mathbf{2\mathbb{Z}_p}$	$\mathbf{1\mathbb{Z}_p}$	$\mathbf{2\mathbb{Z}_p}$
Verifier public key	$\mathbf{2k\mathbb{Z}_k + 4k\mathbb{G}_1}$	$\mathbf{(k+1)\mathbb{G}_1}$ $\mathbf{+3k\mathbb{G}_2}$	$\emptyset$
Verifier secret key	$\mathbf{2\mathbb{Z}_p}$	$\mathbf{1\mathbb{Z}_p}$	$\emptyset$
Signature	$\mathbf{1\mathbb{Z}_p + 4\mathbb{G}_1}$	$\mathbf{2\mathbb{G}_1 + 1\mathbb{G}_2}$	$\mathbf{2\mathbb{G}_1}$
Randomized signature	$\mathbf{6\mathbb{G}_1 + 4\mathbb{G}_2}$	$\mathbf{6\mathbb{Z}_p + 3\mathbb{G}_1}$ $\mathbf{+4\mathbb{G}_2}$	$\mathbf{2\mathbb{G}_1}$

Table 8. Asymptotic size of the keys and signatures of the HIAC scheme, PS scheme, and IHABC scheme.

	HIAC **(bits)**	IHABC (bits)	PS (bits)
Issuer public key	**2688**	**768**	**1536**
Issuer secret key	**512**	**256**	**512**
Verifier public key	$\mathbf{256 \times 2k}$ $\mathbf{+384 \times 6k}$	$\mathbf{384 \times (k+1)}$ $\mathbf{+768 \times 3k}$	$\emptyset$
Verifier secret key	**512**	**256**	$\emptyset$
Signature	**1792**	**1536**	**768**
Randomized signature	**4608**	**5760**	**768**

Table 9. Size of the keys and signatures of the HIAC scheme, PS scheme, and IHABC scheme, using BLS12-381 setup, with compressed elements.

	Issuer	User	Verifier
Sign	$\mathbf{4\mathbb{G}_1}$	$\mathbf{2\mathbb{G}_1}$	$\emptyset$
VerifierSetup	$\mathbf{4\mathbb{G}_1}$	$\emptyset$	$\emptyset$
IntegrityVerification	$\emptyset$	$\emptyset$	$\mathbf{2k\mathbb{Z}_p + 4k\mathbb{G}_1}$
Credential Presentation	$\emptyset$	$\mathbf{12\mathbb{G}_1}$ $\mathbf{+4\mathbb{G}_2}$	$\mathbf{4\mathbb{G}_1}$

Table 10. Communication costs of each operation of the interactive HIAC scheme, for each issuers, users, and verifiers. Only upload costs are considerated. Credential PResentation operation takes into account the transmission of the user's randomized credential and the commitment reveal exchange.

versary can find x such that $h = g^x$ with non negligible probability.

Definition 16. [Computational Diffie Hellman (CDH) Assumption] Given $(g, g^x, g^y) \in \mathbb{G}$, with $\mathbb{G}$ a group generated by g, no PPT adversary can find g^{xy} with non negligible probability.

Definition 17. [Bilinear Diffie Hellman(BDH) Assumption] Given a type 3 bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, with $\langle g_1 \rangle = \mathbb{G}_1$, $\langle g_2 \rangle = \mathbb{G}_2$ and $(g_i^a, g_j^b, g_k^c), \forall i, j, k \in \{1, 2\}$, no PPT adversary can find $e(g_1, g_2)^{abc}$ with non negligible probability.

Definition 18. [Decisional Diffie Hellman (DDH) Assumption] Given $(g, g^x, g^y, h) \in \mathbb{G}$, with $\mathbb{G}$ a group generated by g, no PPT adversary can decide with non negligible probability if $h = g^{xy}$.

E.2 Aggregator Correctness

We prove in this subsection that the aggregator presented in Section 6.1.1 is correct.

Proof. We prove the correctness of the aggregator by developing the verification equation with values produced following the entire protocol:

$$\begin{aligned} e((W)'_{(i)}, C') &= e\left(g_1^{\mathsf{sk}\ r_2 \frac{1}{x_l}}, g_2^{x_l r_1}\right) \\ &= e(g_1, g_2)^{\mathsf{sk}\ r_2\ r_1} \\ &= e\left(g_1^{\mathsf{sk}}, g_2^{r_1\ r_2}\right) \\ &= e(g_1^{sk}, h) \end{aligned}$$

□

E.3 Aggregator Collision-Freedom

We want to prove that the aggregator scheme proposed in 6.1.1 is collision-free. We restate Theorem 2:

Theorem 2. ***CollisionFreedomGame***$(\mathcal{A}, \mathcal{C})$*, for a challenger* $\mathcal{C}$*, and an adversary* $\mathcal{A}$ *:*

- ***Setup****:* $\mathcal{C}$ *runs the aggregator* ***Setup*** *algorithm, chooses a set* $\mathcal{S}$*, runs the* ***Gen*** *algorithm to build a commitment to* $\mathcal{S}$*, and the associated verification key.* $\mathcal{A}$ *is given the aggregator's public information.*
- ***Output****:* $\mathcal{A}$ *outputs* C^* *a commitment to* s^* *and* π_s^**. The game outputs* 1 *if* $s^* \notin \mathcal{S}$*, and* ***Verify***$(pp, \mathsf{Agg}, \mathsf{aux}, \mathsf{sk}, C^*, \pi_s^*) = 1$*, or* 0 *otherwise.*

An Aggregator is said to be element-indistinguishable if for a negligible ϵ*, the probability for a* PPT *Adversary* $\mathcal{A}$ *can to win the* ***CollisionfreedomGame*** *is:*

$$\Pr[\boldsymbol{CollisionfreedomGame}(\mathcal{A}, \mathcal{C}) = 1] \leq \epsilon$$

We will prove Theorem 2 by contradiction, i.e. we assume the adversary is able to find a tuple $(C^*, \pi_s^* = ((W)'^{*}_{(l)}, h^*))$ such that:

$$e((W)'^{*}_{(l)}, C^*) = e(g_1^{\mathsf{sk}}, h^*))$$

We will process as follow:

1. Representation of the elements;

```
Theorem − 6 − Game₁𝒜
$(x_1, ..., x_k) \leftarrow\!\!\$ \, (\mathbb{Z}_p^*)^k$;
$\mathrm{PK}_1 = (X_1^{(1)}, ..., X_1^{(k)}) = (g_1^{x_1}, ..., g_1^{x_k})$
$\mathrm{PK}_2 = (X_2^{(1)}, ..., X_2^{(k)}) = (g_2^{x_1}, ..., g_2^{x_k})$
$\bar{\mathrm{PK}}_1 = (\bar{X}_1^{(1)}, \cdots, \bar{X}_1^{(k)}) = (g_1^{\frac{1}{x_1}}, \cdots, g_1^{\frac{1}{x_k}})$
$(W, \mathsf{sk}) \leftarrow\!\!\$ \mathbf{Gen}(\mathrm{PK}_1, \mathrm{PK}_2, \bar{\mathrm{PK}}_1)$
$(h^*, C^*, (W)'^*_{(l)}, \eta) \leftarrow\!\!\$ \, \mathcal{A}(\mathrm{PK}_1, \mathrm{PK}_2, \bar{\mathrm{PK}}_1, W)$;
$a_1 := e((W_y)I^*, \mathsf{Comm}(s^*))$
$a_2 := e(g_1^{\mathsf{sk}}, h^*)$
If $a_1 = a_2 \wedge X_2^{(I)*} \neq g_2^{x_l \eta}, \forall i \in \{1, ..., k\}$:
    return 1;
Else:
    return 0;
```

Fig. 6. Theorem 2 game representation. PK_1 is the set of aggregated values in $\mathbb{G}_1$ and PK_2 is the set of aggregated values in $\mathbb{G}_2$. η is a randomizing value, known by the adversary.

2. Proof that $(W)'^*_{(l)}$ is a combination of different $(W)_i$ $i \in \{1, \cdots, k\}$;
3. Proof that $(W)'^*_{(l)}$ is composed on only one $(W)_i$ $i \in \{1, \cdots, k\}$; and
4. Proof of Theorem 2.

E.3.1 Representation of the Elements

Figure 6 is a game representation of Theorem 2.

Remark 1. Probability of success of a game i is written as $\Pr[S_i]$. Success is obtained when a game returns 1. the adversary has a negligible advantage in $\mathsf{Game}_1\mathcal{A}$ if $\Pr[S_1] \leq \epsilon$, where ϵ is negligible.
For the ease of the proof, we set a notation for each element produced by the adversary :

$$C^* = g_2^{s_x}$$

$$\begin{aligned} \pi_s^* &= ((W)'^*_{(l)}, h^*) \\ &= (g_1^{s_w}, g_2^{s_h}) \end{aligned} \tag{1}$$

We develop the pairings a_1 and a_2 from Figure 6, using notation from Equation (1).

$$\begin{aligned} a_1 &= e((W)^*_{(l)}, C^*) \\ &= e(g_1^{s_w}, g_2^{s_x}) \\ &= g_T^{s_w s_x} \end{aligned} \tag{2}$$

```
Theorem − 6 − Game₂𝒜
$(x_1, ..., x_k) \leftarrow\!\!\$ \, (\mathbb{Z}_p^*)^k$;
$\mathrm{PK}_1 = (X_1^{(1)}, ..., X_1^{(k)}) = (g_1^{x_1}, ..., g_1^{x_k})$
$\mathrm{PK}_2 = (X_2^{(1)}, ..., X_2^{(k)}) = (g_2^{x_1}, ..., g_2^{x_k})$
$\bar{\mathrm{PK}}_1 = (\bar{X}_1^{(1)}, \cdots, \bar{X}_1^{(k)}) = (g_1^{\frac{1}{x_1}}, \cdots, g_1^{\frac{1}{x_k}})$
$(W, \mathsf{sk}) \leftarrow\!\!\$ \mathbf{Gen}(\mathrm{PK}_1, \mathrm{PK}_2, \bar{\mathrm{PK}}_1)$
$(g_2^{s_h}, g_1^{s_w}, g_2^{s_x} \ , \eta) \leftarrow\!\!\$ \, \mathcal{A}(\mathrm{PK}_1, \mathrm{PK}_2, \bar{\mathrm{PK}}_1, W)$;
$a'_1 := s_w s_x$
$a'_2 := s_h \mathsf{sk}$
If $a'_1 \equiv a'_2 \pmod{p-1} \wedge g_2^{s_x} \neq g_2^{x_i \eta}, \forall i \in \{1, ..., k\}$:
    return 1;
Else:
    return 0;
```

Fig. 7. Theorem 2 game - notation modifications

$$\begin{aligned} a_2 &= e(g_1^{\mathsf{sk}}, h^*) \\ &= e(g_1^{\mathsf{sk}}, g_2^{s_h}) \\ &= g_T^{s_h \mathsf{sk}} \end{aligned} \tag{3}$$

From the Equation $a_1 = a_2$, the Equation (2) and the Equation (3), we can write:

$$\begin{aligned} & a_1 = a_2 \\ \Leftrightarrow \quad & g_T^{s_x s_w} = g_T^{s_h \mathsf{sk}} \\ \Rightarrow \quad & s_x s_w \equiv s_h \mathsf{sk} \pmod{p-1} \end{aligned} \tag{4}$$

We rewrite $\mathsf{Game}_1\mathcal{A}$, using Equation (4) development. This game is presented in Figure 7

$\mathsf{Game}_1\mathcal{A}$ and $\mathsf{Game}_2\mathcal{A}$ are exactly the same games, with notation modification, this implies that $\Pr[S_1] = \Pr[S_2]$.

E.3.2 Proof that $(W)'^*_{(l)}$ is a Combination of Different $(W)_i$ $i \in \{1, \cdots, k\}$

The goal is to prove that $(W)'^*_{(l)}$ is a composition of $(W)_i$. Informally, it is possible to understand this statement as the fact that the adversary needs to output an element in $\mathbb{G}_1$ containing sk. Under CDH assumption, the only way he can achieve this is by using a combination of $(W)_i$.

First, we state that under type-3 bilinear pairing assumption, finding a morphism from $\mathbb{G}_2$ to $\mathbb{G}_1$ is as hard as solving the DL problem [19]. Then, from the equation $a'_1 = a'_2$, we know that $s_w s_x$ depends on sk. And sk is only disclosed as an exponent in $\mathbb{G}_1$. Thus, under DL assumption, the adversary has to include sk inside s_w. In other words $s_w = \mathsf{sk}\zeta$ for some ζ. Because sk is only known to the adversary in $(W)_i$, under CDH assumption we can deduct that $(W)'^{*}_{(l)}$ is a linear combination of $(W)_i$ values.

From the above discussion, s_w is composed of W values. Therefore, we can represent s_w as follow:

$$s_w = \mathsf{sk} \sum_{i=1}^{k} \left(\frac{b_i}{x_i} \right) \qquad (5)$$

Where the $b_i \in \mathbb{Z}_p, 1 \leq i \leq k$, are factors known and chosen by the adversary, and $\sum_{i=1}^{k} (|b_i|) \geq 1$.

Furthermore, we know that $\frac{s_w s_x}{s_h} = \mathsf{sk}$. Let us assume $x_1, \cdots, x_k = \mathsf{sk}$. In this case, $\deg(s_w) = 0$ in term of values unknown by the adversary, i.e. $\mathsf{sk}, x_1, \cdots, x_k$ (from Equation (5)). We thus face an easier problem : $\frac{s_x}{s_h} = \mathsf{sk}$. Solving the first problem implies being able to solve the second one. Furthermore, s_x and s_h are outputed by the adversary in $\mathbb{G}_2$, this mean that they are buil using (g_2, g_2^{sk}). From the equation $\frac{s_x}{s_h} = \mathsf{sk}$ we know that s_h should be of degree 0, or lower. However, getting $\deg(s_h) < 0$ is directly equivalent to the Diffie Hellman inversion problem in this case, which is equivalent to CDH problem [12]. Thus $\deg(s_h) = 0$, in other world, the adversary knows the discrete logarithm of $g_2^{s_h}$. We can thus conclude that $\deg(s_x) = 1$.

According to this discussion, we write a new game, presented in Figure 8. We call this new game: $\mathsf{Game}_3\mathcal{A}$.

The transition between $\mathsf{Game}_3\mathcal{A}$ and $\mathsf{Game}_2\mathcal{A}$ is a failure based transition, where the failure event is the probability for the adversary to find a solution to the CDH problem or the DL problem. This means :

$$|\Pr[S_3] - \Pr[S_2]| \leq \epsilon_{\mathrm{CDH}} + \epsilon_{\mathrm{DL}}$$

E.3.3 Proof that $(W)'^{*}_{(l)}$ is Composed of Only One $(W)_i$ $i \in \{1, \cdots, k\}$

If we rewrite the equation $a'_1 = a'_2$ from $\mathsf{Game}_3\mathcal{A}$, we obtain:

$$s_x \mathsf{sk} \sum_{i=1}^{k} \left(\frac{b_i}{x_i} \right) = s_h \mathsf{sk}$$

$$\Leftrightarrow s_x \sum_{i=1}^{k} \left(\frac{b_i}{x_i} \right) = s_h$$

```
Theorem – 6 – Game₃𝒜
$(x_1, ..., x_k) \leftarrow\$ (\mathbb{Z}_p^*)^k;$
$\mathrm{PK}_1 = (X_1^{(1)}, ..., X_1^{(k)}) = (g_1^{x_1}, ..., g_1^{x_k})$
$\mathrm{PK}_2 = (X_2^{(1)}, ..., X_2^{(k)}) = (g_2^{x_1}, ..., g_2^{x_k})$
$\bar{\mathrm{PK}}_1 = (\bar{X}_1^{(1)}, \cdots, \bar{X}_1^{(k)}) = (g_1^{\frac{1}{x_1}}, \cdots, g_1^{\frac{1}{x_k}})$
$(W, \mathsf{sk}) \leftarrow\$ \mathbf{Gen}(\mathrm{PK}_1, \mathrm{PK}_2, \bar{\mathrm{PK}}_1)$
$(g_2^{s_h}, g_1^{\mathsf{sk} \sum_{i=1}^{k} \left( \frac{b_i}{x_i} \right)}$
    $, g_2^{s_x}, \eta) \leftarrow\$ \mathcal{A}(\mathrm{PK}_1, \mathrm{PK}_2, \bar{\mathrm{PK}}_1, W);$
$a'_1 := s_x \mathsf{sk} \sum_{i=1}^{k} \left( \frac{b_i}{x_i} \right)$
$a'_2 := s_h \mathsf{sk}$
If $a'_1 = a'_2 \wedge g_2^{s_y} \neq g_2^{x_i \eta}, \forall i \in \{1, ..., k\}$:
    return 1;
Else:
    return 0;
```

Fig. 8. s_w representation game

We need to study the value $\frac{s_x}{s_h}$. We want to prove that no adversary can output $\frac{s_x}{s_h} = \frac{1}{\sum_{i=1}^{k} \left(\frac{b_i}{x_i} \right)}$ unless $b_i = 0$ for $(k-1)$ i's. Or, in other words, that $\frac{s_x}{s_h} = \frac{x_I}{b_I}, 1 \leq I \leq k$, and $b_I \in \mathbb{Z}_p^*$.

Another way to explain this expression is to say that the Adversary has to use exactly one witness to build the $(W)_I^*$ value.

Lemma 2. Given $(g_2, g_2^{x_1}, ..., g_2^{x_k})$, no adversary can output, with non negligible probability, $g_2^{s_x}$ and $g_2^{s_h}$ such that $\frac{s_x}{s_h} = \frac{1}{\sum_{i=1}^{k} \left(\frac{b_i}{x_i} \right)}$, and more than one $b_i \neq 0, \forall i \in \{1, ..., k\}$.

We will prove Lemma 2 by contradiction, proving that finding such values $g_2^{s_h}$ and $g_2^{s_x}$ is at least as hard as solving the CDH problem.

Proof. Hypothesis 1 : Given $(g_2, g_2^{x_1}, ..., g_2^{x_k})$, an adversary can output, with non negligible probability, $g_2^{s_x}$ and $g_2^{s_h}$ such that $\frac{s_x}{s_h} = \frac{1}{\sum_{i=1}^{k} \left(\frac{b_i}{x_i} \right)}$, and more than one $b_i \neq 0, \forall i \in \{1, ..., k\}$.

We know from the previous step of the proof that $\deg(s_h) = 0$, thus, we only study the values taken by s_x, assuming s_h is a value known by the adversary, independent of the variables $x_i, \forall i \in \{1, \cdots, k\}$.

We study the case where they are only two variables, i.e., we give to the adversary $(g_2, g_2^{x_1}, g_2^{x_2})$. We will prove that CDH problem can be reduced to this simplified version of the problem. We write the restricted version of the game:

```
Theorem − 6 − Game$_5\mathcal{A}$
$(x_1, ..., x_k) \leftarrow\!\$ \, (\mathbb{Z}_p^*)^k$;
$\mathrm{PK}_1 = (X_1^{(1)}, ..., X_1^{(k)}) = (g_1^{x_1}, ..., g_1^{x_k})$
$\mathrm{PK}_2 = (X_2^{(1)}, ..., X_2^{(k)}) = (g_2^{x_1}, ..., g_2^{x_k})$
$\bar{\mathrm{PK}}_1 = (\bar{X}_1^{(1)}, \cdots, \bar{X}_1^{(k)}) = (g_1^{\frac{1}{x_1}}, \cdots, g_1^{\frac{1}{x_k}})$
$(W, \mathsf{sk}) \leftarrow\!\$ \mathbf{Gen}(\mathrm{PK}_1, \mathrm{PK}_2, \bar{\mathrm{PK}}_1)$
$(g_2^{s_h}, I, g_1^{\mathsf{sk}\frac{b_I}{x_I}}$
    $, g_2^{s_x}, \eta) \leftarrow\!\$ \, \mathcal{A}(\mathrm{PK}_1, \mathrm{PK}_2, \bar{\mathrm{PK}}_1, W)$;
$a_1' := \mathsf{sk} s_x b_I \frac{b_I}{x_I}$
$a_2' := \mathsf{sk} s_h$
If $a_1' = a_2' \wedge g_2^{s_x} \neq g_2^{x_i \eta}, \forall i \in \{1, \cdots, k\}$:
    return 1;
Else:
    return 0;
```

Fig. 9. Theorem 2 game with reduction of the s_w value

Game 1. Let $\mathbb{G}_2$ be a group, let g_2 be a generator of this group, and let x_1, x_2 be two elements in $\mathbb{Z}_p^*$. Given $(g_2, g_2^{x_1}, g_2^{x_2})$, output $g_2^{\frac{x_1 x_2}{x_1 b_2 + x_2 b_1}}$, for some $b_1, b_2 \in \mathbb{Z}_p^*$.

Remark 2. It is easy to reduce Lemma 2 to Game 1, by adding ad hoc values $x_3 = 1, \cdots, = x_k = 1$ to the construction of Game 1.
If the adversary has a non negligible probability of success in Game 1, then he has access to an algorithm A, that given (g, g^{x_1}, g^{x_2}), outputs with non negligible probability $g^{\frac{x_1 x_2}{x_1 b_1 + x_2 b_2}}$, for any values b_1, b_2, independent from x_1 and x_2. Given this algorithm, the adversary can query: $A(g, g^{1+x_1}, g^{1-x_1}) = g^{\frac{(1+x_1)(1-x_1)}{(1+x_1)+(1-x_1)}} = g^{\frac{1-x_1^2}{2}}$. Lets call this value a. Then, the adversary is able to extract $g^{x^2} = a^{-2} \cdot g$. Thus, the algorithm A can solve the Diffie Hellman squaring problem, which is equivalent to CDH problem [12]. Therefore, under CDH assumption, no PPT adversary can solve Game 1 with probability greater than ϵ_{CDH}, with ϵ_{CDH} the probability to solve the CDH problem. By extension, hypothesis 1 is contradicted. Thus Lemma 2 is proven. □

E.3.4 Proof of Theorem 2

We have $\frac{s_x}{s_h} = \frac{x_I}{b_I}, 1 \leq I \leq k$, and $b_I \in \mathbb{Z}_p^*$.

We rewrite Game$_3\mathcal{A}$ using Lemma 2. This new game is provided in Figure 9. We call this new game: Game$_4\mathcal{A}$.

```
Theorem − 6 − Game$_6\mathcal{A}$
$(x_1, ..., x_k) \leftarrow\!\$ \, (\mathbb{Z}_p^*)^k$;
$\mathrm{PK}_1 = (X_1^{(1)}, ..., X_1^{(k)}) = (g_1^{x_1}, ..., g_1^{x_k})$
$\mathrm{PK}_2 = (X_2^{(1)}, ..., X_2^{(k)}) = (g_2^{x_1}, ..., g_2^{x_k})$
$\bar{\mathrm{PK}}_1 = (\bar{X}_1^{(1)}, \cdots, \bar{X}_1^{(k)}) = (g_1^{\frac{1}{x_1}}, \cdots, g_1^{\frac{1}{x_k}})$
$(W, \mathsf{sk}) \leftarrow\!\$ \mathbf{Gen}(\mathrm{PK}_1, \mathrm{PK}_2, \bar{\mathrm{PK}}_1)$
$(g_2^{s_x}, \eta') \leftarrow\!\$ \, \mathcal{A}(\mathrm{PK}_1, \mathrm{PK}_2, \bar{\mathrm{PK}}_1, W)$;
If $g_2^{s_x} = g_2^{x_I \eta'}, \forall I \in \{1, ..., k\}$
    $\wedge g_2^{s_x} \neq g_2^{x_I \eta'}, \forall I \in \{1, ..., k\}$:
    return 1;
Else:
    return 0;
```

Fig. 10. Theorem 2 game with develloped a_1' and a_2'

The transition between Game$_4\mathcal{A}$ and Game$_3\mathcal{A}$ is a failure based transition, where the failure event is the probability for the adversary to find a solution to the CDH problem. This means :

$$\begin{aligned} |\Pr[S_4] - \Pr[S_3]| &\leq F \\ &\leq \epsilon_{\mathrm{CDH}} \end{aligned}$$

We develop again $a_1' = a_2'$ with values from Game$_5\mathcal{A}$:

$$\begin{aligned} \mathsf{sk} s_x \frac{b_I}{x_I} &= \mathsf{sk} s_h \prod_{j=1}^{k} (x_j) \\ \Leftrightarrow s_x &= x_I \frac{s_h}{b_I} \end{aligned}$$

We write a final game, directly developing a_1' and a_2', with $\frac{s_h}{b_I} = \eta'$. This game is presented in Figure 10

This last transition is a notation modification, thus $\Pr[S_5] = \Pr[S_4]$.

And the probability of success of Game$_6\mathcal{A}$ is obviously 0, as the condition of success is a contradiction.

Now we can evaluate probability of success of Game$_1\mathcal{A}$:

$$|\Pr[S_5] - \Pr[S_1]| \leq 2 \cdot \epsilon_{\mathrm{CDH}} + \epsilon_{\mathrm{DL}}$$

And, as $\Pr[S_5] = 0$:

$$\Pr[S_1] \leq 2 \cdot \epsilon_{\mathrm{CDH}} + \epsilon_{\mathrm{DL}} \tag{6}$$

Probability of success of Game$_1\mathcal{A}$ is negligible. Therefore our aggregator construction is collision free.

E.4 Proof of Element-Indistinguishability

Theorem 3. *The aggregator scheme presented in Section 6.1.1 is element indistinguishable. Given an aggregator Agg_x, the set of secret values accumulated in the aggregator $(x_1, \cdots, x_k\}$, $k \geq 2$, a set-membership proof π_s, and a randomized commitment C' to an element of the set $\mathcal{S}$, no* PPT *adversary can find with non negligible probability the element s the user committed to.*

The element indistinguishable property states that no malicious verifier can, with non negligible probability, decide which elements the user commits to. In order to prove this statement, we need a preliminary lemma:

Lemma 3. Given $(g, g^{x_1}, \cdots, g^{x_n}) \in \mathbb{Z}_p^n$. Under DDH assumption, no PPT adversary is able to distinguish between non linear expression of the variables with non negligible probability.

Proof. We will prove Lemma 3 by contradiction.

We assume that, given $(g, g^{x_1}, \cdots, g^{x_n}, h)$, the adversary (malicious verifier) is able to decide with non negligible probability if $h \overset{?}{=} g^{x_1 + \cdots + x_i x_j + \cdots + x_n}$, for $i, j \in \{1, \cdots, n\}$. If we set $x_k = 0, \forall k \neq \{i, j\}$, then this decision the adversary is able to make is equivalent to find, with non negligible probability, a solution to the DDH problem. Under DDH assumption, this has negligible probability to occur. This implies that Lemma 3 is true. □

Given this lemma, it is possible to prove Theorem 3:

Proof. Let define element indistinguishability with two elements x_1 and x_2. With the bilinear map e, and the elements he is given, the adversary is able to compute the tuple $(g_T, g_T^{r_1 a}, g_T^{r_2 b}, g_T^{r_1 r_2}, g_T^{r_1 r_2 ab})$. He must decide, with non negligible probability whether $(a = x_1 \wedge b = x_2) \vee (a = x_2 \wedge b = x_1)$.

The order of $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ is prime, thus every element of these groups is a generator, except from the neutral element. Therefore, every elements presented in Table 12 will be seen, from the adversary point of view, as elements chosen uniformly at random in one of these groups.

Furthermore the value $g_T^{r_1 r_2 ab}$ is symmetrical. The adversary knows this value, but cannot use it in any way to distinguish between a and b. And $g_T^{r_1 r_2}$ does not depend on a nor b.

Therefore, the adversary has to combine and compare the different element he is given to be able to distinguish between a and b.

However, we can argue, using Lemma 3, that the adversary is not able to combine the values $(g_T, g_T^{r_1 a}, g_T^{r_2 b}, g_T^{r_1 r_2})$ to take its decision.

Thus the adversary is not able to decide whether $(a = x_1 \wedge b = x_2) \vee (a = x_2 \wedge b = x_1)$ under DDH assumption. Thus our aggregator implementation is element indistinguishable. □

E.5 Signature Correctness

We prove in this subsection that the signature presented in Section 6.1.2 is correct.

Proof. First, the aggregator is correct, thus, the verification of an honestly built aggregator will always output 1. Then, let $\sigma' = (h_1', h_2', \sigma_1', \sigma_2')$ be a randomized signature built following the above protocol, let $(X^{(I)'}, Y_2^{(I)'})$ be commitments to trusted issuer's keys, and let $vsk = (\mathsf{sk}_x, \mathsf{sk}_y)$ be a secret verifier key. We can verify that the signature is correct:

$$
\begin{aligned}
e(h_1^{'}, X^{(i)'} Y_2^{(i)' \mathsf{H}(m)}) \\
&= e(g_1^{r_{I,1} r_u'}, g_2^{x_I r_u + r_u y_I R_y \mathsf{H}(m)}) \\
&= e(g_1, g_2)^{r_{(I,1)} r_u' r_u (x_I + y_I R_y \mathsf{H}(m))} \\
&= e(g_1^{r_{(I,1)} r_u' r_u (x_I + y_I R_y \mathsf{H}(m))}, g_2) \\
&= e(\sigma_1', g_2)
\end{aligned}
$$

$$
\begin{aligned}
e(h_2^{'}, X^{(i)'} Y_2^{(i)' \mathsf{H}(\mathsf{H}(m))}) \\
&= e(g_1^{r_{(I,2)} r_u^{''}}, g_2^{x_I r_u + r_u y_I R_y \mathsf{H}(\mathsf{H}(m))}) \\
&= e(g_1, g_2)^{r_{(I,2)} r_u^{''} r_u (x_I + y_I R_y \mathsf{H}(\mathsf{H}(m)))} \\
&= e(g_1^{r_{(I,2)} r_u^{''} r_u (x_I + y_I R_y \mathsf{H}(\mathsf{H}(m)))}, g_2) \\
&= e(\sigma_2', g_2)
\end{aligned}
$$

□

E.6 EUF-CMA Proof

We prove that the signature scheme presented in Section 6.1.2 has the EUF-CMA property.

Here is a summary of the steps of the proof:

1. We use Appendix E.3 proof to represent the commitment the adversary makes to the issuer's keys;
2. We present the elements of the signature as polynomials, as a function of the elements published by the verifier and the issuers;

3. We reduce these polynomials; and
4. We analyze the reduced polynomials, and we prove that if the adversary is able to output such signature, either he is also able to invert a hash funcion, or he is using an already queried signature.

As a preliminary, we give the notations we will use to discuss a forged signature. This notation is used for a potential σ^* value, which is a forged signature on a message m^* never queried to one of the verifier's trusted issuer's signing oracle. We note this signature:

$$\begin{aligned} \sigma^* &= (h_1^*, h_2^*, \sigma_1^*, \sigma_2^*) \\ &= (g_1^{s_1}, g_1^{s_2}, g_1^{s_{\sigma_1}}, g_1^{s_{\sigma_2}}) \end{aligned} \quad (7)$$

Theorem 7. Given the tuple $(g_1, Y_1, Y_2, X, \bar{X}_1, \bar{Y}_1)$, and given access to a signing oracle, that can sign messages on behalf of all issuers, no adversary can output, with non negligible probability, a signature $(h_1^*, h_2^*, \sigma_1^*, \sigma_2^*, X^{(I_1)*}, Y_2^{(I_2)*})$ that was not queried to the signing oracle, and that verifies:

$$e(h_1^*, X^{(I_1)*}(Y_2^{(I_2)*})^{\mathsf{H}(m^*)}) = e(\sigma_1^*, g_2)$$
$$e(h_2^*, X^{(I_1)*}(Y_2^{(I_2)*})^{\mathsf{H}(\mathsf{H}(m^*))}) = e(\sigma_2^*, g_2)$$

Remark 3. Theorem 7 represents commitment to x and y issuer's keys as independent. Indeed, the two aggregator, and therefore the two associated set-membership proofs are independent. At this point of the proof, we can not know if I_1 and I_2 are equal or different. We will see later that they are indeed equal.

Proof. We will prove Theorem 7 by contradiction.

We present in Figure 11 the EUF-CMA property as a game, for a generic signature scheme. We define M as the set of messages already queried to the oracle. $\mathsf{OSign}(\mathsf{sk}, \bullet)$ represents the access to the signing oracle. Any query to the oracle outputs k different signature, issued by the k different trusted issuers.

```
EUF − CMA_1 A
(sk, pk) ←$ KeyGen($1^\lambda$);
($m^*, \sigma^*$) ←$ $\mathcal{A}^{\mathsf{OSign}(\mathsf{sk},\bullet)}(\mathsf{pk})$;
If Verify($pk, m^*, \sigma^*$) = 1 and $m^* \notin M$:
    return 1;
Else:
    return 0;
```

Fig. 11. EUF-CMA game. M represents the already queried messages.

The advantage of the adversary in the EUF-CMA game is:

$$\mathsf{Adv}_{\Sigma}^{\text{euf-cma}}(\mathcal{A}) = |\Pr[S_1] - (p-1)^{-1}| \quad (8)$$

Where $\Pr[S_i] = \Pr[\mathsf{Game}_i\mathcal{A} = 1]$.

We modify the generic game presented in Figure 11 with the values of our protocol. The new game is presented in Figure 12.

```
EUF − CMA_2 A
$SK_x = (x_1, ..., x_k) \leftarrow\$ (\mathbb{Z}_p^*)^k$;
$SK_y = (y_1, ..., y_k) \leftarrow\$ (\mathbb{Z}_p^*)^k$;
$X = (X^{(1)}, \cdots, X^{(k)}) = (g_2^{x_1}, \cdots, g_2^{x_k})$
$Y_1 = (Y_1^{(1)}, \cdots, Y_1^{(k)}) = (g_1^{y_1}, \cdots, g_1^{y_k})$
$Y_2 = (Y_2^{(1)}, \cdots, Y_2^{(k)}) = (g_2^{y_1}, \cdots, g_2^{y_k})$
$\bar{X}_1 = (\bar{X}_1^{(1)}, \cdots, \bar{X}_1^{(k)}) = (g_1^{\frac{1}{x_1}}, \cdots, g_1^{\frac{1}{x_k}})$
$\bar{Y}_1 = (\bar{Y}_1^{(1)}, \cdots, \bar{Y}_1^{(k)}) = (g_1^{\frac{1}{y_1}}, \cdots, g_1^{\frac{1}{y_k}})$
$PK = \{X, \bar{X}_1, Y_1, \bar{Y}_1, Y_2\}$
$(m^*, h_1^*, h_2^*, \sigma_1^*, \sigma_2^*,$
        $X^{(I_1)*}, Y^{(I_2)*}) \leftarrow\$ \mathcal{A}^{\mathsf{OSign}(\mathrm{SK_x},\mathrm{SK_y},\bullet)}(PK)$;
$a_1 = e(h_1^*, X^{(I_1)*}(Y_2^{(I_2)*})^{\mathsf{H}(m)^*})$
$a_2 = e(\sigma_1^*, g_2)$
$a_3 = e(h_2^*, X^{(I_1)*}(Y_2^{(I_2)*})^{\mathsf{H}(\mathsf{H}(m))^*})$
$a_4 = e(\sigma_2^*, g_2)$
If $a_1 = a_2 \wedge a_3 = a_4 \wedge m^* \notin M$:
    return 1;
Else:
    return 0;
```

Fig. 12. EUF-CMA game - notation modifications. M represents the already queried messages.

We rewrite this game, using the result of Theorem 2, i.e. $X^{(I_1)*} = g_2^{x_{I_1}\eta_x}$ and $Y_2^{(I_2)*} = g_2^{y_{I_2}\eta_y}$. Where I_1 and I_2 represent two verifier's trusted issuers, and η_x and η_y

```
EUF − CMA₃𝒜
1 :  $SK_x = (x_1, ..., x_k) \leftarrow\$ (\mathbb{Z}_p^*)^k$;
2 :  $SK_y = (y_1, ..., y_k) \leftarrow\$ (\mathbb{Z}_p^*)^k$;
3 :  $X = (X^{(1)}, ..., X^{(k)}) = (g_2^{x_1}, ..., g_2^{x_k})$
4 :  $Y_1 = (Y_1^{(1)}, \cdots, Y_1^{(k)}) = (g_1^{y_1}, \cdots, g_1^{y_k})$
5 :  $Y_2 = (Y_2^{(1)}, \cdots, Y_2^{(k)}) = (g_2^{y_1}, \cdots, g_2^{y_k})$
6 :  $\bar{X}_1 = (\bar{X}_1^{(1)}, \cdots, \bar{X}_1^{(k)}) = (g_1^{\frac{1}{x_1}}, \cdots, g_1^{\frac{1}{x_k}})$
7 :  $\bar{Y}_1 = (\bar{Y}_1^{(1)}, \cdots, \bar{Y}_1^{(k)}) = (g_1^{\frac{1}{y_1}}, \cdots, g_1^{\frac{1}{y_k}})$
8 :  $PK = \{X, \bar{X}_1, Y_1, \bar{Y}_1, Y_2\}$
9 :  $(m^*, g_1^{s_1}, g_1^{s_2}, g_1^{s_{\sigma_1}}, g_1^{s_{\sigma_1}}, X^{(I_1)*} = g_2^{x_{I_1}\eta_x}$,
10 :   $Y^{(I_2)*} = g_2^{y_{I_2}\eta_y}, \eta_x, \eta_y, ) \leftarrow\$ \mathcal{A}^{\mathsf{OSign}(\mathrm{SK_x}, \mathrm{SK_y}, \bullet)}(PK)$;
11 :  $a_1 = e(g_1^{s_1}, g_2^{x_{I_1}\eta_x} g_2^{\mathsf{H}(m^*) y_{I_2}\eta_y})$
12 :  $a_2 = e(g_1^{s_{\sigma_1}}, g_2)$
13 :  $a_3 = e(g_1^{s_2}, g_2^{x_{I_1}\eta_x} g_2^{y_{I_2}\mathsf{H}(\mathsf{H}(m^*))\eta_y})$
14 :  $a_4 = e(g_1^{s_{\sigma_2}}, g_2)$
15 :  If $a_1 = a_2 \wedge a_3 = a_4 \wedge m^* \notin M$:
16 :    return 1;
17 :  Else:
18 :    return 0;
```

Fig. 13. EUF-CMA game - Theorem 2 modifications

represent two random values known by the adversary. We also use the notations from Equation (7). We assume the adversary makes n queries to the Signing Oracles. This new game is represented in Figure 13.

The difference between probability of success of the game $\mathsf{EUF-CMA}_3\mathcal{A}$ and the game $\mathsf{EUF-CMA}_2\mathcal{A}$ is the probability for the Adversary to solve the problem defined in Theorem 2:

$$|\Pr[S_3] - \Pr[S_2]| \leq 2 \cdot \epsilon_{\mathrm{CDH}} + \epsilon_{\mathrm{DL}}$$

We develop the equations in line 13 from the game $\mathsf{EUF-CMA}_3\mathcal{A}$. The goal is to represent these equations as polynomials as a function of public values given to the adversary. With $h^* = \mathsf{H}(m^*)$:

$$\begin{cases} a_1 = a_2 \\ a_3 = a_4 \end{cases}$$

$$\Leftrightarrow \begin{cases} e(g_1^{s_1}, g_2^{x_{I_1}\eta_x} g_2^{h^* y_{I_2}\eta_y}) = e(g_1^{s_{\sigma_1}}, g_2) \\ e(g_1^{s_2}, g_2^{x_{I_1}\eta_x} g_2^{y_{I_2}\mathsf{H}(h^*)\eta_y}) = e(g_1^{s_{\sigma_2}}, g_2) \end{cases}$$

$$\Leftrightarrow \begin{cases} g_T^{s_1(x_{I_1}\eta_x + h^* y_{I_2}\eta_y)} = g_T^{s_{\sigma_1}} \\ g_T^{s_2(x_{I_1}\eta_x + y_{I_2}\mathsf{H}(h^*)\eta_y)} = g_T^{s_{\sigma_2}} \end{cases}$$

$$\Rightarrow \begin{cases} s_{\sigma_1} = s_1(x_{I_1}\eta_x + y_{I_2}h^*\eta_y) \pmod{p-1} \\ s_{\sigma_2} = s_2(x_{I_1}\eta_x + y_{I_2}\mathsf{H}(h^*)\eta_y) \pmod{p-1} \end{cases} \tag{9}$$

The adversary outputs $g_1^{s_1}, g_1^{s_{\sigma_1}}, g_1^{s_2}, g_1^{s_{\sigma_2}}$ in $\mathbb{G}_1$. Thus these elements must be computed using element in $\mathbb{G}_1$ – because there is no efficiently computable morphism from $\mathbb{G}_2$ to $\mathbb{G}_1$ and from $\mathbb{G}_T$ to $\mathbb{G}_1$, under type-3 pairing assumption. The elements known to the adversary in $\mathbb{G}_1$ after n queries to the Signing Oracles, are:

$$E = \Bigg(g_1, Y_1, \bar{X}_1, \bar{Y}_1, \{g_1^{(r_{(I_j,1)})_j}\}_{j=1}^n, \{g_1^{(r_{I_j,2})_j}\}_{j=1}^n, \left\{ g_1^{r_{I_j,1}(x_{I_j} + h_j y_{I_j}(R_y)_j)} \right\}_{j=1}^n, \left\{ g_1^{r_{I_j,2}(x_{I_j} + \mathsf{H}(h_j) y_{I_j}(R_y)_j)} \right\}_{j=1}^n \Bigg).$$

In our case we want to express s_1, s_2, s_{σ_1} and s_{σ_2} as polynomials, as a function of the values unknown to the adversary. First, we represent an ad hoc element $g_1^{z_\alpha}$ in the form of a polynomial as a function of the values unknown to the adversary under DL assumption.

We then represent each signature's elements (i.e. $s_1, s_2, s_{\sigma_1}, s_{\sigma_2}$) in the same way as we did with the ad hock element $g_1^{z_\alpha}$.

We assume the adversary makes n queries to the Signing Oracles. Therefore, he receives n signatures from the issuer I_1, and n signatures from the issuer I_2. The variable's factors potentially added by the adversary are represented by a $11 \times n$ matrix α, and an additional factor w_α.

Here is the representation of $g_1^{z_\alpha}$:

$$g_1^{z_\alpha} = g_1^{w_\alpha} \cdot \prod_{j=1}^{k} \left((\bar{X}_1^{(j)})^{\alpha_{(1,j)}} \cdot (\bar{Y}_1^{(j)})^{\alpha_{(2,j)}} \cdot (Y_1^{(j)})^{\alpha_{(3,j)}} \right) \cdot \prod_{i=1}^{n} \Bigg((h_{(1,I_1)})_i^{\alpha_{(4,i)}} \cdot (h_{(1,I_2)})_i^{\alpha_{(5,i)}} \cdot (h_{(2,I_1)})_i^{\alpha_{(6,i)}} \cdot (h_{(2,I_2)})_i^{\alpha_{(7,i)}} \cdot (\sigma_{(1,I_1)})_i^{\alpha_{(8,i)}} \cdot (\sigma_{(2,I_1)})_i^{\alpha_{(9,i)}} \cdot (\sigma_{(1,I_2)})_i^{\alpha_{(10,i)}} \cdot (\sigma_{(2,I_2)})_i^{\alpha_{(11,i)}} \Bigg)$$

$$
\begin{aligned}
\Rightarrow z_\alpha =& w_\alpha + \sum_{j=1}^{k} \left(\alpha_{(1,j)} \frac{1}{x_j} + \alpha_{(1,j)} \frac{1}{y_j} + \alpha_{(3,j)} y_j \right) \\
&+ \sum_{i=1}^{n} \Bigg((r_{(I_1,1)})_i \alpha_{(4,i)} + (r_{(I_2,1)})_i \alpha_{(5,i)} \\
&+ (r_{(I_1,2)})_i \alpha_{(6,i)} + (r_{(I_2,2)})_i \alpha_{(7,i)} \\
&+ (r_{(I_1,2)})_i \alpha_{(8,i)} (x_{I_1} + R_y y_{I_1} h_i) \\
&+ (r_{(I_1,2)})_i \alpha_{(9,i)} (x_{I_1} + R_y y_{I_1} \mathsf{H}(h_i)) \\
&+ (r_{(I_2,1)})_i \alpha_{(10,i)} (x_{I_2} + R_y y_{I_2} h_i) \\
&+ (r_{(I_2,2)})_i \alpha_{(11,i)} (x_{I_2} + R_y y_{I_2} \mathsf{H}(h_i)) \Bigg) \\
&\pmod{p-1} \qquad (10)
\end{aligned}
$$

Remark 4. If $n > k$, columns 1 to 3 of matrix α are shorter than the other columns. We assume these columns are filed with zeroes to match the size of the matrix. If $k > n$ then it is the columns from 4 to 11 which are filled with zeroes.

Furthermore, we assume that all R_y values are similar. This does not change the validity of the proof, and it simplifies the notations.

Lemma 4. The probability for the adversary to output an element in $\mathbb{G}_1$ different from the z_α representation is inferior or equal to the probability of winning in one instance of the CDH game.

Proof. We prove this lemma by contradiction. We assume the adversary is able to output something different from representation presented in Equation (10). This is equivalent to say the adversary has access to an Oracle $O(g_1, g_1^{z_1}, ..., g_1^{z_{3k}}, g_1^{z_{3k+1}}, ..., g_1^{z_{3k+1+8n}})$. With $z_i, i \in \{1, ..., 3k+1+8n\}$, the variables unknown to the adversary. This oracle outputs with non negligible probability an element $g_1^{z_\alpha + z_a z_b}$, where $a, b \in \{1, ..., 3k+1+8n\}$. This oracle allows the adversary to output $g_1^{z_a f(z_b)}$, with f a polynomial function, and $\deg f_{(}z_b) \neq 0$. Thus, for any given CDH problem $(g_1, g_1^{z_a}, g_1^{z_b})$, an adversary with such an oracle can set all variables to 0, except z_a and z_b, and the oracle solves an instance of the CDH problem. □

We represents s_1, s_2, s_{σ_1}, and s_{σ_2} in the same way we represented z_α in Equation (10). Then, we replace s_1, s_2, s_{σ_1}, and s_{σ_2} in Equation (9) with this new representation.

Factor matrices of s_1, s_2, s_{σ_1}, and s_{σ_2} representations (i.e. the α matrices in Equation (10)) will be written respectively a, b, c and d. To simplify this discussion, we will assume the transformations applied to the first part of Equation (9) can be applied symmetrically to the second part of the equation.

Let us take:

$$
s_{\sigma_1} = s_1(x_{I_1}\eta_x + y_{I_2} h^* \eta_y) \qquad (11)
$$

If s_{σ_1} and s_1 validates Equation (11), then either $s_{\sigma_1} = 0$ and $s_1 = 0$, or s_{σ_1} and s_1 depends on the variables $x_{I_1}, x_{I_2}, y_{I_1}, y_{I_2}, (r_{(I_1,1)})_i, (r_{(I_1,2)})_i, (r_{(I_2,1)})_i$, and $(r_{(I_2,2)})_i$, $\forall i \in \{1, \cdots, n\}$. However $s_{\sigma_1} = 0$ or $s_1 = 0$ is a rejecting condition, thus s_{σ_1} and s_1 depends on the variables of the equation. Under DL and CDH assumptions, the adversary has a negligible probability to output a value a, b, c or d that depends on these variables. Therefore, we can reduce s_{σ_1} and s_1 to:

$$
\begin{aligned}
s_1 = \sum_{i=1}^{n} \Bigg(&(r_{(I_1,1)})_i a_{(4,i)} + (r_{(I_2,1)})_i a_{(5,i)} \\
&+ (r_{(I_1,2)})_i a_{(6,i)} + (r_{(I_2,2)})_i a_{(7,i)} \Bigg) \qquad (12)
\end{aligned}
$$

$$
\begin{aligned}
s_{\sigma_1} = \sum_{i=1}^{n} \Bigg(&(r_{(I_1,1)})_i c_{(8,i)} (x_{I_1} + R_y y_{I_1} h_i) \\
&+ (r_{(I_1,2)})_i c_{(9,i)} (x_{I_1} + R_y y_{I_1} \mathsf{H}(h_i)) \\
&+ (r_{(I_2,1)})_i c_{(10,i)} (x_{I_2} + R_y y_{I_2} h_i) \\
&+ (r_{(I_2,2)})_i c_{(11,i)} (x_{I_2} + R_y y_{I_2} \mathsf{H}(h_i)) \Bigg) \qquad (13)
\end{aligned}
$$

Now that we simplified representations of s_1 and s_{σ_1} we want to study the possibility for I_1 and I_2 to represent two distinct issuers.

We see that right side of Equation (11) contains elements $(r_{(I_1,1)})_i$ and $(r_{(I_1,2)})_i$ multiplied by y_{I_2}. At the same time, there are also elements $(r_{(I_2,1)})_i$ and $(r_{(I_2,2)})_i$ multiplied by x_{I_1}. These terms does not exist on the left side of the equation (in s_{σ_1}). We can deduce that, either the whole equation is reduced to zero, which is impossible since $g_1^{s_1} \neq 1_{\mathbb{G}_1}$, or, $I_1 = I_2 = I$. This means that all elements come from only one issuer.

We represent Equation (11) with these new representations of s_{σ_1} and s_1.

Remark 5. In the Equation (14), c'_1 is the concatenation of c_8 and c_{10}, $c'_1 = \left(\begin{smallmatrix} c_8 \\ c_{10} \end{smallmatrix}\right)$. c'_2 is the concatenation of row c_9 and row c_{11}, $c'_2 = \left(\begin{smallmatrix} c_9 \\ c_{11} \end{smallmatrix}\right)$. a'_1 is the concatenation of row a_4 and row a_5, $a'_1 = \left(\begin{smallmatrix} a_4 \\ a_5 \end{smallmatrix}\right)$. And a'_2 is the concatenation of row a_6 and row a_7, $a'_2 = \left(\begin{smallmatrix} a_6 \\ a_7 \end{smallmatrix}\right)$.

$$\begin{aligned}
&\sum_{i=1}^{2n}\Bigg((r_{(I,1)})_i c'_{(1,i)}(x_I + R_y y_I h_i)\\
&+ (r_{(I,2)})_i c'_{(2,i)}(x_I + R_y y_{I_1}\mathsf{H}(h_i))\Bigg)\\
&= \sum_{i=1}^{2n}\Bigg((r_{(I,1)})_i a'_{(1,i)}\\
&+ (r_{(I,2)})_i a'_{(2,i)}\Bigg)(x_I\eta_x + y_I h^*\eta_y)\\
\Leftrightarrow &\sum_{i=1}^{2n}\Bigg(x_I((r_{(I,1)})_i(c'_{(1,i)} - a'_{(1,i)}\eta_x)\\
&+ (r_{(I,2)})_i(c'_{(2,i)} - a'_{(2,i)}\eta_x))\\
&+ y_I((r_{(I,1)})_i(R_y c'_{(1,i)} h_i - a'_{(1,i)}\eta_y h^*)\\
&+ (r_{(I,2)})_i(R_y c'_{(2,i)}\mathsf{H}(h_i) - a'_{(2,i)}\eta_y h^*)\Bigg)\\
&= 0 \qquad (14)
\end{aligned}$$

In Equation (14), factors a' and c' have negligible probability to be function of the variables (i.e. r, x and y), as this would imply the adversary is able to break DL or CDH problem, from the Lemma 4 result. This directly implies that each part of the above sum is independently equal to zero.

We obtain the system of equations:

$$\begin{cases}
c'_{(1,1)} - a'_{(1,1)}\eta_x = 0\\
\qquad\qquad \cdots\\
c'_{(1,2n)} - a'_{(1,2n)}\eta_x = 0\\
c'_{(2,1)} - a'_{(2,1)}\eta_x = 0\\
\qquad\qquad \cdots\\
c'_{(2,2n)} - a'_{(2,2n)}\eta_x = 0\\
R_y c'_{(1,1)} h_1 - a'_{(1,1)}\eta_y h^* = 0\\
\qquad\qquad \cdots\\
R_y c'_{(1,2n)} h_{2n} - a'_{(1,2n)}\eta_y h^* = 0\\
R_y c'_{(2,1)}\mathsf{H}(h_1) - a'_{(2,1)}\eta_y h^* = 0\\
\qquad\qquad \cdots\\
R_{I,y,2n} c'_{2,2n}\mathsf{H}(h_{2n}) - a'_{2,2n}\eta_y h^* = 0
\end{cases} \qquad (15)$$

We write :

- $A'_1 = \begin{pmatrix} a'_{(1,1)} & \dots & 0 \\ & \ddots & \\ 0 & \dots & a'_{(1,2n)} \end{pmatrix}$;
- $A'_2 = \begin{pmatrix} a'_{(2,1)} & \dots & 0 \\ & \ddots & \\ 0 & \dots & a'_{(2,2n)} \end{pmatrix}$;
- $C'_1 = \begin{pmatrix} c'_{(1,1)} & \dots & 0 \\ & \ddots & \\ 0 & \dots & c'_{(1,2n)} \end{pmatrix}$;
- $C'_2 = \begin{pmatrix} c'_{(2,1)} & \dots & 0 \\ & \ddots & \\ 0 & \dots & c'_{(2,2n)} \end{pmatrix}$;
- $D'_1 = \begin{pmatrix} d'_{(1,1)} & \dots & 0 \\ & \ddots & \\ 0 & \dots & d'_{(1,2n)} \end{pmatrix}$
- $D'_2 = \begin{pmatrix} d'_{(2,1)} & \dots & 0 \\ & \ddots & \\ 0 & \dots & d'_{(2,2n)} \end{pmatrix}$;
- $R_y = \begin{pmatrix} R_y & \dots & 0 \\ & \ddots & \\ 0 & \dots & R_y \end{pmatrix}$;
- $H = \begin{pmatrix} h_1 & \dots & 0 \\ & \ddots & \\ 0 & \dots & h_{2n} \end{pmatrix}$; and
- $\mathsf{H}(H) = \begin{pmatrix} \mathsf{H}(h_1) & \dots & 0 \\ & \ddots & \\ 0 & \dots & \mathsf{H}(h_{2n}) \end{pmatrix}$.

We only use diagonal matrices, thus, each matrix is inversible, and the product between two matrices is commutative. We now have the linear system:

$$\begin{cases}
C'_1 - A'_1\eta_x = 0\\
C'_2 - A'_2\eta_x = 0\\
R_y C'_1 H - A'_1\eta_y h^* = 0\\
R_y C'_2\mathsf{H}(H) - A'_2\eta_y h^* = 0
\end{cases} \qquad (16)$$

$$\Leftrightarrow \begin{cases}
C'_1 = A'_1\eta_x\\
C'_2 = A'_2\eta_x\\
R_y C'_1 H = A'_1\eta_y h^*\\
R_y C'_2\mathsf{H}(H) = A'_2\eta_y h^*
\end{cases} \qquad (17)$$

$$\Leftrightarrow \begin{cases}
\frac{1}{\eta_x}C'_1 = A'_1\\
\frac{1}{\eta_x}C'_2 = A'_2\\
R_y C'_1 H = \frac{1}{\eta_x}C'_1\eta_y h^*\\
R_y C'_2\mathsf{H}(H) = \frac{1}{\eta_x}C'_2\eta_y h^*
\end{cases} \qquad (18)$$

We assume all elements on the diagonal of C'_1 and C'_2 are different from 0:

$$\Leftrightarrow \begin{cases}
\frac{1}{\eta_x}C'_1 = A'_1\\
\frac{1}{\eta_x}C'_2 = A'_2\\
R_y H = \frac{1}{\eta_x}\eta_y h^* I\\
R_y\mathsf{H}(H) = \frac{1}{\eta_x}\eta_y h^* I
\end{cases}
\quad \Leftrightarrow \begin{cases}
\frac{1}{\eta_x}C'_1 = A'_1\\
\frac{1}{\eta_x}C'_2 = A'_2\\
\frac{\eta_x}{\eta_y}R_y H = h^* I\\
\frac{\eta_x}{\eta_y}R_y\mathsf{H}(H) = h^* I
\end{cases}$$

$$\Leftrightarrow \mathsf{H}(H) = H \qquad (19)$$

This last equation is impossible with a collision resistant hash function. This means that our hypothesis, is wrong, i.e. $C'_1 = 0$ or $C'_2 = 0$.

Remark 6. We proved here that there is at least one element equal to 0 in C'_1 or C'_2, but we can then reduce the matrices to be of size $2n-1 \times 2n-1$, and have the same result, until C'_1 or C'_2 is totally reduced to 0.

Remark 7. From Theorem 2, we know that η_x and η_y are different from zero. This implies that $(C'_1 = 0) \Rightarrow (A'_1 = 0)$, and $(C'_2 = 0) \Rightarrow (A'_2 = 0)$. From this, we deduce that C'_1 and C'_2 can't be reduced to 0 at the same time, as this would imply that $s_1 = 0$, which is a rejecting condition for the verifier.
Symmetrically, we deduce the same thing from second line of Equation (9):

$$\begin{cases} \frac{1}{\eta_x} D'_1 = B'_1 \\ \frac{1}{\eta_x} D'_2 = B'_2 \\ R_y D'_1 H = \frac{1}{\eta_x} D'_1 \eta_y \mathsf{H}(h^* I) \\ R_y D'_2 \mathsf{H}(H) = \frac{1}{\eta_x} D'_2 \eta_y \mathsf{H}(h^* I) \end{cases} \tag{20}$$

(21)

Thus $D'_1 = 0$ or $D'_2 = 0$, and D'_1 and D'_2 cannot be both reduced to 0 at the same time.

There are four cases to study: $(C'_1, D'_1) = (0,0)$, $(C'_1, D'_2) = (0,0)$, $(C'_2, D'_1) = (0,0)$, and $(C'_2, D'_2) = (0,0)$.

First, $(C'_1, D'_1) = (0,0)$ and $(C'_2, D'_2) = (0,0)$ lead back to $H = \mathsf{H}(H)$, which is impossible.

Then, case $(C'_2, D'_1) = (0,0)$ gives us the system:

$$\begin{cases} \frac{\eta_x}{\eta_y} R_y H = h^* I \\ \frac{\eta_x}{\eta_y} R_y \mathsf{H}(H) = \mathsf{H}(h^* I) \end{cases} \tag{22}$$

If we apply the hash function, we obtain:

$$\mathsf{H}\big(\frac{\eta_x}{\eta_y} R_y H\big) = \frac{\eta_x}{\eta_y} R_y \mathsf{H}(H)$$

There are two cases:

1. $\frac{\eta_y}{\eta_x} = R_y$
2. $\frac{\eta_y}{\eta_x} \neq R_y$

Case 1 tells us that $h^* I = H$. In this case, the signature is a signature on an already signed message. Case 2 implies that the adversary is able to find a collision in a collision resistant hash function. In the Random Oracle Model, this has a negligible probability to occure.

```
EUF − CMA_4 $\mathcal{A}$
$(x_1, ..., x_k) \leftarrow\$ (\mathbb{Z}_p^*)^k$;
$(y_1, ..., y_k) \leftarrow\$ (\mathbb{Z}_p^*)^k$;
$X = (X^{(1)}, ..., X^{(k)}) = (g_2^{x_1}, ..., g_2^{x_k})$
$Y_1 = (Y_1^{(1)}, \cdots, Y_1^{(k)}) = (g_1^{y_1}, \cdots, g_1^{y_k})$
$Y_2 = (Y_2^{(1)}, \cdots, Y_2^{(k)}) = (g_2^{y_1}, \cdots, g_2^{y_k})$
$\bar{X}_1 = (\bar{X}_1^{(1)}, \cdots, \bar{X}_1^{(k)}) = (g_1^{\frac{1}{x_1}}, \cdots, g_1^{\frac{1}{x_k}})$
$\bar{Y}_1 = (\bar{Y}_1^{(1)}, \cdots, \bar{Y}_1^{(k)}) = (g_1^{\frac{1}{y_1}}, \cdots, g_1^{\frac{1}{y_k}})$
$PK = \{X, \bar{X}_1, Y_1, \bar{Y}_1, Y_2\}$
$(\eta_x, \eta_y, R_y, h*, H) \leftarrow\$ \mathcal{A}(PK)$;
$t_1 = \frac{\eta_x}{\eta_y} R_y H$
$t_2 = \frac{eta_x}{\eta_y} R_y \mathsf{H}(H)$
$t_3 = \mathsf{H}(\frac{\eta_x}{\eta_y} R_y \mathsf{H}(H))$
$t_4 = \mathsf{H}(\frac{\eta_x}{\eta_y} R_y H)$
$t_5 = \mathsf{H}(H)$
If $(t_1 = t_3 \vee t_2 = t_4 \vee H = t_5 \vee H = h^* I) \wedge m^* \notin M$:
    return 1;
Else:
    return 0;
```

Fig. 14. EUF-CMA game - reduction

Finally, case $(C'_1, D'_2) = (0,0)$ gives us the system:

$$\begin{cases} \frac{\eta_x}{\eta_y} R_y \mathsf{H}(H) = h^* I \\ \frac{\eta_x}{\eta_y} R_y H = \mathsf{H}(h^*) I \end{cases} \tag{23}$$

If we apply the hash function, we obtain:

$$\mathsf{H}\big(\frac{\eta_x}{\eta_y} R_y \mathsf{H}(H)\big) = \frac{\eta_x}{\eta_y} R_y H$$

This is equivalent for adversary to find a collision in a collision resistant hash function.

The study of the four cases imply that succeeding in $(\mathsf{EUF-CMA}_3\mathcal{A})$ implies that either the adversary is able to find a collision in a collision resistant hash function, or the message signed was already queried to the singing oracle. We represent this as a new game transition. This transition is represented in Figure 14.

The transition between $\mathsf{EUF-CMA}_3\mathcal{A}$ and $\mathsf{EUF-CMA}_4\mathcal{A}$ is a failure based transition, where the failure is equivalent to the probability for the adversary to break Lemma 4.

```
EUF − CMA_5 A
$(x_1, ..., x_k) \leftarrow\!\$\ (\mathbb{Z}_p^*)^k$;
$(y_1, ..., y_k) \leftarrow\!\$\ (\mathbb{Z}_p^*)^k$;
$X = (X_1, ..., X_k) = (g_2^{x_1}, ..., g_2^{x_k})$
$Y_1 = (Y_1^{(1)}, \cdots, Y_1^{(k)}) = (g_1^{y_1}, \cdots, g_1^{y_k})$
$Y_2 = (Y_2^{(1)}, \cdots, Y_2^{(k)}) = (g_2^{y_1}, \cdots, g_2^{y_k})$
$\bar{X}_1 = (\bar{X}_1^{(1)}, \cdots, \bar{X}_1^{(k)}) = (g_1^{\frac{1}{x_1}}, \cdots, g_1^{\frac{1}{x_k}})$
$\bar{Y}_1 = (\bar{Y}_1^{(1)}, \cdots, \bar{Y}_1^{(k)}) = (g_1^{\frac{1}{y_1}}, \cdots, g_1^{\frac{1}{y_k}})$
$PK = \{X, \bar{X}_1, Y_1, \bar{Y}_1, Y_2\}$
$(\eta_x, \eta_y, R_y, h*, H) \leftarrow\!\$\ \mathcal{A}(PK)$;
$t_1 = \frac{\eta_x}{\eta_y} R_y H$
$t_2 = \frac{eta_x}{\eta_y} R_y \mathsf{H}(H)$
$t_3 \leftarrow\!\$\ \{1, \cdots, p-1\}$
$t_4 \leftarrow\!\$\ \{1, \cdots, p-1\}$
$t_5 \leftarrow\!\$\ \{1, \cdots, p-1\}$
If $(t_1 = t_3 \lor t_2 = t_4 \lor H = t_5 \lor H = h^*I)\ \land m^* \notin M$:
    return 1;
Else:
    return 0;
```

Fig. 15. EUF-CMA game - Random Oracle Model

$$
\begin{aligned}
|\Pr[S_3] - \Pr[S_2]| &\leq F \\
&\leq \epsilon_{\text{CDH}}
\end{aligned}
$$

In the Random Oracle Model, we represent the hash function as a Random Oracle. We build a new transition presented in Figure 15. The new game is called $\mathsf{EUF-CMA}_5\mathcal{A}$.

This transition is a representation of $\mathsf{EUF-CMA}_4\mathcal{A}$ in the Random Oracle model. Thus $\Pr[S_5] = \Pr[S_4]$.

We develop the success condition of the game presented in Figure 15:

$$
\begin{aligned}
&(t_1 = t_3 \lor t_2 = t_4 \lor H = t_5 \lor H = h^*I) \land h^* \notin M \\
\Leftrightarrow &(t_1 = t_3 \land h^* \notin M) \lor (t_2 = t_4 \land h^* \notin M) \\
&\lor (H = t_5 \land h^* \notin M) \lor (H = h^*I \land h^* \notin M) \quad (24)
\end{aligned}
$$

We need to analyze each condition individually. The first three conditions are true if the values t_1 or t_2 chosen by the adversary are equal to a number, chosen uniformly at random in $\{1, \cdots, p-1\}$. The probability for this condition to be true is $(p-1)^{-1}$. Furthermore, the last condition is clearly a contradiction. It asks h^* to be at the same time different and equal to some previously signed message. We can represent the condition presented in Equation (24) as:

$$\Leftrightarrow((p-1)^{-1}) \lor ((p-1)^{-1}) \lor ((p-1)^{-1}) \lor (0)$$

We can deduce that, the probability for an adversary to succeed in $\mathsf{Game}_4\mathcal{A}$ is $3 \cdot p^{-1}$. Now, we can deduce the probability for a PPT adversary to forges the signature scheme:

$$
\begin{aligned}
|\Pr[S_5] - \Pr[S_1]| = |\Pr[S_4] - \Pr[S3]| &+ |\Pr[S_3] - \Pr[S_2]| \\
&+ |\Pr[S_2] - \Pr[S_1]| \\
&\leq 3 \cdot \epsilon_{\text{CDH}} + \epsilon_{\text{DL}}
\end{aligned}
$$

And $\Pr[S_5] = 3 \cdot (p-1)^{-1}$. Thus :

$$\Pr[S_1] \leq 3 \cdot (p-1)^{-1} + 3 \cdot \epsilon_{\text{CDH}} + \epsilon_{\text{DL}}$$

Finally, we deduce that :

$$
\begin{aligned}
\mathsf{Adv}_{\Sigma}^{\text{euf-cma}}(\mathcal{A}) &= |\Pr[S_1] - (p-1)^{-1}| \\
&\leq 2 \cdot (p-1)^{-1} + 3 \cdot \epsilon_{\text{CDH}} + \epsilon_{\text{DL}}
\end{aligned}
$$

If the order p is sufficiently large, this value is negligible. Therefore, the adversary has a negligible probability of forging this scheme, even with multiple access to the signing oracles. In conclusion, our signature scheme has EUF-CMA property. □

E.7 Issuer-Indistinguishability proof

In this section, we will prove that the issuer of a randomized signature is indistinguishable among the set of trusted issuers of a malicious verifier. We assume the adversary (in this case, the Malicious Verifier) keeps track of multiple **VerifyRandomized** transactions. We also assume the adversary can collude with *all* the issuers, and therefore knows all their private keys, and all (non randomized) issued signatures.

We define an issuer Indistinguishability game, represented in Figure 16. The adversary succeeds in this game if he is able to distinguish between a re-randomized signature, issued by the issuer I_1, and a signature issued different issuer I_2.

Remark 8. The R subscript is here to allows us to fix the random variables of the algorithm **Randomize**. This concerns the values $r_u, r'_u, r''_u, r_{(u,x)}$, and $r_{(u,y)}$.

```
HiddenIssuerGame$_1\mathcal{A}$
$pp \leftarrow\!\$\ \mathcal{C}(1^\lambda)$
$(\mathsf{sk}_{I_1}, \mathsf{pk}_{I_1})$ =$\mathbf{IssuerKeygen(pp)}$
$(\mathsf{sk}_{I_2}, \mathsf{pk}_{I_2})$ =$\mathbf{IssuerKeygen(pp)}$
$(vpk, vsk) \leftarrow\!\$\ \mathbf{VerifierSetup}(pp, \{I_1, I_2\})$
$m \leftarrow\!\$\ \mathbb{Z}_p^*$
$u \leftarrow\!\$\ \mathbb{Z}_p^*$
$\sigma_1 = \mathbf{Sign}(pp, m, \mathsf{sk}_{I_1}, u)$
$\sigma_2 = \mathbf{Sign}(pp, m, \mathsf{sk}_{I_2}, u)$
$a_1 = (X^{(I_1)'}, Y^{(I_1)'}, \pi_x, \pi_y, \sigma_1') =$
    $\mathbf{Randomize}_R(pp, \mathsf{pk}_{I_1}, vpk,$
                                        $\{I_1, I_2\}, \sigma_1, u)$
$a_2 = (X^{(I_1)''}, Y^{(I_1)''}, \pi_x', \pi_y', \sigma_1'') =$
    $\mathbf{Randomize}'_R(pp, \mathsf{pk}_{I_1}, vpk,$
                                        $\{I_1, I_2\}, \sigma_1, u)$
$b \leftarrow\!\$\ \{0, 1\}$
if $b = 1$:
    $a_2 = (X^{(I_1)'''}, Y^{(I_1)'''}, \pi_x'', \pi_y'', \sigma_1''') =$
        $\mathbf{Randomize}'_R(pp, \mathsf{pk}_{I_2}, vpk,$
                                            $\{I_1, I_2\}, \sigma_2, u)$
$c \leftarrow\!\$\ \mathcal{A}(m, a_1, a_2, \mathsf{sk}_{I_1}, \mathsf{sk}_{I_2}, \mathsf{pk}_{I_1}, \mathsf{pk}_{I_2}, vpk, vsk, \sigma_1, \sigma_2)$
If $c = b$:
    return 1;
Else:
    return 0;
```

Fig. 16. Issuer indistinguishability game, where the randomize algorithm uses as random values the R given in subscript.

Theorem 6. *The signature scheme presented in 6.1.2 is issuer indistinguishable. No* PPT *adversary can win the game presented in Figure 16 with probability greater than $\frac{1}{2} + \epsilon$, where ϵ is a negligible value.*

```
HiddenIssuerGame$_2\mathcal{A}$
$pp \leftarrow\!\$\ \mathcal{C}(1^\lambda)$
$(\mathsf{sk}_{I_1}, \mathsf{pk}_{I_1})$ =$\mathbf{IssuerKeygen(pp)}$
$(\mathsf{sk}_{I_2}, \mathsf{pk}_{I_2})$ =$\mathbf{IssuerKeygen(pp)}$
$(vpk, vsk) \leftarrow\!\$\ \mathbf{VerifierSetup}(pp, \{I_1, I_2\})$
$m \leftarrow\!\$\ \mathbb{Z}_p^*$
$u \leftarrow\!\$\ \mathbb{Z}_p^*$
$\sigma_1 = \mathbf{Sign}(pp, m, \mathsf{sk}_{I_1}, u)$
$\sigma_2 = \mathbf{Sign}(pp, m, \mathsf{sk}_{I_2}, u)$
$a_1 = (X^{(I_1)'}, Y^{(I_1)'}, \pi_x, \pi_y, \sigma_1') =$
    $\mathbf{Randomize}_R(pp, \mathsf{pk}_{I_1}, vpk,$
                                        $\{I_1, I_2\}, \sigma_1, u)$
$a_2 = (X^{(I_1)''}, Y^{(I_1)''}, \pi_x', \pi_y', \sigma_1'', \sigma_2'') =$
    $\mathbf{Randomize}'_R(pp, \mathsf{pk}_{I_1}, vpk,$
                                        $\{I_1, I_2\}, \sigma_1, u)$
$b \leftarrow\!\$\ \{0, 1\}$
if $b = 1$ :
    $a_2 = ((X^{(I_1)''})^{\frac{x_2}{x_1}}, (Y^{(I_1)''}))^{\frac{y_2}{y_1}}, (W_x)_{I_1}^{''\frac{x_1}{x_2}}, (W_y)_{I_1}^{''\frac{y_1}{y_2}},$
        $h_x'', h_y'', \sigma_1^{''\frac{x_2+y_2R_ym}{x_1+y_1R_ym}}, \sigma_2^{''\frac{x_2+y_2R_y\mathsf{H}(m)}{x_1+y_1R_y\mathsf{H}(m)}}, h_1'', h_2'')$
$c \leftarrow\!\$\ \mathcal{A}(m, a_1, a_2, \mathsf{sk}_{I_1}, \mathsf{sk}_{I_2}, \mathsf{pk}_{I_1}, \mathsf{pk}_{I_2}, vpk, vsk, \sigma_1, \sigma_2)$
If $c = b$:
    return 1;
Else:
    return 0;
```

Fig. 17. Issuer indistinguishability game, where the randomize algorithm uses as random values the R given in subscript

The game presented in Figure 16 can be modified to represent the signature built for the second issuer as a modification of the first signature. This modification is given in Figure 17. The transition between these two games is a notation difference. Indeed, the resulting elements are the same in both cases. Therefore, the probability of success of the second game is the same as the probability of success of the first game.

The difference between the two possible signatures given to the adversary only depends on x_1, y_1, x_2, y_2, m, R_y, and the random elements added by the user.

We want to compare the re-randomized signature (case $b = 0$), which we will call σ_0, and the signature with the modified issuer (case $b = 1$), which we will call σ_1.

σ_0 and σ_1 are exclusively composed of elements expressed in $\mathbb{G}_1$ and in $\mathbb{G}_2$. This means that the adversary can combine them using the bilinear pairing e. Therefore, from one randomized signature, the adversary is able to compute 40 different combinations in $\mathbb{G}_T$. We represent all these combinations in Table 11 and in Table 12. These combinations are represented in term of variables (random elements added by the user), and in term of distinguishing elements. It is to say the elements that distinguish a message signed by one issuer, and the other. These variables and distinguishing elements are $r_u, r'_u, r''_u, r_{(u,x)}, r_{(u,y)}$, x_1, y_1, x_2, y_2, m, R, and R_y.

The order of $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ is prime, thus every element of these groups is a generator, except from the neutral element. Therefore, every elements presented in Table 12 will be seen, from the adversary point of view, as elements chosen uniformly at random in one of these groups. This implies, under DL assumption, that a single element cannot be used by an adversary to decide whether the issuer of a credential is I_1 or I_2. Therefore, the only way an adversary could identify the issuer of a signature would be by comparing elements of the signature.

We can distinguish two types of elements in Table 11 and in Table 12: the elements which are function of the variables x_1, y_1, x_2, y_2, and the elements which are not. These variables are the one that the adversary can use to identify the actual issuer of the signature. Using only elements without variables would not allow him to distinguish the issuer, as these elements are equivalent in σ_0 and in σ_1.

In the 40 elements list characterizing the re-randomized signature σ_0, we can see that there are two cells of the tabular that matches with two other cells. These are the elements $e(g_1, h_x) = e(W_x, X^{(1)})$ and $e(g_1, h_y) = e(W_y, Y^{(1)})$. In the σ_1 representation, these matching elements does not depend on the distinguishing variables, and are represented in the same way in σ_0. Furthermore, the aggregator's integrity verification algorithm ensures that the different values $e((W_x)_i, X^{(i)}), \forall i \in \{1, \cdots, k\}$ are equal. This means that the adversary cannot distinguish the issuer using this method.

In these tables, there are two other matching expressions. These are the one that are supposed to match in the **VerifyRandomized** algorithm. But these equations are symmetrical in term of distinguishing values, thus they do not distinguish σ_0 and σ_1. We could think, from the representation we made in Table 11, that $e(\sigma_1, g_2)$ should match with $e(h_1, X^{(I)})$, but σ_1 is factor of an element not represented here for clarity. This element is $(x_1 + R_y y_1 m)$. Indeed, the values match, but the adversary would still need to use the value $e(h_1, Y^{(I)})$ to be able to decide anything.

We can verify, with an algorithm, that each element the adversary as access to in $\mathbb{G}_T$ is distinct from every other one, by at least one random secret element. Or more precisely, there are no linear relationship that links the different element given to the adversary. The only exception are the four cases discussed above, and they do not allow the adversary to distinguish between both cases (i.e. I_1 issued the credential or I_2 issued the credential).

All elements given to the adversary are disjoint in term of the random elements $(R_y, r_{u,x}, r_{u,y}, r_u, r'_u, r''_u, R_y + R)$. $(R_y + R)$ is the element given to the adversary during the **Sign** algorithm. This element can be seen as random in $\mathbb{Z}_p$ because of the addition of R which is only used once. It is a Pedersen commitment, which is totally hiding [41].

We know that the adversary can only distinguish between linear expression of the given elements (Lemma 3). All the elements the adversary has access to are chosen uniformally at random from the adversary point of view, and one by one disjoint in term of random variables. This means that all linear expression of these elements are function of at least one random element. This implies that the adversary cannot distinguish the variation induced by the modification of the issuer unless he is able to break DL assumption or DDH assumption.

Therefore, the adversary is not able to distinguish between σ_0 and σ_1, under DL assumption and DDH assumption. Our scheme is issuer indistinguishable.

Remark 9. The adversary model takes into account potential replay attack. However two randomized signatures use two set of newly generated random elements. From the above proof, we know that a replay attack would give to the adversary elements that would not help the adversary to build linear relationships between the elements. Thus the scheme is also secure against replay attacks.

E.8 Interactive Protocol

In this section, we give proofs of correctness, collision freedom, and element indistinguishability for the interactive version of HIAC discussed in Section 6.2.

	g_2	h_x	h_y	$X^{(1)}$	$Y^{(1)}$
g_1	1	$r_u r_{u,x}$	$R_y r_u r_{u,y}$	r_u	$R_y r_u$
h_1	r'_u	$r_u r'_u r_{u,x}$	$R_y r_u r'_u r_{u,y}$	$r_u r'_u$	$R_y r_u r'_u$
h_2	r''_u	$r_u r''_u r_{u,x}$	$R_y r_u r''_u r_{u,y}$	$r_u r''_u$	$R_y r_u r''_u$
W_x	$r_{u,x}$	$r_u r^2_{u,x}$	$R_y r_u r_{u,x} r_{u,y}$	$r_u r_{u,x}$	$R_y r_u r_{u,x}$
W_y	$r_{u,y}$	$r_u r_{u,x} r_{u,y}$	$R_y r_u r^2_{u,y}$	$r_u r_{u,y}$	$R_y r_u r_{u,y}$
σ_1	$r_u r'_u$	$r^2_u r'_u r_{u,x}$	$R_y r^2_u r'_u r_{u,y}$	$r^2_u r'_u$	$R_y r^2_u r'_u$
σ_2	$r_u r''_u$	$r^2_u r''_u r_{u,x}$	$R_y r^2_u r''_u r_{u,y}$	$r^2_u r''_u$	$R_y r^2_u r''_u$
u	$(R_y + R)$	$r_u r_{u,x}(R_y + R)$	$R_y r_u r_{u,y}(R_y + R)$	$r_u(R_y + R)$	$R_y r_u(R_y + R)$

Table 11. Resulting elements of the rerandomized signature σ_0 after application of the bilinear pairing.

	g_2	h_x	h_y	$X^{(1)}$	$Y^{(1)}$
g_1	1	$r_u r_{u,x}$	$R_y r_u r_{u,y}$	$r_u \frac{x_2}{x_1}$	$R_y r_u \frac{y_2}{y_1}$
h_1	r'_u	$r_u r'_u r_{u,x}$	$R_y r_u r'_u r_{u,y}$	$r_u r'_u \frac{x_2}{x_1}$	$R_y r_u r'_u \frac{y_2}{y_1}$
h_2	r''_u	$r_u r''_u r_{u,x}$	$R_y r_u r''_u r_{u,y}$	$r_u r''_u \frac{x_2}{x_1}$	$R_y r_u r''_u \frac{y_2}{y_1}$
W_x	$r_{u,x} \frac{x_1}{x_2}$	$r_u r^2_{u,x} \frac{x_1}{x_2}$	$R_y r_u r_{u,x} r_{u,y} \frac{x_1}{x_2}$	$r_u r_{u,x}$	$R_y r_u r_{u,x} \frac{y_2}{y_1} \frac{x_1}{x_2}$
W_y	$r_{u,y} \frac{y_1}{y_2}$	$r_u r_{u,x} r_{u,y} \frac{y_1}{y_2}$	$R_y r_u r^2_{u,y} \frac{y_1}{y_2}$	$r_u r_{u,y} \frac{x_2}{x_1} \frac{y_1}{y_2}$	$R_y r_u r_{u,y}$
σ_1	$r_u r'_u \frac{x_2+R_y y_2 m}{x_1+R_y y_1 m}$	$r^2_u r'_u r_{u,x} \frac{x_2+R_y y_2 m}{x_1+R_y y_1 m}$	$R_y r^2_u r'_u r_{u,y} \frac{x_2+R_y y_2 m}{x_1+R_y y_1 m}$	$r^2_u r'_u \frac{x_2}{x_1} \frac{x_2+R_y y_2 m}{x_1+R_y y_1 m}$	$R_y r^2_u r'_u \frac{y_2}{y_1} \frac{x_2+R_y y_2 m}{x_1+R_y y_1 m}$
σ_2	$r_u r''_u \frac{x_2+R_y y_2 \mathsf{H}(m)}{x_1+R_y y_1 \mathsf{H}(m)}$	$r^2_u r''_u r_{u,x} \frac{x_2+R_y y_2 \mathsf{H}(m)}{x_1+R_y y_1 \mathsf{H}(m)}$	$R_y r^2_u r''_u r_{u,y} \frac{x_2+R_y y_2 \mathsf{H}(m)}{x_1+R_y y_1 \mathsf{H}(m)}$	$r^2_u r''_u \frac{x_2}{x_1} \frac{x_2+R_y y_2 \mathsf{H}(m)}{x_1+R_y y_1 \mathsf{H}(m)}$	$R_y r^2_u r''_u \frac{y_2}{y_1} \frac{x_2+R_y y_2 \mathsf{H}(m)}{x_1+R_y y_1 \mathsf{H}(m)}$
u	$(R_y + R)$	$r_u r_{u,x}(R_y + R)$	$R_y r_u r_{u,y}(R_y + R)$	$r_u(R_y + R)\frac{x_2}{x_1}$	$R_y r_u(R_y + R)\frac{y_2}{y_1}$

Table 12. Resulting elements of the signature with modified issuer σ_1 after application of the bilinear pairing.

E.8.1 Correctness

Lemma 5. The interactive version of the aggregator presented in Section 6.2 keeps it correct.

Proof. We want to prove that the value output $(W)'_I$ of the interactive protocol *Commitment reveal exchange* in Figure 5 is the same as the one outputted by the original **Randomize** algorithm, i.e $(W)'_I = g_1^{r_1 \mathsf{sk}(\frac{1}{X_I})}$. We have:

$$
\begin{aligned}
(W)'_I &= (W)''_I \cdot (C''_1)^{-\frac{1}{r_d}} \\
&= (W)''_I \cdot \left((C'_1)^{\frac{1}{r_b}} \cdot (C'_2)^{\frac{1}{r_c}}\right)^{-\frac{1}{r_d}} \\
&= (W)''_I \cdot \left((C_1)^{\mathsf{sk}\frac{1}{r_b} r_d} \cdot (C_2)^{\frac{1}{r_c} r_d}\right)^{-\frac{1}{r_d}} \\
&= (W)''_I \cdot (C_1)^{-\mathsf{sk}\frac{1}{r_b}} \cdot (C_2)^{-\frac{1}{r_c}} \\
&= (W)''_I \cdot g_1^{-r_1 r_b \mathsf{sk}\frac{1}{r_b}} \cdot g_1^{-r_c r_a \frac{1}{r_c}} \\
&= (W)''_I \cdot g_1^{-r_1 \mathsf{sk}} \cdot g_1^{-r_a} \\
&= (W)^{r_1}_I \cdot g_1^{r_a} \cdot g_1^{-r_1 \mathsf{sk}} \cdot g_1^{-r_a} \\
&= g_1^{r_1 \mathsf{sk}(1+\frac{1}{X_I})} \cdot g_1^{r_a} \cdot g_1^{-r_1 \mathsf{sk}} \cdot g_1^{-r_a} \\
&= g_1^{r_1 \mathsf{sk}(\frac{1}{X_I})}
\end{aligned}
$$

□

```
Game$_1\mathcal{A}$
$(x_1, ..., x_k) \leftarrow\$ (\mathbb{Z}_p^*)^k$;
$X_1 = (X_1^{(1)}, ..., X_1^{(k)}) = (g_1^{x_1}, ..., g_1^{x_k})$
$X_2 = (X^{(1)}, ..., X^{(k)}) = (g_2^{x_1}, ..., g_2^{x_k})$
$\bar{X}_1 = (\bar{X}_1^{(1)}, \cdots, \bar{X}_1^{(k)}) = (g_1^{\frac{1}{x_1}}, \cdots, g_1^{\frac{1}{x_k}})$
$(W, \mathsf{sk}) \leftarrow\$ \mathbf{Gen}(X_1,\ X_2,\ \bar{X}_1)$
$C_\sigma^* = ((C_1^*), (C_2^*),$
            $h^*, (W)_I^*, X_2^{(I)*}) \leftarrow\$ \mathcal{A}(X_1, X_2, \bar{X}_1, W)$;
$r_c \leftarrow\$ \mathbb{Z}_p^*$
$(C_1^*)' = (C_1^*)_y^{r_c\mathsf{sk}}$
$(C_2^*)' = (C_2^*)_y^{r_c}$
$(C_1^*)'' \leftarrow \mathcal{A}(C_\sigma^*, \mathrm{pg}, X_1, X_2, (C_2^*)', (C_1^*)')$
If $e((W)_I^*(C_1^*)^{''-(\frac{1}{r_d})}, X_2^{(I)*}) = e(T_1^{\mathsf{sk}}, h^*) \wedge$
   $X_2^{(I)*} \neq g_2^{x_l\eta}, \forall i \in \{1, ..., k\}$:
   return 1;
Else:
   return 0;
```

Fig. 18. Collision freedom game for the interactive protocol of Section 6.2

E.8.2 Collision Freedom

Lemma 6. The interactive version of the aggregator presented in Section 6.2 keeps it collision free.

Proof. The goal is to show that the secret sk value keeps the scheme unforgeable, even if the scheme is modified accordingly to Section 6.2. To do so, we will analyze a modified version of Theorem 2, and use the result of the original theorem, along with the proof that the transformation does not give an advantage to the adversary (in this case, malicious user).

We represent Lemma 6 as a game, presented in Figure 18.

This new game is ultimately equivalent to the game used in Theorem 2. Indeed, there is the same number of unknown (potentially harmful) elements. And the success condition is also equivalent. We only need to analyze the resulting $(W)_I^*(C_1^*)^{''-(\frac{1}{r_d})}$ value.

Firstly, thanks to the proof of knowledge of the elements $(C_1)^*$ and $(C_2)^*$, we know that these elements cannot contain any of the given (Agg, X_1, X_2) under DL assumption. Secondly, under CDH assumption, and because of the r_d value used by the verifier at the end of the protocol, C_1'' must be a linear combination of C_1' and C_2'. Otherwise, the resulting value would depend on r_d, and this value did not even exist when the adversary computed the values $(\mathsf{Comm}'(s)(W)_I'', \pi_s)$. Thus if $(W)_I^*(C_1^*)^{''-(\frac{1}{r_d})}$ depends on r_d then the probability for the verification algorithm to output 1 is negligible (in fact, this probability is p^{-1}). Always under CDH assumption, C_1'' only contains sk, r_c and elements known to the adversary. This means that we can represent $(W)_I^*(C_1^*)^{''-(\frac{1}{r_d})}$ as $g_1^{s_w} g_1^{s_1\mathsf{sk}} g_1^{s_2}$, with s_1 and s_2 value known to the adversary.

We use the same idea as in Appendix E.3, we know that $g_1^{s_w} g_1^{s_1\mathsf{sk}} g_1^{s_2} = g_1^{\mathsf{sk}\gamma}$, for some γ independent from sk. At this point, the only difference between the game presented in Figure 18 and Appendix E.3's proof is that the adversary is given the extra values $g_1^{skr_d}$ and $g_1^{r_d}$. We use Appendix E.3 proof again and more precisely Appendix E.3.2 proof. The setup we have here is exactly the same as the one used in the simplified game of this proof. Thus the conclusion is the same, i.e. γ is a composition of $(W)_i$ values.

We can resume Appendix E.3's proof without further modification, we will get the same result, i.e. $X_2^{(I)*} = g_2^{x_l\eta}$. □

E.8.3 Indistinguishability of the Signature with Commitment Reveal Exchange

Lemma 7. The interactive version of the aggregator presented in Section 6.2 keeps it element indistinguishable.

Proof. We want to prove that the indistinguishability property is not affected by the modification made in Section 6.2. This transformation can be proven to keep issuer indistinguishability property and unlinkability property, using the theorem proved in Appendix E.7.

Indeed, the transformation makes the user share three more values, C_1, C_2, and C_1''. We will prove these values indistinguishable in the same way we did in Appendix E.7. We know that $C_1 = g_1^{r_1 r_b}$ and $C_2 = g_1^{r_a r_c}$. From the ZKP ϕ_2 , we also know that $C_1' = C_1^{s_1}$ and $C_2' = C_2^{s_2}$, for some unknown values s_1 and s_2. This implies that $C_1'' = C_1^{\frac{1}{r_b}s_1} C_2^{\frac{1}{r_c}s_2} = g_1^{r_{(u,x)}s_1 + r_a s_2}$. Any new request from the adversary would lead to a new commitment value, with new random values. This implies that the adversary has no advantage in requesting multiple commitments.

The values disclosed by C_1 and C_2 are randomized with elements used only once. The adversary cannot extract any information from them. The value disclosed by C_1'' can be considered as a value chosen uniformly at random in $\mathbb{G}_1$ because of the addition of r_a, used only there. Thus, C_1'' does not disclose information about the issuer nor the user. There is no linear relationship between C_1, C_2, C_1'' and the other values of the scheme. Thus the modification made in Section 6.2 keeps the indistinguishability property. □