Aggregate Signatures with Versatile Randomization and Issuer-Hiding Multi-Authority Anonymous Credentials

Omid Mir*
Johannes Kepler University Linz
LIT Secure and Correct Systems Lab
Linz, Austria
mir@ins.jku.at

Balthazar Bauer
IRIF, CNRS
Paris, France
Balthazar.Bauer@ens.fr

Scott Griffy
Brown University
Providence, USA
scott_griffy@brown.edu

Anna Lysyanskaya
Brown University
Providence, USA
anna_lysyanskaya@brown.edu

Daniel Slamanig
AIT Austrian Institute of Technology
Vienna, Austria
daniel.slamanig@ait.ac.at

ABSTRACT

Anonymous credentials (AC) offer privacy in user-centric identity management. They enable users to authenticate anonymously, revealing only necessary attributes. With the rise of decentralized systems like self-sovereign identity, the demand for efficient AC systems in a decentralized setting has grown. Relying on conventional AC systems, however, require users to present independent credentials when obtaining them from different issuers, leading to increased complexity. AC systems should ideally support being multi-authority for efficient presentation of multiple credentials from various issuers. Another vital property is issuer hiding, ensuring that the issuer's identity remains concealed, revealing only compliance with the verifier's policy. This prevents unique identification based on the sole combination of credential issuers. To date, there exists no AC scheme satisfying both properties simultaneously.

This paper introduces Issuer-Hiding Multi-Authority Anonymous Credentials (IhMA), utilizing two novel signature primitives: Aggregate Signatures with Randomizable Tags and Public Keys and Aggregate Mercurial Signatures. We provide two constructions of IhMA with different trade-offs based on these primitives and believe that they will have applications beyond IhMA. Besides defining the notations and rigorous security definitions for our primitives, we provide provably secure and efficient constructions, and present benchmarks to showcase practical efficiency.

CCS CONCEPTS

• **Security and privacy → Cryptography**; **Privacy-preserving protocols**.

*First and corresponding author; remaining authors in alphabetical order.

CCS '23, November 26–30, 2023, Copenhagen, Denmark

ACM ISBN 979-8-4007-0050-7/23/11.
https://doi.org/10.1145/3576915.3623203

KEYWORDS

Aggregate Signatures, Anonymous Credentials, Multi-Authority, Issuer-Hiding, Equivalence-class signatures, Mercurial Signatures

ACM Reference Format:
Omid Mir, Balthazar Bauer, Scott Griffy, Anna Lysyanskaya, and Daniel Slamanig. 2023. Aggregate Signatures with Versatile Randomization and Issuer-Hiding Multi-Authority Anonymous Credentials. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security (CCS '23), November 26–30, 2023, Copenhagen, Denmark.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3576915.3623203

1 INTRODUCTION

Authentication and authorization are essential and security-critical tasks in a digital world. They are aimed to ensure that the communication partner is the one it claims to be and to enforce access control to digital resources such as services. A central concept is that of a digital identity, which can be seen as a collection of attributes (e.g., name, age, nationality, gender, etc.) representing a (real-world) entity in the digital realm.

On the Internet, a widely adopted practice is to have centralized identity providers (IdP), e.g., Google or Meta, to maintain the digital identity of users. Other services can then simply rely on the identity provided by the IdP. From a privacy perspective, however, this is problematic as users lose control over their digital identity (all their attributes reside at the IdP), and the IdP learns all the services a user consumes on the Internet (and data related to the use).

Already in the 1980s, Chaum [24, 25] envisioned cryptographic techniques for creating more privacy-friendly and user-centric solutions to authentication and authorization. They put users in control of their identity and allow users to selectively reveal information (i.e., attributes) about their digital identities in an unlinkable and thus untraceable way. Such techniques are commonly known as anonymous credentials (ACs), and there is a vast body of research into different approaches to construct such AC systems [2, 3, 12, 17–20, 27, 32, 35, 39, 49, 52, 54].

While early AC systems such as U-Prove [51] and Idemix [22] did not see a widespread adoption, nowadays related techniques such as direct anonymous attestation (DAA) [14, 15] and Enhanced Privacy ID (EPID) [13] are deployed in billions of devices. Most recently, ACs have seen adoption within the popular Signal messenger to realize private groups [23]. They also see increasing popularity in

the form of anonymous tokens (with private or public metadata bit) [31, 42, 56]. Among the applications are private browsing with DDoS protection being standardized by the IETF[1] (Privacy Pass [31] and Private Access Tokens [45]) or the PrivateStats by Facebook[2] to privately collect client-side telemetry from WhatsApp.

Decentralized identity. Like with centralized IdPs, all AC solutions mentioned so far are in a centralized setting, i.e., a single party called the issuer is issuing credentials to users. Today we however see a trend to move away from this centralized setting towards a decentralized identity. A popular concept in the decentralized identity space is that of self-sovereign identity (SSI) with Sovrin[3] being a prominent example. In SSI users are collecting certified attributes (called verifiable credentials) from *different sources* and then presenting (subsets of) verifiable credentials from this collection. There is an increasing push towards standardization of this verifiable credentials concept within W3C[4] and large efforts such as the future European data infrastructure (Gaia-X)[5] or the European Blockchain Services Infrastructure (EBSI)[6] are adopting this approach.

Within the verifiable credential initiative in W3C, it is also observed that privacy related features are important. In particular well-known features from AC systems such as supporting selective disclosure and proving predicates about attributes[7]. To realize this functionality within W3C it is intended to base this upon the BBS+ signature scheme[8], a well-known building block for ACs currently being standardized as the BBS variant [58] within the IETF[9].

Privacy in a decentralized setting. The aforementioned approach allows to preserve privacy in a setting where a user wants to show a single verifiable credential issued by a single party. However, for a decentralized setting, where typically a subset of a collection of verifiable credentials from different issuers needs to be shown, the problem of how to efficiently realize this arises. A naive way is to conduct a parallel credential showing with all the required verifiable credentials. However, apart from reduced efficiency, this also has privacy implications. In particular, every verifiable credential reveals the exact issuer providing a lot of contextual partial information, e.g., a passport issued from a certain country or a driving license issued by a certain state reveals geographic information. This can be highly privacy intrusive in many settings and undermining the very objective of SSI systems [10]. Consequently, it would be desirable to be able to show a credential in a way that it is only revealed that it comes from one of a larger set of issuers acceptable by a verifier. A set of recent independent works introduced a property providing this features for AC systems, which is called issuer-hiding [6, 10, 27]. While this is a step towards countering the above privacy issues, these works only consider single issuers and are thus not yet suitable for a decentralized setting with multiple issuers.

[1]https://datatracker.ietf.org/wg/privacypass/about/
[2]https://research.fb.com/privatestats
[3]https://sovrin.org/
[4]https://www.w3.org/TR/vc-data-model/
[5]https://gaia-x.eu/
[6]https://ec.europa.eu/digital-building-blocks/wikis/display/EBSI/Home
[7]https://www.w3.org/TR/vc-data-model/#privacy-considerations
[8]https://w3c-ccg.github.io/ldp-bbs2020/
[9]https://datatracker.ietf.org/doc/draft-irtf-cfrg-bbs-signatures/

ACs in a decentralized setting. Due to not being directly comparable as they are either only threshold or require a dedicated infrastructure (i.e., a transparency log, Byzantine system, or a blockchain) and TDAC by [49] and the lack of space we defer to [47] for a discussion of existing approaches in a decentralized setting due to Garman et al. [36], Sonnino et al. [57], Doerner et al. [33] and Rosenberg et al. [53].

Finally, and most related, we want to discuss the work by Hébant and Pointcheval [40]. The authors introduced the concept of (traceable) Multi-Authority Anonymous Credentials (MA-ACs). Loosely speaking, their approach to realize MA-ACs is based on so called aggregate signatures with randomizable tags and allows to aggregate showings of credentials of different issuers (but with respect to the same tag) into one compact showing. Due to randomizability of signatures and tags, it is possible to produce unlinkable showings. Moreover, the tag component has a secret part representing the user secret. While this is an interesting concept, it does not provide an efficient way of providing the issuer-hiding (IH) feature [6, 10, 27]. There is an obvious generic way to use a succinct NIZK (i.e., a zk-SNARK) and prove that the aggregated signature verifies for the given attributes under a subset of issuer keys without revealing which ones. While this can lead to an asymptotically compact solution, the prover will concretely be very expensive due to the size of the verification keys (they are of size $\mathbb{G}_2^{3+2n}$ each with n being the maximum number (types) of attributes) and the complexity of the verification equation in [40] which is proven with a zk-SNARK. Switching to non-succinct Schnorr-type NIZK obtained via Fiat-Shamir as done in [6] (in Construction 2), however, will result in a non-compact showing of size $O(n \cdot K)$ with K being the number of issuers used in the aggregated showing (even when ignoring the size of the proof corresponding to the non-shown attributes).

In this paper, our goal is to efficiently combine these features and propose the first AC system that is specifically designed to provide multi-authority and issuer-hiding features at the same time.

Aggregate signatures. Aggregate signatures, introduced by Boneh et al. in [9], allow to combine multiple signatures σ_i for messages m_i and associated public keys vk_i into a single signature σ, that authenticates the entire set of messages w.r.t the set of public keys. Ideally, the aggregated signatures is of length identical to a single signatures and thus allows to compress a set of signatures into a single one.

This primitive is valuable in optimizing storage and bandwidth and minimizing cryptographic overhead in scenarios such as compressing certificate chains or aggregating signatures in blockchains. Many different variants have been proposed [4, 8, 38, 50] and we will briefly mention some relevant schemes. Sequential aggregation, studied in [44], requires signers to interact sequentially. Synchronized aggregation, examined in [1], assumes synchronization among signers such that in every time period t each signer only contributes one signature at most. Indexed or tag-based aggregated signatures, introduced in [40], allow aggregation of signatures for different messages under different public keys if they share the same tag or index. These signatures are useful for constructing an AC system.

Unfortunately, existing aggregate signature schemes do not explicitly possess properties to make them amenable for the design of

efficient decentralized AC systems with advanced properties. We will close this gap by introducing aggregate (structure-preserving) signatures with the ability to randomize signatures, tags, (messages,) and verification keys.

1.1 Our Contribution

Our contribution in this paper is twofold:

Aggregate signatures with randomization features. The key technique to achieve our goal is to introduce tag-based aggregate signatures with randomizable tags and public keys. We further extend them to additionally support randomization of messages resembling the functionality of equivalence class signatures (SPSEQ) [35]. For both of these types of schemes we provide rigorous formal security models as well as instantiations that are provably secure in this model. More precisely, we introduce:

Aggregate signatures with randomizable keys and tags (AtoSa[10] for short) where signatures are associated to tags (consisting of a private and a public part) and signatures with respect to the *same* tag can be aggregated. Aside from signatures, verification keys and tags can be randomized. Tags and verification keys are defined with respect to equivalence classes and randomization switches between representatives of these classes.[11] Then existing signatures can be adapted to ones that verify under the randomized public keys and tags. We provide an AtoSa scheme based on the well-known Pointcheval-Sanders (PS) signatures [52]. PS signatures have already served as a basis for various privacy-preserving primitives such as group signatures and anonymous credentials [52], redactable [54, 55] or dynamically malleable signatures [5]. They are very efficient and have interesting features such as support for blind signing, i.e., signing of committed (hidden) messages, and efficient ways of proving their knowledge.

Aggregate Mercurial Signatures with Randomizable Tags (ATMS) extend the functionality of AtoSa to support the randomization of messages, i.e., equivalence classes of messages similar to (SPSEQ). This means that in addition to AtoSa existing signatures can be adapted to verify under randomized messages (i.e., other representatives of the message class). Consequently, we obtain a version of mercurial signatures [30] that is both aggregatable and has randomizable tags. To the best of our knowledge, this is the first instance of an aggregate structure-preserving signature (and, additionally the first aggregatable SPSEQ). We provide an ATMS construction inspired by the message-indexed SPS in [29], which on itself is a variant of Ghadafi's SPS [37] scheme.

Restrictions of our Constructions. We should mention that in contrast to standard aggregate signatures, our constructions 1) either require that all aggregated messages and corresponding verification keys are known before requesting the first signature or 2) to make the same assumption as within synchronized aggregate signatures [1, 41]. In particular, adapted to our setting, latter means that every issuer ensures that for each tag only a single signature is issued. We will present our results based on the first approach and discuss adaptions for the second (which do not change any of the interfaces or security definitions and proofs). Since our main application is anonymous credentials, depending on the concrete application scenario either the first or the second approach can be chosen. It remains an interesting open question to get fully dynamic signatures without any of the above assumptions.

Like other types of signatures with randomization features, we also expect that our schemes will find applications beyond the one presented here.

Issuer-Hiding Multi-Authority Anonymous Credentials. We present a rigorous formal model for issuer-hiding multi-authority anonymous credentials (IhMA). Then we present two constructions based on AtoSa (called $\mathsf{IhMA}_{\mathsf{AtoSa}}$) and ATMS (called $\mathsf{IhMA}_{\mathsf{ATMS}}$) respectively, where both are concretely very efficient but offer some trade-offs (as discussed below). Thus this represents an important contribution to the field of ACs in that it provides a solution that addresses the challenges of user privacy and scalability in multi-authority (decentralizing) settings. In our constructions, obtaining a credential amounts to obtaining signatures on desired attributes from a set of issuers on different attributes, but under the same tag (which can be thought of as the user's identity in credential schemes). Showing simply amounts to randomizing signatures from issuers that should be shown as well as the tags and aggregating them. Finally, one provides the aggregated signature and either opens (subsets of) attributes or proves predicates over them along with proof of knowledge of the secret tag part.

Supporting the issuer-hiding feature [7, 27] works roughly as follows: Each verifier generates a so-called *key-policy*, which defines a set of issuers (via their verification keys) that the verifier would accept an (aggregated) credential from. This policy is a collection of SPSEQ signatures on verification keys of the AtoSa or ATMS scheme. Since the equivalence classes of the SPSEQ (the message space) match with the key equivalence class of AtoSa and ATMS, showing a credential then works as above, but all verification keys of the AtoSa or ATMS are randomized, and the respective SPSEQ signatures in the key-policy are adapted accordingly.

For the $\mathsf{IhMA}_{\mathsf{ATMS}}$ scheme, instead of directly signing attributes, we use the framework of Fuchsbauer et al. [35]. Here the signature scheme is used to sign set commitments to attribute sets. Moreover, in order to prove the anonymity of this construction as an additional contribution we introduce a generalization of the decisional uber assumption family by Boyen [11] along with an interactive version. Using this approach is however not straightforward as we have to make set commitments compatible with the message space of our ATMS. While $\mathsf{IhMA}_{\mathsf{AtoSa}}$ and $\mathsf{IhMA}_{\mathsf{ATMS}}$ share a common aim, the differences in constructions entail certain trade-offs in terms of functionality and efficiency:

- Credential size: The $\mathsf{IhMA}_{\mathsf{ATMS}}$ scheme can yield a fixed-sized credential, while the $\mathsf{IhMA}_{\mathsf{AtoSa}}$ scheme does not achieve this without utilizing Zero Knowledge Proof of Knowledge (ZKPOK) of signatures.
- Efficiency: The $\mathsf{IhMA}_{\mathsf{ATMS}}$ scheme is more efficient at showing and verifying credentials compared to the $\mathsf{IhMA}_{\mathsf{AtoSa}}$ scheme.
- Need for a trusted party: The $\mathsf{IhMA}_{\mathsf{ATMS}}$ scheme requires a trusted party, while the $\mathsf{IhMA}_{\mathsf{AtoSa}}$ scheme does not. This is

[10] The (ancient) Greek transliteration of the old Persian name Utauθa. Atossa means "bestowing very richly" or "well trickling" or "well granting". It refers to an Achaemenid empress who was the daughter of Cyrus the Great, and the wife of Darius the Great.
[11] This can be seen as aggregate signatures with randomizable tags as introduced in [40] with the additional features of randomizable keys with appropriate signature adaption.

because $\mathsf{IhMA}_{\mathsf{ATMS}}$ relies on a trusted party to hold a trapdoor to generate set commitments, whereas $\mathsf{IhMA}_{\mathsf{AtoSa}}$ does not require such a trusted party.
- Expressiveness: The $\mathsf{IhMA}_{\mathsf{ATMS}}$ supports revealing a subset of attributes from a set of attributes per issuer, i.e., selective disclosure per issuer. The $\mathsf{IhMA}_{\mathsf{AtoSa}}$ scheme only supports a single attribute for each credential. Consequently, it only supports selective disclosure over all issuers. However, both schemes allow for proving arbitrary predicates over signed messages.

Overall, the choice of the concrete construction depends on the specifics of the use case or application and priorities set in the overall system.

1.2 Comparison of IhMA with Previous Work

We have already discussed that there is only one dedicated MA-AC scheme [40]. This is however not issuer-hiding (IH) and as mentioned, adding IH comes with a significant overhead. In Table 1, we compare our IhMA approaches to other schemes in the literature that provide the IH feature [6, 10, 27] and for comparison we use the naive approach to achieve MA, i.e., parallel showings of single credentials, which we indicate by $\approx$. We compare them in terms of the size of credential $|\mathbf{Cred}|$, communication cost of showing $|\mathbf{Show}|$, and computational cost of showing **Show** for user **(P)** and verifier **(V)**. We provide concrete analysis for our schemes' communication cost in our full version [47]. To ensure a fair comparison between the schemes, we consider a typical case of k out of n attributes from K out of N issuers where n is the total number of attributes given to the user by N issuers, and k is the number of attributes involved in the showing (and K the number of issuers involved).

With respect to credential size $|\mathbf{Cred}|$, the naive approach to **MA** leads to $O(K)$ complexity. Our $\mathsf{IhMA}_{\mathsf{ATMS}}$ scheme maintains a constant credential size even when there are $K > 1$ issuers, while our $\mathsf{IhMA}_{\mathsf{AtoSa}}$ scheme has $O(K)$ credentials. However, we can aggregate credentials and then during showing apply a ZKPOK of a PS signature, which allows us to reduce the credential size to a constant size. In contrast, others have a credential size linear in the number of issuers K.

In terms of communication cost in showing ($|\mathbf{Show}|$), our schemes require sending the randomized vks of the K issuers, along with two signatures (one for the credential and one for the key policy), overall giving $O(K)$. In [6], the communication size is based on sending K blinded credentials and K blinded signatures in the key policy and provide a ZKPOK of having correctly done so. The scheme in [10] is similar to [6], but the size of the policy is fixed. Finally, in the scheme described in [27], one needs to prove knowledge of K out of N verification keys (a linear sized OR statement) and sends them along with K credentials. Note that the size of ZKPOK includes many group elements and significantly more than only transferring K verification keys, as it is the case for our constructions.

When it comes to the computational cost of showing, i.e., **Show (P)** and **Show (V)**, our $\mathsf{IhMA}_{\mathsf{AtoSa}}$ scheme has a minimal computational cost for provers as they only need to perform a small/constant number of operations for aggregation, along with K exponentiations for randomizing the verification keys vk. Our $\mathsf{IhMA}_{\mathsf{ATMS}}$ scheme involves additional computation in the creation of a witness for set commitments corresponding to undisclosed attributes (a multi-exponentiation of $O(u)$). In [6], this cost includes proving knowledge of k signatures (in the key policy), K credentials, and k disclosed attributes. Similarly, [10] requires the computation of generating witness for their aggregator (accumulator) on K credentials, proving knowledge of k credential, but it does not need to prove knowledge of signatures in the policy. Moreover, in [27], proving knowledge of K-out-of-N verification keys is necessary, along with the computation of generating witness on undisclosed attributes for set commitments on K credentials. Again, the cost of ZKPOK for credentials or committed attributes is significantly more expensive than in our case, which is needed only to prove a secret key and some multi-exponentiation for creating witness. We should mention here that by leveraging ZKPOK, arbitrary relationships can be proved on attributes.

In summary, while the efficiency of different schemes may appear to be close asymptotically, our IhMA approaches are significantly more efficient than existing approaches while providing both properties simultaneously. Indeed, we only need $O(k)$ group operations in $\mathbb{G}_i$. In contrast, other schemes require proving knowledge of signatures or keys, which is significantly more expensive.

2 PRELIMINARIES

Notation. We use $\mathsf{BG} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, P, \hat{P}) \leftarrow \mathsf{BGGen}(1^\lambda)$ to denote a bilinear group generator for asymmetric type 3 groups, where p is a prime of bitlength λ. When applying a scalar a componentwise to a vector $\mathbf{T} \in \mathbb{G}_1^n$ we write $\mathbf{T}^a = (T_1^a, T_2^a, \ldots, T_n^a)$. We write $[x]_{\mathcal{R}}$ to denotes denote representative x of the equivalence class for given relation $\mathcal{R}$. Given a finite set S, we denote by $x \leftarrow S$ or $x \xleftarrow{\$} S$ the sampling of an element uniformly at random from S. For an algorithm A, let $y \leftarrow \mathsf{A}(x)$ be the process of running A on input x with access to uniformly random coins and assigning the result to y. With $\mathcal{A}^{\mathsf{B}}$ we denote that $\mathcal{A}$ has oracle access to B. We use $\langle O \rangle$ to denote oracles defined in games and use ϵ to indicate a negligible function. We assume all algorithms are polynomial-time (PPT) unless otherwise specified and public parameters are an implicit input to all algorithms in a scheme.

Indexed Diffie-Hellman Message Space $\mathcal{M}^H_{\mathsf{iDH}}$ [29]. Given a bilinear group $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e, g, \hat{g}) \leftarrow \mathsf{BGGen}(1^\lambda)$, an index set $\mathcal{I}$, and a random oracle $H : \mathcal{I} \rightarrow \mathbb{G}_1$, $\mathcal{M}^H_{\mathsf{iDH}}$ is an indexed Diffie-Hellman (DH) message space if $\mathcal{M}^H_{\mathsf{iDH}} \subset \{(id, \tilde{M}) \mid id \in \mathcal{I}, m \in \mathbb{Z}_p, \tilde{M} = (H(id)^m, \hat{g}^m) \in \mathbb{G}_1 \times \mathbb{G}_2\}$ and the following index uniqueness property holds: for all $(id, \tilde{M}) \in \mathcal{M}^H_{\mathsf{iDH}}$, $(id', \tilde{M}') \in \mathcal{M}^H_{\mathsf{iDH}}$, $id = id' \Rightarrow \tilde{M} = \tilde{M}'$. One can define the equivalence class for each message $\tilde{M} = (M, N) \in \tilde{\mathcal{M}}^H_{\mathsf{iDH}}$, as $\mathsf{EQ}_{\mathsf{iDH}}(M, N) = \{(M^r, N) \mid \exists\, r \in \mathbb{Z}_p\}$. One can efficiently decide subset membership by checking $e(M, \hat{P}) = e(h, N)$. The uniqueness property guarantees that no two messages use the same index, which needs to be ensured by signers. We use the Camenisch-Stadler notation [21] for ZKPOK. Please refer to the full version for complete definitions [47].

3 AGGREGATE SIGNATURES WITH RANDOMIZABLE KEYS AND TAGS

Now we introduce a novel primitive named AtoSa where one can aggregate signatures of different messages under different keys

Table 1: Comparison of AC schemes in MA setting (n: Attributes; k: Disclosed attributes, u: Undisclosed attributes, N: Total issuers in policy, K: issuers in showing)

	[27] ‡	[10]★★	[6]★★	IhMA$_{\text{AtoSa}}$	IhMA$_{\text{ATMS}}$
IH	✓	✓	✓	✓	✓
MA	≈	≈	≈	✓	✓
|Cred|	$O(N)$	$O(N)$	$O(N)$	$O(N)^{\star}$	$O(N)^{\star}$
|Show|	$O(K \cdot N)$	$O(k \cdot K)$	$O(k \cdot 2K)$	$O(K)$	$O(K)$
Show (P)	$O(KuN)$	$O(k \cdot K)$	$O(k \cdot 2K)$	$O(K)^{\dagger}$	$O(u \cdot K)$
Show (V)	$O(KkN)$	$O(k \cdot K)$	$O(k \cdot 2K)$	$O(k)$	$O(k \cdot K)$

★ We present the scheme in a way that supports ad-hoc attribute/issuer aggregation, but for fixed signatures, a constant size credential is achievable. For ATMS we will show how to achieve this in Section 5.2.
★★ K refers to proving knowledge of K credentials and K signatures of key policy in Showing.
† Since the ad-hoc aggregation cost is negligible, it is skipped here. Also, without considering IH, it becomes $O(1)$.
‡ This scheme uses standard assumptions in the ROM while other schemes use the GGM.

only if they are associated with the same tag (consisting of a private and a public part). Moreover, apart from allowing randomizing signatures, verification keys as well as tags can be randomized. Unlike mercurial signatures, our AtoSa scheme does not allow for randomization of messages. Tags and verification keys are defined with respect to equivalence classes and randomization switches between representatives of these classes. We introduce a comprehensive formal model and a construction which as a starting point takes PS signatures [52]. For our AtoSa scheme we show how to integrate tags into PS signatures, use the above discussed features to make them aggregatable, and show that the key-randomization features of PS signatures (cf. [26] with $\Delta_2 = 0$) applies to our modification.

3.1 Formal Definitions

The public key randomization is similar to that of mercurial signatures [30], which allow to define equivalence classes on the key space $[\mathsf{vk}]_{\mathcal{R}_{\mathsf{vk}}}$, $[\mathsf{sk}]_{\mathcal{R}_{\mathsf{sk}}}$. Let a tag be $(\tau, \mathbf{T})$, where τ and $\mathbf{T}$ are the secret and public parts of tag respectively. For the tag randomization, we define equivalence classes $[\mathbf{T}]_{\mathcal{R}_\tau}$ ($[\tau]_{\mathcal{R}_\tau}$ for secret parts) on the tag space $\mathcal{T}$ similar to $[\mathsf{vk}]_{\mathcal{R}_{\mathsf{vk}}}$ and $[\mathsf{sk}]_{\mathcal{R}_{\mathsf{sk}}}$ as:

$$\mathcal{R}_\tau = \begin{Bmatrix} (\mathbf{T}', \mathbf{T}) \in (\mathbb{G}_1^*)^\ell \times (\mathbb{G}_1^*)^\ell \mid \exists \mu \in \mathbb{Z}_p^* : \ \mathbf{T}' = \mathbf{T}^\mu \\ (\tau', \tau) \in (\mathbb{Z}_p^*)^\ell \times (\mathbb{Z}_p^*)^\ell \mid \exists \mu \in \mathbb{Z}_p^* : \ \tau' = \tau \cdot \mu \end{Bmatrix}$$

We denote the space of all tags as $\mathcal{T}$ and the messages space is $\mathbb{Z}_p$. In contrast to SPSEQ (and mercurial) signatures, we do not consider equivalence classes on the message space for AtoSa.

*Definition 1 (Aggregate Signatures with Randomizable Public Keys and Tag (*AtoSa*)).* An AtoSa for parameterized equivalence relations $\mathcal{R}_\tau$, $\mathcal{R}_{\mathsf{sk}}$ and $\mathcal{R}_{\mathsf{vk}}$, consists of the following algorithms:

- $\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$: On input the security parameter λ, output the public parameters pp.
- $\mathsf{KeyGen}(\mathsf{pp}) \to (\mathsf{sk}, \mathsf{vk})$: On input the public parameters pp, output a key pair $(\mathsf{sk}, \mathsf{vk})$.
- $\mathsf{VKeyGen}(\mathsf{sk})$: On input a secret key sk, output a verification key vk.
- $\mathsf{GenAuxTag}(S) \to (\{\mathsf{aux}_j\}_{j\in[n]}, (\tau, \mathbf{T}))$: Given a message-key set $S = \{(m_j, \mathsf{vk}_j)_{j\in[n]}\}$, output auxiliary data $\{\mathsf{aux}_j\}_{j\in[n]}$ correlated to (vk_j, m_j) and a tag pair $(\tau, \mathbf{T})$, where all vk_j should be distinct.
- $\mathsf{Sign}(\mathsf{sk}_j, \tau, \mathsf{aux}_j, m_j) \to \sigma_j$: On input a secret key sk_j, tag's secret τ, auxiliary data aux_j and message $m_j \in \mathbb{Z}_p$, output a signature σ_j for $(\tau, \mathbf{T})$ and m_j under the verification key vk_j.
- $\mathsf{Verify}(\mathsf{vk}_j, \mathbf{T}, m_j, \sigma_j) \to \{0, 1\}$: Given a verification key vk_j, tag's public $\mathbf{T}$, message m_j and signature σ_j, output 1 if σ_j is valid relative to vk_j, m_j and $\mathbf{T}$, and 0 otherwise.
- $\mathsf{AggrSign}(\mathbf{T}, \{(\mathsf{vk}_j, m_j, \sigma_j)\}_{j=1}^{\ell}) \to \sigma$: Given ℓ signatures, $(\sigma_j)_{j\in[\ell]}$ for messages $(m_j)_{j\in[\ell]}$ under verification keys, $(\mathsf{vk}_j)_{j\in[\ell]}$ on the same tag $\mathbf{T}$, output an aggregate signature σ on all messages $\mathbb{M} = (m_j)_{j\in[\ell]}$ under the tag $\mathbf{T}$ and aggregated verification key $\mathsf{avk} = (\mathsf{vk}_j)_{j\in[\ell]}$.
- $\mathsf{VerifyAggr}(\mathsf{avk}, \mathbf{T}, \mathbb{M}, \sigma) \to \{0, 1\}$: Given an aggregated verification key avk, tag $\mathbf{T}$, messages $\mathbb{M}$ and signature σ, output 1 if σ is valid relative to avk, $\mathbb{M}$ and $\mathbf{T}$, and 0 otherwise.
- $\mathsf{ConvertTag}(\mathbf{T}, \mu) \to \mathbf{T}'$: On input a tag $\mathbf{T}$ and randomness μ, output a new randomized tag $\mathbf{T}' \in [\mathbf{T}]_{\mathcal{R}_\tau}$.
- $\mathsf{RndSigTag}(\mathsf{vk}, \mathbf{T}, m, \sigma, \mu) \to (\sigma', \mathbf{T}')$: (Randomize Signature and Tag together) Given a signature σ on a message m under tag $\mathbf{T}$ and vk, and randomness μ. Return a randomized signature and tag $(\sigma', \mathbf{T}')$ s.t $\mathsf{Verify}(\mathsf{vk}, \mathbf{T}', m, \sigma') = 1$, where $\mathbf{T}' \leftarrow \mathsf{ConvertTag}(\mathbf{T}, \mu)$.
- $\mathsf{ConvertSK}(\mathsf{sk}, \omega) \to \mathsf{sk}'$: On input a sk and key converter ω, output a new secret key sk′.
- $\mathsf{ConvertVK}(\mathsf{vk}, \omega) \to \mathsf{vk}'$: On input a vk and key converter ω, output a new public key vk′.
- $\mathsf{ConvertSig}(\mathsf{vk}, m, \mathbf{T}, \sigma, \omega) \to \sigma'$: On input a vk, message m, tag $\mathbf{T}$, signature σ, and key converter ω, return a new signature σ' s.t $\mathsf{Verify}(\mathsf{vk}', \mathbf{T}, m, \sigma') = 1$, where $\mathsf{vk}' \leftarrow \mathsf{ConvertVK}(\mathsf{vk}, \omega)$.

We note that VKeyGen is only required in the security definition and is never used in the construction. Although the signer receives the tag secret key τ, we replace this with a ZKP in our IhMA scheme.

3.2 Security Definitions

Correctness. We require that honest signatures verify as expected, but need to consider all the randomizations and aggregation.

Unforgeability. We model unforgeability following the ideas in the chosen-key model [9, 46], where the adversary $\mathcal{A}$ is given a single public key vk′ and access to a signing oracle on the challenge key. The adversary wins if the aggregate signature, σ, is a valid aggregate signature on a vector of messages $\mathbb{M} = (m_1, \ldots, m_n)$

under keys $(\mathsf{vk}_1, \ldots, \mathsf{vk}_n)$, and σ is nontrivial, i.e., the adversary did not request a signature on a m_j for $\mathsf{vk}_j = \mathsf{vk}'$ or more precisely where vk_j is in the same equivalence class as the challenge key vk'. $\mathcal{A}$ has the power to choose all public keys except the challenger's public key vk'. For our instantiation, however, we have to work in a slightly weakened model which is equivalent to the certified-keys model [43, 44]. In this setting the $\mathcal{A}$ registers pairs of (vk, sk) with exception of the challenge key. To model this, we have the adversary output the secret keys of the verification keys they provide in our security games. In the real world, such a key registration can be realized by requiring issuers to prove knowledge of their sk, which in the formal analysis allows a reduction to extract the secret key.

Definition 2 (Unforgeability). An AtoSa signature is unforgeable if for all PPT algorithms $\mathcal{A}$ having access to the oracle $O^{\mathsf{Sign}()}$, there exists a negligible function ϵ such that: $\Pr[\mathsf{ExpUnf}_{\mathsf{AtoSa},\mathcal{A}}(\lambda) = 1] \leq \epsilon(\lambda)$ where the experiment $\mathsf{ExpUnf}_{\mathsf{AtoSa},\mathcal{A}}(\lambda)$ is defined in Fig. 1 and Q is the set of queries that $\mathcal{A}$ has issued to the O^{Sign}.

Privacy guarantees. Similar to mercurial signatures [30], we define the following privacy notion for randomized keys vk and tags:

Definition 3 (Public key class-hiding). For all PPT adversaries $\mathcal{A}$, and $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$ there exists a negligible ϵ such that:

$$\Pr\left[\begin{array}{l}(\mathsf{vk}_1, \mathsf{sk}_1) \leftarrow \mathsf{KeyGen}(\mathsf{pp}); (\mathsf{vk}_2^0, \mathsf{sk}_2^0) \leftarrow \mathsf{KeyGen}(\mathsf{pp});\\ r \xleftarrow{\$} \mathbb{Z}_p; \mathsf{vk}_2^1 = \mathsf{ConvertVK}(\mathsf{vk}_1, r); \mathsf{sk}_2^1 = \mathsf{ConvertSK}(\mathsf{sk}_1, r);\\ b \leftarrow \{0,1\}; b' \leftarrow \mathcal{A}^{\mathsf{Sign}(\mathsf{sk}_1,\cdot),\mathsf{Sign}(\mathsf{sk}_2^b,\cdot)}(\mathsf{vk}_1, \mathsf{vk}_2^b) : b' = b\end{array}\right] \leq \frac{1}{2} + \epsilon(\lambda)$$

Definition 4 (Tag class-hiding). For all PPT adversaries $\mathcal{A}$ there is a negligible function $\epsilon(\cdot)$ such that

$$\Pr\left[\begin{array}{l}b \leftarrow \{0,1\}, \mathsf{BG} \leftarrow \mathsf{BGGen}(1^\lambda), \mathbf{T} \leftarrow \mathcal{T}, \mathbf{T}^{(0)} \leftarrow \mathcal{T},\\ \mathbf{T}^{(1)} \leftarrow [\mathbf{T}]_{\mathcal{R}}, b^* \leftarrow \mathcal{A}(\mathsf{BG}, \mathbf{T}, \mathbf{T}^{(b)}) : b^* = b\end{array}\right] - \frac{1}{2} \leq \epsilon(\lambda)$$

The tag class-hiding property for $\mathcal{R}_\tau$ is implied by the DDH assumption.

The following definition guarantees that a signature with tag $\mathbf{T}$ on a message m under vk output by ConvertSig and fed into RndSigTag produces a uniformly random signature under a uniformly random tag (from the respective tag class) and uniformly random key (from the respective key class).

Definition 5 (Origin-hiding of ConvertSig*).* For all λ, and $\mathsf{pp} \in \mathsf{Setup}(1^\lambda)$, for all $(\mathsf{vk}, m, \sigma, \mathbf{T}, \omega, \mu)$, if $\mathsf{Verify}(\mathsf{vk}, \mathbf{T}, m, \sigma) = 1$, and $(\omega, \mu) \in (\mathbb{Z}_p^*)^2$, then $(\sigma', \mathbf{T}') \leftarrow \mathsf{RndSigTag}(\mathsf{vk}, \mathbf{T}, m, \mathsf{ConvertSig}(\mathsf{vk}, m, \mathbf{T}, \sigma, \omega), \mu)$ outputs uniformly random elements in signature space and $[\mathbf{T}]_{\mathcal{R}_\tau}$) such that $\mathsf{Verify}(\mathsf{vk}', \mathbf{T}', m, \sigma') = 1$, and $\mathsf{vk}' \xleftarrow{\$} \mathsf{ConvertVK}(\mathsf{vk}, \omega)$ is a uniformly random element of $[\mathsf{vk}]_{\mathcal{R}_{\mathsf{vk}}}$.

We also require a similar definition for ConvertTag and the tag randomization:

Definition 6 (Origin-hiding of ConvertTag*).* For all λ, for all $\mathsf{pp} \in \mathsf{Setup}(1^\lambda)$, for all $(\mathsf{vk}, m, \sigma, \mathbf{T}, \mu)$, if $\mathsf{Verify}(\mathsf{vk}, \mathbf{T}, m, \sigma) = 1$, and $\mu \in \mathbb{Z}_p^*$, then $(\sigma', \mathbf{T}') \leftarrow \mathsf{RndSigTag}(\mathsf{vk}, \mathsf{ConvertTag}(\mathbf{T}, \mu), m, \sigma, \mu)$ outputs uniformly random elements in the signature space and $[\mathbf{T}]_{\mathcal{R}_\tau}$ such that $\mathsf{Verify}(\mathsf{vk}, \mathbf{T}', m, \sigma') = 1$.

3.3 Construction

We construct the AtoSa scheme based on the PS signature [52]. We can observe that to make PS signatures (h_i, s_i) aggregateable, we need the h_i components to be identical for all signatures to be aggregated. While in the original PS construction h is a random element independently chosen during signing, this can be emulated in AtoSa by generating h for all signatures via a hash function based on some common information embedded in aux. For example, aux, could be a concatenation of all the messages and the tag. This technique was implicitly used in Coconut [57] and Camenisch et al. [16], and has recently been formalized by Crites et al. in [29].

We note that we should be careful when computing h, i.e., in choosing aux, as in PS signatures one can forge signatures when obtaining two signatures on two different messages with respect to the same element h. To prevent forgeries when aiming to aggregate signatures, a unique base h for a set of messages signed under the same tag is required. Therefore, we compute h as a hash of a concatenation of the messages to be signed and corresponding verification keys, denoted as aux. This approach ensures that every signer computes signatures on the same base h. We also introduce a new definition and function:

Aux binding. To ensure this property of h while making our construction modular, we define a straightforward property of $\mathsf{GenAuxTag}(S)$, i.e., no adversary can "open" an aux to two messages for the same signer. This definition is paired with the function VerifyAux which is called by Sign.

Definition 7 (Aux binding). We split aux into a preimage and an opening: (c, o). For all PPT $\mathcal{A}$, and $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$ and $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{VKeyGen}(1^\lambda)$ there exists a negligible ϵ such that:

$$\Pr\left[\begin{array}{l}(h, \mathsf{aux} = (c, o), \mathsf{aux} = (c', o'), \tau, m, \tau', m') \leftarrow \mathcal{A}(\mathsf{vk});\\ \mathsf{VerifyAux}(\mathsf{sk}, (c, o), \tau, m) = 1\\ \wedge\ \mathsf{VerifyAux}(\mathsf{sk}, (c', o'), \tau', m') = 1;\\ c = c' \wedge ([\tau]_{\mathcal{R}_\tau} \neq [\tau']_{\mathcal{R}_\tau} \vee m \neq m')\end{array}\right] \leq \epsilon(\lambda)$$

We will then hash the preimage, c in our construction to reduce to the GPS assumption [29] effectively. The o value in this definition may seem unnecessary, but it will become useful when we introduce our IhMA construction in Section 5. We've left aux binding out of our definition and rather defined it in our construction in order to make our definition more generic as aux binding is simply a property we use in the proof to ensure that our construction satisfies the definition of AtoSa.

Synchronicity assumption. We note that when we do not want to fix messages and verification keys in aux beforehand, then we can make assumption as in synchronized aggregate signatures [1, 41] and require each signer to only issue *a single signature per tag*. In this case aux only contains the tag and in the construction below we set $c = P^{\rho_1} || P^{\rho_2}$ and Definition 7 is trivially satisfied.

We involve the tag in signatures by exponentiating the component h with the secret part of the tag h^ρ and compute the component s using this value, which clearly can be checked via a pairing with the tag's public part and verified like a standard PS signature. Moreover, AtoSa allows the randomization of tag, vk and signatures via a change of representatives tag, vk and a matching signature update.

$\underline{\mathsf{ExpUnf}_{\mathsf{AtoSa},\mathcal{A}}(\lambda)}$:

- $Q := \emptyset$; $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$;
- $(\mathsf{vk}', \mathsf{sk}') \leftarrow \mathsf{KeyGen}(\mathsf{pp})$;
- $(j', \mathsf{avk} = (\mathsf{vk}_j)_{j\in[\ell]}, a\mathsf{sk} = (\mathsf{sk}_j)_{j\in[\ell]\setminus j'}, \mathbb{M}^* = (m_j^*)_{j\in[\ell]}, (\tau^*, \mathbf{T}^*), \sigma^*) \leftarrow \mathcal{A}^{O}(\mathsf{pp}, \mathsf{vk}')$
- $(\mathsf{vk}_j^*) := \left(\mathsf{VKeyGen}\left(\mathsf{sk}_j\right)\right)_{j\in[\ell]\setminus j'}$,

return:

$$\begin{pmatrix} \mathsf{VerifyAggr}\left(\mathsf{avk}, \mathbf{T}^*, \sigma^*, \mathbb{M}^*\right) = 1 \ \wedge \forall j \in [\ell], j \neq j' : \\ [\mathsf{vk}_j^*]_{\mathcal{R}_{\mathsf{vk}}} = [\mathsf{vk}_j]_{\mathcal{R}_{\mathsf{vk}}} \wedge \ [\mathsf{vk}']_{\mathcal{R}_{\mathsf{vk}}} = [\mathsf{vk}_{j'}]_{\mathcal{R}_{\mathsf{vk}}} \\ \wedge \forall (m, \mathbf{T}) \in Q : m \neq m_j^* \vee [\mathbf{T}]_{\mathcal{R}_\tau} \neq [\mathbf{T}^*]_{\mathcal{R}_\tau} \end{pmatrix}$$

$\underline{O^{\mathsf{Sign}}(m, \mathsf{aux}, (\tau, \mathbf{T}))}$:

- $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}', \tau, \mathsf{aux}, m)$
- $Q = Q \cup \{m, \mathbf{T}\}$,

return σ

Figure 1: Experiment $\mathsf{ExpUnf}_{\mathsf{AtoSa},\mathcal{A}}(\lambda)$

Our construction. The construction is as follows:

$\mathsf{Setup}(1^\lambda)$: Run $\mathsf{BG} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, P, \hat{P}, e) \leftarrow \mathsf{BGGen}(1^\lambda)$ with a prime number order p, where P is a generator of $\mathbb{G}_1$, $\hat{P}$ a generator of $\mathbb{G}_2$. Pick H as a hash function: $H : \{0,1\}^* \rightarrow \mathbb{G}_1$. Output public parameters $\mathsf{pp} = \{\mathsf{BG}, H\}$.

$\mathsf{KeyGen}\,(\mathsf{pp})$: Choose $(x, y_1, y_2) \xleftarrow{\$} \mathbb{Z}_p$ and set the secret key $\mathsf{sk} = (x, y_1, y_2)$ and verification key $\mathsf{vk} = (\hat{Y}_1 = \hat{P}^{y_1}, \hat{Y}_2 = \hat{P}^{y_2}, \hat{X} = \hat{P}^x)$.

$\mathsf{VKeyGen}\,(\mathsf{sk})$: On input a secret key $\mathsf{sk} = (x, y_1, y_2)$, output $\mathsf{vk} = (\hat{Y}_1 = \hat{P}^{y_1}, \hat{Y}_2 = \hat{P}^{y_2}, \hat{X} = \hat{P}^x)$.

$\mathsf{GenAuxTag}(S)$: Given a set $S = \{(m_j, \mathsf{vk}_j)_{j\in[\ell]}\}$, choose $(\rho_1, \rho_2) \xleftarrow{\$} \mathbb{Z}_p$, set $c = P^{\rho_1}||P^{\rho_2}||(m_j, \mathsf{vk}_j)_{j\in[\ell]}$. Next set all $\mathsf{aux}_j = (c, \perp)$. Compute $h = H(c)$ and output aux and a tag pair $(\tau = (\rho_1, \rho_2), \mathbf{T} = (T_1 = h^{\rho_1}, T_2 = h^{\rho_2}))$.

$\mathsf{VerifyAux}(\mathsf{sk}, \mathsf{aux}, \tau, m_j)$ Parse aux as (c, o). Check that $\tau \in c$ (i.e., that c has the form $P^{\rho_1}||P^{\rho_2}||...)$ and $(m_j, \mathsf{vk}) \in c$ where vk is a verification key related to sk (in the same equivalence class). Also check that no other vk_j in aux has the same equivalence class as sk. This can be done by checking that $\hat{Y}_2 = \hat{Y}_1^{\frac{y_2}{y_1}}$ and that $\hat{X} = \hat{Y}_2^{\frac{x}{y_2}}$. If these checks pass, it means that this is in the same equivalence class as the verifier's key. If the check doesn't pass, it means the vk_j is not in the same equivalence class.

$\mathsf{Sign}(\mathsf{sk}_j, \tau, \mathsf{aux}_j, m_j)$: Given a $\mathsf{sk}_j = (y_{1j}, y_{2j}, x_j)$, τ, aux_j and a message m_j. If $\mathsf{VerifyAux}(\mathsf{sk}_j, \mathsf{aux}_j, \tau, m_j) \neq 1$ return $\perp$. Else, parse aux as (c, o) and compute $h = H(c)$ and output:

$$\sigma_j = \left(h', s_j\right) = \left(h' = h^{\rho_1}, s_j = (h^{\rho_1})^{x_j + y_{1j}\cdot m_j} \cdot (h^{\rho_2})^{y_{2j}}\right)$$

$\mathsf{Verify}(\mathsf{vk}_j, \mathbf{T}, m_j, \sigma_j)$: Given a vk_j, tag $\mathbf{T} = (T_1, T_2)$, message m_j and signature σ_j, parse σ_j as $\left(h', s_j\right)$ and return 1 if the following checks hold and 0 otherwise:

$$e(h', \hat{X} \cdot \hat{Y}_1^{m_1}) e(T_2, \hat{Y}_2) = (s_j, \hat{P}) \ \wedge \ T_1 = h' \neq 1_{\mathbb{G}}$$

$\mathsf{AggrSign}(\mathbf{T}, \{(\mathsf{vk}_j, m_j, \sigma_j)\}_{j=1}^{\ell})$: Given ℓ valid signatures such that $\forall j \in [\ell]$, $\sigma_j = (h', s_j)$ for m_j under vk_j and the same tag $\mathbf{T}$, where $j \in [\ell]$, outputs an aggregate signature σ on the messages $\mathbb{M} = (m_j)_{j\in[\ell]}$ under the tag $\mathbf{T}$ and aggregated verification key $\mathsf{avk} = (\mathsf{vk}_j)_{j\in[\ell]}$ as: $\sigma' = \left(h', s' = \prod_{j=1}^{\ell} s_j\right)$.

$\mathsf{VerifyAggr}(\mathsf{avk}, \mathbf{T}, \mathbb{M}, \sigma)$: Given an avk, tag $\mathbf{T}$, messages $\mathbb{M}$ and aggregate signature $\sigma = (h', s)$, it outputs 1 if the following checks holds and 0 otherwise:

$$e\left(h', \prod_{j\in[\ell]} \hat{X}_j \cdot \hat{Y}_{1j}^{m_j}\right) e\left(h^{\rho_2}, \prod_{j\in[\ell]} \hat{Y}_{2j}\right) = e\left(s, \hat{P}\right) \wedge T_1 = h' \neq 1_{\mathbb{G}}$$

$\mathsf{ConvertTag}(\mathbf{T}, \mu) \rightarrow \mathbf{T}'$: On input a tag $\mathbf{T}$ and randomness μ, output a randomized tag $\mathbf{T}' = \mathbf{T}^\mu = (T_1^\mu, T_2^\mu)$.

$\mathsf{RndSigTag}(\mathsf{vk}, \mathbf{T}, m, \sigma, \mu) \rightarrow (\sigma', \mathbf{T}')$: Given a signature σ on message m under a valid tag $\mathbf{T}$ and vk, and randomness μ. Return a randomized signature σ' and a randomized tag:

$$\sigma' = \left(h'^\mu, s^\mu\right), \ \mathbf{T}' \leftarrow \mathsf{ConvertTag}(\mathbf{T}, \mu)$$

where is a valid signature for a new tag representative $\mathbf{T}' \in [\mathbf{T}]_{\mathcal{R}_\tau}$.

$\mathsf{ConvertSK}(\mathsf{sk}, \omega)$: On input sk and a key converter $\omega \in \mathbb{Z}_p^*$, output a new secret key sk' as $\mathsf{sk}' = \mathsf{sk} \cdot \omega$.

$\mathsf{ConvertVK}(\mathsf{vk}, \omega)$: On input vk and a key converter $\omega \in \mathbb{Z}_p^*$, output a new public key as $\mathsf{vk}' = \mathsf{vk}^\omega$.

$\mathsf{ConvertSig}(\mathsf{vk}, m, \mathbf{T}, \sigma, \omega)$: On input a vk, message m, signature σ, tag $\mathbf{T}$, and key converter $\omega \in \mathbb{Z}_p^*$, return a new signature σ' s.t. $\mathsf{Verify}(\mathsf{vk}', \mathbf{T}, m, \sigma') = 1$, where $\mathsf{vk}' \xleftarrow{\$} \mathsf{ConvertVK}(\mathsf{vk}, \omega)$ as follows: $\sigma' = (h', s' = s^\omega)$.

The correctness of our construction follows from inspection. We formally show the unforgeability and privacy notations.

Theorem 8 (Unforgeability). *Our construction achieves the EUF-CMA security stated in Def 2, under the hardness of GPS assumption, in the random oracle model.*

Theorem 9 (Privacy). *Our construction is origin-hiding of ConvertSig, origin-hiding of* RndSigTag*, tag class hiding and has public key class-hiding based on Def. 5, Def. 6, Def. 4, and Def. 3, respectively.*

The proofs of Theorem 9 and Theorem 8 are provided in the full version [47].

4 AGGREGATE MERCURIAL SIGNATURES WITH RANDOMIZABLE TAGS

We now present an aggregate mercurial signature with randomizable tags (ATMS). Similar to AtoSa, (see Def. 1), one can aggregate mercurial signatures of different messages under different keys under the same tag and randomize those signatures, public keys, and tags. ATMS differs from AtoSa by in addition supporting equivalence classes on the message space. This further allows the randomization of messages, leading to a feature known from

structure-preserving signature on equivalence classes (SPSEQ) and, more precisely, mercurial signatures.

To achieve the aggregation property, we follow the strategy presented by Crites et al. in context of threshold SPS [29], where the authors define a so called Indexed Diffie-Hellman message space $\mathcal{M}^H_{\mathsf{iDH}}$. But the main problem with this approach, as it is defined over both groups, is that we can not define indistinguishable equivalence classes over $\mathbb{G}_1^k \times \mathbb{G}_2^k$, since spanning both groups makes DDH easy and would yield trivial linkability. Note that given both $((M_1, M_2), (N_1, N_2))$ and $((M_1', M_2'), (N_1', N_2'))$, one can easily link them together by checking that $e(M_1, N_2') = e(M_2, N_1')$ and $e(M_1', N_2) = e(M_2', N_1)$ hold. So we adapt $\mathcal{M}^H_{\mathsf{iDH}}$ and define a new message space called a Tag-based DH message space $\mathcal{M}^H_{\mathsf{TDH}}$ and its corresponding EQ relation. We essentially define one equivalence class per group and tie them together via the message, the tag, and an index obtained via some auxiliary information (similar to the aux in the case of AtoSa). Indeed we adapt the Diffie-Hellman message space $\mathcal{M}_{\mathsf{DH}}$ to a Tag-based DH message space $\mathcal{M}^H_{\mathsf{TDH}}$ for a tuple $(\mathsf{aux}, h, \mathbf{T}, M, N)$, which includes a tag $\mathbf{T}$ with auxiliary data aux (instead of the *id*).

This new message space then allows us to aggregate and define an equivalence (EQ) relation which gives an indistinguishable message space.

4.1 Formal Definitions

We begin our definitions by introducing Tag-based DH message space $\mathcal{M}^H_{\mathsf{TDH}}$ and give an instantiation in the random oracle model (ROM). Then we define a new EQ relation regarding this message space $\mathcal{M}^H_{\mathsf{TDH}}$, and finally, we define our new primitive ATMS.

A Tag-based DH **message space**. We adapt the message indexing technique introduced by [29] (cf. Def. 2) to tags:

Definition 10 (A Tag-based DH *message space ($\mathcal{M}^H_{\mathsf{TDH}}$)).* Let H be a random oracle. For the aux and tag $\mathbf{T} = (h^{\rho_i})_{i\in[k]}$, we define $\mathcal{M}^H_{\mathsf{TDH}}$ as a tag based DH message space, if the following property hold: For the messages vector $(\mathbf{M}, \mathbf{N}) = (M_1, \ldots, M_k, N_1, \ldots, N_k)$ there exists $m_i \in \mathbb{Z}_p$ s.t. for each tuple $(\mathsf{aux}, T_i = h^{\rho_i}, M_i = T_i^{m_i}, N_i = \hat{P}^{m_i})$, the following holds: $e(M_i, \hat{P}) = e(T_i, N_i)$.

We provide an instantiation in Fig. 2. Let us assume WLOG a message vector with the length $k = 2$ as $\mathbf{m} = (m_1, m_2)$, this can be generalized to any length $k > 1$.

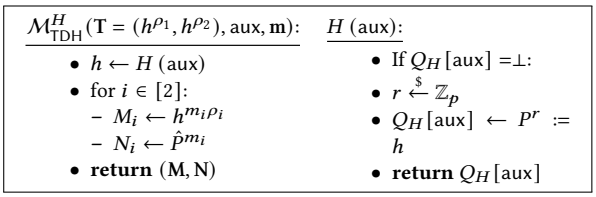


Figure 2: Tag based Diffie-Hellman message space in ROM

Equivalence relations (EQ) over $\mathcal{M}^H_{\mathsf{TDH}}$. Let the message space $\mathcal{M}^H_{\mathsf{TDH}}$ be defined as $(\mathbf{M}, \mathbf{N}) = (M_1, \ldots .M_k, N_1, \ldots, N_k) \in (\mathbb{G}_1^*)^k \times (\mathbb{G}_2^*)^k$ such that for $(h, \mathbf{T})$, and $i \in [k]$: $e(M_i, \hat{P}) = e(T_i, N_i)$. Now we can define a family of equivalence relations $\mathbb{R}^\ell$ so that for any ℓ with $1 < k \leq \ell$. We define the following equivalence relation $\mathcal{R}_{\mathsf{TDH}} \in \mathbb{R}^\ell$ and the equivalence class $[(\mathbf{M}, \mathbf{N})]_{\mathcal{R}_{\mathsf{TDH}}}$ of a message vector with size k. More concretely, for a fixed bilinear group BG and (k, ℓ), we define $\mathcal{R}_{\mathsf{TDH}} \in \mathbb{R}^\ell$ as follows:

Definition 11 (Equivalence relations of $\mathcal{M}^H_{\mathsf{TDH}}$ message spaces). If vectors of a pair $(\mathbf{M}, \mathbf{N}) \in (\mathbb{G}_1^*)^k \times (\mathbb{G}_2^*)^k$ is a message vector from $\mathcal{M}^H_{\mathsf{TDH}}$, then the equivalence relations $[(\mathbf{M}, \mathbf{N})]_{\mathcal{R}_{\mathsf{TDH}}}$ defined as

$$\mathcal{R}_{\mathsf{TDH}} = \left\{ \begin{matrix} (\mathbf{M}, \mathbf{N}), (\mathbf{M}', \mathbf{N}') \in (\mathbb{G}_1^* \times \mathbb{G}_2^*)^k \times (\mathbb{G}_1^* \times \mathbb{G}_2^*)^k \Leftrightarrow \exists (\mu, \upsilon) \in \mathbb{Z}_p^* : \\ \mathbf{M}' = \mathbf{M}^{\mu\upsilon}, \mathbf{N}' = \mathbf{N}^{\upsilon} \end{matrix} \right\}$$

Note that the EQ relation for an aggregate signature on a set of vectors $\mathbb{M} = ((\mathbf{M}_j, \mathbf{N}_j))_{j\in[\ell]}$ is the family (set) of relation as above, while all vectors use the same randomness $\mathbb{M} = ((\mathbf{M}_j^{\mu\upsilon}, \mathbf{N}_j^{\upsilon}))_{j\in[\ell]}$. For instance, the j'th message vector $(\mathbf{M}_j, \mathbf{N}_j) \in [(\mathbf{M}, \mathbf{N})]^j_{\mathcal{R}^j_{\mathsf{TDH}}}$ is in the class $\mathcal{R}^j_{\mathsf{TDH}} \in \mathbb{R}^\ell$ and if one more signature-message pair is added to the set, we have $\mathcal{R}^{j+1}_{\mathsf{TDH}} \in \mathbb{R}^\ell$, where $j + 1 < \ell$. Moreover, we consider the EQ relation for verification keys vk and Tag similar to AtoSa and indicate as $\mathcal{R}_{\mathsf{vk}}$ and $\mathcal{R}_\tau$ as stated in Def. 3.1. We again denote by $\mathcal{T}$ the space of all tags and present the ATMS in Def. 12.

*Definition 12 (Aggregate Mercurial Signatures with Randomizable Tag (*ATMS*)).* An ATMS scheme, associated with the parameterized equivalence relations $\mathbb{R}^\ell$, $\mathcal{R}_{\mathsf{TDH}}$, $\mathcal{R}_\tau$ and $\mathcal{R}_{\mathsf{vk}}$, and also message space $\mathcal{M}^H_{\mathsf{TDH}}$ consists of the algorithms:

- $\mathsf{Setup}(1^\lambda) \rightarrow \mathsf{pp}$: On input the security parameter λ, output the public parameters pp.
- $\mathsf{KeyGen}(\mathsf{pp}) \rightarrow (\mathsf{sk}, \mathsf{vk})$: On input the public parameters pp, output a key pair $(\mathsf{sk}, \mathsf{vk})$.
- $\mathsf{VKeyGen}(\mathsf{sk})$: On input a secret key sk, output a verification key vk.
- $\mathsf{GenAuxTag}(S) \rightarrow (\mathsf{aux}_j, (\tau, \mathbf{T}))$: Given a set $S = \left(\left(\mathbf{M}_j, \mathbf{N}_j\right), \mathsf{vk}_j\right)_{j\in[n]}$ of messages and keys, output auxiliary data aux_j and a tag pair $(\tau, \mathbf{T})$ where τ is the secret part and $\mathbf{T}$ is the public part of tag and all vk_j should be distinct.
- $\mathsf{Sign}(\mathsf{sk}_j, \tau, \mathsf{aux}_j, (\mathbf{M}_j, \mathbf{N}_j)) \rightarrow \sigma_j$: On input a secret key sk_j, tag's secret τ, auxiliary data aux_j and message vector $(\mathbf{M}_j, \mathbf{N}_j) \in \mathcal{M}^H_{\mathsf{TDH}}$, output a signature σ_j under the τ, vk_j and $(\mathbf{M}_j, \mathbf{N}_j)$.
- $\mathsf{Verify}(\mathsf{vk}_j, \mathbf{T}, (\mathbf{M}_j, \mathbf{N}_j), \sigma_j) \rightarrow \{0, 1\}$: Given a verification key vk_j, tag's public $\mathbf{T}$, message vector $(\mathbf{M}_j, \mathbf{N}_j)$ and signature σ_j, output 1 if σ_j is valid relative to vk_j, $(\mathbf{M}_j, \mathbf{N}_j)$ and $\mathbf{T}$, and 0 otherwise.
- $\mathsf{VerifyTag}(\mathbf{T}, \tau, \sigma) \rightarrow \{0, 1\}$: Given a tag's public $\mathbf{T}$, tag's secret signature σ, output 1 if $\mathbf{T}$ is valid relative to σ, and τ, and 0 otherwise.
- $\mathsf{AggrSign}(\mathbf{T}, (\mathsf{vk}_j, (\mathbf{M}_j, \mathbf{N}_j), \sigma_j)^\ell_{j=1}) \rightarrow \sigma'$ Given ℓ signed messages $(\mathbf{M}_j, \mathbf{N}_j)$ in σ_j under vk_j for $j \in [\ell]$ and the same tag $\mathbf{T}$, output a signature σ on the messages $\mathbb{M} = ((\mathbf{M}_j, \mathbf{N}_j))_{j\in[\ell]}$ under the tag $\mathbf{T}$ and verification key $\mathsf{avk} = (\mathsf{vk}_j)_{j\in[\ell]}$.
- $\mathsf{VerifyAggr}(\mathsf{avk}, \mathbf{T}, \mathbb{M}, \sigma) \rightarrow \{0, 1\}$: Given a verification key avk, tag $\mathbf{T}$, messages $\mathbb{M}$ and signature σ, output 1 if σ is valid relative to avk, $\mathbb{M}$ and $\mathbf{T}$, and 0 otherwise.
- $\mathsf{ConvertTag}(\mathbf{T}, \mu) \rightarrow \mathbf{T}'$: On input a tag $\mathbf{T}$ and randomness μ, output a randomized tag $\mathbf{T}' \in [\mathbf{T}]_{\mathcal{R}_\tau}$ (i.e., a new representative of tag).

ChangRep$((\mathbf{M},\mathbf{N}),\sigma,\mathbf{T},(\mu,\upsilon)) \rightarrow (\sigma',\mathbf{T}')$: On input a representative $(\mathbf{M},\mathbf{N}) \in [(\mathbf{M},\mathbf{N})]_{\mathcal{R}_{\mathsf{TDH}}}$, $\mathbf{T} \in [\mathbf{T}]_{\mathcal{R}_\tau}$, signature σ and randomness (μ,υ), return a new signature $((\mathbf{M}',\mathbf{N}'),\mathbf{T}',\sigma')$, where $\mathbf{M}' = \mathbf{M}^{\mu\upsilon} \wedge \mathbf{N}' = \mathbf{N}^{\upsilon} \in [(\mathbf{M},\mathbf{N})]_{\mathcal{R}_{\mathsf{TDH}}}$ and $\mathbf{T}' \leftarrow$ ConvertTag$(\mathbf{T},\mu)$ are the new representatives and σ' is valid for $(\mathbf{M}',\mathbf{N}')$ and $[\mathbf{T}]_{\mathcal{R}_\tau}$.
This will also apply for a set representative $\mathbb{M}$ such that one can get a new set representative $\mathbb{M}'$ by scaling all message with the same (μ,υ).

ConvertSK$(\mathsf{sk},\omega) \rightarrow \mathsf{sk}'$: On input a sk and key converter ω, output a new secret key sk'.

ConvertVK$(\mathsf{vk},\omega) \rightarrow \mathsf{vk}'$: On input a vk and key converter ω, output a new public key vk'.

ConvertSig$(\mathsf{vk},\mathbf{T},(\mathbf{M},\mathbf{N}),\sigma,\omega) \rightarrow \sigma'$: On input a vk, message vector $(\mathbf{M},\mathbf{N})$, signature with tag $(\sigma,\mathbf{T})$, and key converter ω, return a new signature σ' such that Verify$(\mathsf{vk}',\mathbf{T},(\mathbf{M},\mathbf{N}),\sigma') = 1$, where $\mathsf{vk}' \leftarrow$ ConvertVK(vk,ω).

The VerifyTag and VKeyGen are only used for the security game.

4.2 Security Definitions

Correctness. As usual we require that honest signatures verify as expected, but need to consider all the randomizations as well as the aggregation.

Unforgeability. The unforgeability game follows the unforgeability definition of AtoSa (see Def. 2). It is slightly modified to fit with our additional EQ relation (Def. 11), i.e., unforgeability is defined with respect to message classes and in addition need to check VerifyTag.

Definition 13 (Unforgeability). An ATMS is unforgeable if for all PPT $\mathcal{A}$ having access to the oracle $O^{\mathsf{Sign}()}$ there exists a negligible function ϵ s.t: $\Pr[\mathsf{ExpUnf}_{\mathsf{ATMS},\mathcal{A}}(\lambda) = 1] \leq \epsilon(\lambda)$ where the experiment $\mathsf{ExpUnf}_{\mathsf{ATMS},\mathcal{A}}(\lambda)$ is defined in Fig. 3 and Q is the set of queries that $\mathcal{A}$ has issued to $O^{\mathsf{Sign}()}$.

Privacy guarantees. Similar as in Section 3, we consider the privacy notations *Origin-hiding of ConvertSig*, and *Public key class-hiding* (it is the same as Def. 3). We note that all definitions can be updated due to $\mathcal{M}^H_{\mathsf{TDH}}$ message space (receptively EQ relations of $\mathcal{M}^H_{\mathsf{TDH}}$) instead of the vector $\mathbf{M}$. Origin-hiding of ConvertSig definition can be updated straightforwardly as follows:

Definition 14 (Origin-hiding of ConvertSig *for* ATMS*).* For all λ, and $\mathsf{pp} \in \mathsf{Setup}(1^\lambda)$, for all $(\mathsf{vk},(\mathbf{M},\mathbf{N}),\sigma,\mathbf{T},\omega,\upsilon,\mu)$, if Verify$(\mathsf{vk},\mathbf{T},(\mathbf{M},\mathbf{N}),\sigma) = 1$, and $(\omega,\upsilon,\mu) \in (\mathbb{Z}_p^*)^3$, then $\sigma' \leftarrow$ ChangRep$((\mathbf{M},\mathbf{N})$, ConvertSig$(\mathsf{vk},\mathbf{T},(\mathbf{M},\mathbf{N}),\sigma,\omega),\mathbf{T},(\upsilon,\mu))$ outputs a uniformly random element in the respective space s.t. Verify$(\mathsf{vk}',\mathbf{T}',(\mathbf{M}',\mathbf{N}'),\sigma') = 1$, where $\mathsf{vk}' \xleftarrow{\$}$ ConvertVK(vk,ω) outputs a uniformly random element of $[\mathsf{vk}]_{\mathcal{R}_{\mathsf{vk}}}$.

However, since this is a variant of SPSEQ we consider the *adaption property* similar to [35] below, an additional property which guarantees that signatures from ChangRep and Sign are identically distributed. This definition also covers Origin-hiding of ConvertTag.

Definition 15 (Perfect Adaption of Signatures). An ATMS scheme perfectly adapts signatures if for all $(\mathsf{vk},\mathbf{T},(\mathbf{M},\mathbf{N}),\sigma,\mu,\upsilon)$ with $(\mathbf{M},\mathbf{N}) \in \mathcal{M}^H_{\mathsf{TDH}} \wedge$ Verify$(\mathsf{vk},\mathbf{T},(\mathbf{M},\mathbf{N}),\sigma) = 1 \wedge\ (\mu,\upsilon) \in \mathbb{Z}_p^*$ we have that the output of $(\sigma',\mathbf{T}') \leftarrow$ ChangRep$(\sigma,(\mathbf{M},\mathbf{N}),\mathbf{T},(\mu,\upsilon))$ is a uniformly random element in the respective space, conditioned on Verify$(\mathsf{vk},\mathbf{T}^\mu,(\mathbf{M}^{\mu\upsilon},\mathbf{N}^\upsilon),\sigma') = 1$.

4.3 Construction

Our construction is inspired by the message-indexed SPS by Crites et al. [29], which is a variant of Ghadafi's SPS [37]. We use the tag-based message definition $\mathcal{M}^H_{\mathsf{TDH}}$ (Def. 10) instead of the message-indexed (Def. 2). For simplicity, we assume a message vector with the length $k = 2$ as $(\mathbf{M},\mathbf{N}) = ((M_1,M_2),(N_1,N_2))$, but this can be straightforwardly generalized to any length $k > 1$. Similar to the construction in Section 3.3, we again need aux binding to make this particular construction work.

Definition 16 (Aux binding for ATMs*).* We split aux into a preimage and an opening: (c,o). For all PPT $\mathcal{A}$, and $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$ and $(\mathsf{sk},\mathsf{vk}) \leftarrow \mathsf{VKeyGen}(1^\lambda)$ there exists a negligible ϵ such that:

$$\Pr\left[\begin{array}{l}(\mathsf{aux}=(c,o),\mathsf{aux}=(c',o'),\tau,(\mathbf{M},\mathbf{N}),\tau',(\mathbf{M}',\mathbf{N}')) \leftarrow \mathcal{A}(\mathsf{vk});\\ \mathsf{VerifyAux}(\mathsf{sk},(c,o),\tau,(\mathbf{M},\mathbf{N})) = 1\\ \wedge\ \mathsf{VerifyAux}(\mathsf{sk},(c',o'),\tau',(\mathbf{M}',\mathbf{N}')) = 1 \wedge\\ c = c' \wedge (\tau \neq \tau' \vee (\mathbf{M},\mathbf{N}) \neq (\mathbf{M}',\mathbf{N}'))\end{array}\right] \leq \epsilon(\lambda)$$

Synchronicity assumption. Same as in Section 3.3, instead of fixing messages and verification keys in aux, we can make same assumption as in synchronized aggregate signatures and simply set $c = P^{\rho_1}||P^{\rho_2}$ in the construction below and Definition 7 is trivially satisfied.

Our construction. The construction is as follows:

Setup(1^λ): Run $\mathsf{BG} = (p,\mathbb{G}_1,\mathbb{G}_2,\mathbb{G}_T,P,\hat{P},e) \leftarrow \mathsf{BGGen}(1^\lambda)$ with a prime number order p, where P a generator of $\mathbb{G}_1$, $\hat{P}$ a generator of $\mathbb{G}_2$ and H a hash function: $H : \{0,1\}^* \rightarrow \mathbb{G}_1$, output $\mathsf{pp} = (\mathbb{G}_1,\mathbb{G}_2,\mathbb{G}_T,P,\hat{P},H)$.

KeyGen(pp): Given pp, sample $\mathsf{sk} = (x,y_1,y_2,z_1,z_2) \xleftarrow{\$} (\mathbb{Z}_p^*)^5$, and $\mathsf{vk} = (\hat{X} = \hat{P}^x, \hat{Y}_1 = \hat{P}^{y_1}, \hat{Y}_2 = \hat{P}^{y_2}, \hat{Z}_1 = \hat{P}^{z_1}, \hat{Z}_2 = \hat{P}^{z_2})$.

VKeyGen (sk): Given $\mathsf{sk} = (x,y_1,y_2,z_1,z_2)$, return $\mathsf{vk} = (\hat{X} = \hat{P}^x, \hat{Y}_1 = \hat{P}^{y_1}, \hat{Y}_2 = \hat{P}^{y_2}, \hat{Z}_1 = \hat{P}^{z_1}, \hat{Z}_2 = \hat{P}^{z_2})$.

GenAuxTag(S): Given a set $S = \{(\mathbf{M}_j,\mathbf{N}_j,\mathsf{vk}_j)_{j\in[n]}\}$, choose $(\rho_1,\rho_2) \xleftarrow{\$} \mathbb{Z}_p$, set $\tau = (\rho_1,\rho_2), \mathbf{T} = (T_1 = h^{\rho_1}, T_2 = h^{\rho_2})$, and $c = \left(P^{\rho_1}||P^{\rho_2}||(\mathbf{N}_j,\mathsf{vk}_j)_{j\in[n]}\right)$, where $h = H(c)$ and $\mathsf{aux}_j = (c, o = \perp)$.

VerifyAux(sk, aux, (τ_1,τ_2), $((M_1,M_2),(N_1,N_2))$) : Extract (T_1,T_2), parse aux as (c,o). Check that $((\mathbf{M},\mathbf{N}),[\mathsf{VKeyGen}(\mathsf{sk})]) \in$ aux (i.e., $c = ...||((\mathbf{M},\mathbf{N}),[\mathsf{VKeyGen}(\mathsf{sk})])||...$) s.t no other vk in aux related to sk and check that $(T_1,T_2) = (h^{\tau_1},h^{\tau_2})$. Compute $h := H(c)$. Output $\bigwedge_{i=1}^2 e(M_i,\hat{P}) = e(h^{\tau_i},N_i)$.

Sign $\left(\mathsf{sk}_j,\tau,\mathsf{aux}_j,(\mathbf{M},\mathbf{N})\right)$: Given a sk_j, τ, $\mathsf{aux}_j = (c,\perp)$, and message $(\mathbf{M},\mathbf{N}) = ((M_1,M_2),(N_1,N_2)) \in \mathcal{M}^H_{\mathsf{TDH}}$. Parse τ as (ρ_1,ρ_2). Run VerifyAux(sk, aux, τ, $(\mathbf{M},\mathbf{N})$) and verify that this outputs 1. If so compute $h = H(c)$ and output a signature as:

$$\sigma = (h, b = \prod_{j\in[2]} h^{\rho_j\cdot z_j}, s = (h^x \cdot \prod_{j\in[2]} M_j^{y_j})).$$

Verify$(\mathsf{vk},\mathbf{T},(\mathbf{M},\mathbf{N}),\sigma)$: Given a vk, tag $\mathbf{T} = (T_1 = h^{\rho_1}, T_2 = h^{\rho_2})$, message $(\mathbf{M},\mathbf{N})$ and signature $\sigma = (h,b,s)$ return 1 if the

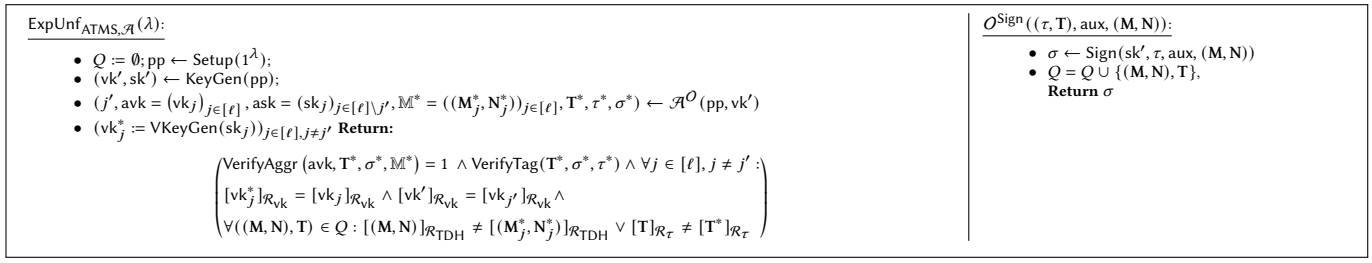


Figure 3: Experiment $\mathsf{ExpUnf}_{\mathsf{ATMS},\mathcal{A}}(\lambda)$

following holds and 0 otherwise:

$$e(h, \hat{X}) \prod_{j\in[2]} e(M_j, \hat{Y}_j) = e(s, \hat{P}) \ \wedge \ e(b, \hat{P}) = \prod_{j\in[2]} e(T_j, \hat{Z}_j)$$
$$\bigwedge_{j=1}^{2} e(T_j, N_j) = e(M_j, \hat{P})$$

$\mathsf{VerifyTag}(\mathbf{T}, \tau, \sigma)$: Given $\tau = (\tau_1, \tau_2)$, $\sigma = (h, b, s)$, output 1 if $T_i = h^{\tau_i}$ for all $i \in \{1, 2\}$, and 0 otherwise.

$\mathsf{AggrSign}(\mathbf{T}, (\mathsf{vk}_i, (\mathbf{M}_i, \mathbf{N}_i), \sigma_i)_{i=1}^{\ell})$: Given ℓ valid signatures $\sigma_i = (h, b_i, s_i)$ for $(\mathbf{M}_i, \mathbf{N}_i)$ under vk_i and the same tag $\mathbf{T}$ for $i \in [\ell]$, return $\perp$ if all h are not the same, else output a signature σ on the messages $\mathbb{M} = ((\mathbf{M}_i, \mathbf{N}_i))_{i\in[\ell]}$ under the tag $\mathbf{T}$ and aggregated verification key $\mathsf{avk} = (\mathsf{vk}_1, \ldots, \mathsf{vk}_\ell)$ as follows: $\sigma = \left(h, b' = \prod_{i=1}^{\ell} b_i, s' = \prod_{i=1}^{\ell} s_i\right)$.

$\mathsf{VerifyAggr}(\mathsf{avk}, \mathbf{T}, \mathbb{M}, \sigma)$: Given $\mathsf{avk} = (\mathsf{vk}_1, \ldots, \mathsf{vk}_\ell)$, tag $\mathbf{T} = (T_1 = h^{\rho_1}, T_2 = h^{\rho_2})$, messages $\mathbb{M}$ and signature $\sigma = (h, b, s)$, check if the following checks holds and 0 otherwise:

$$\prod_{i\in[\ell]} e(h, \hat{X}_i) \prod_{j\in[2]} e(M_{ij}, \hat{Y}_{ij}) = e(s, \hat{P}) \ \wedge \ e(b, \hat{P}) = \prod_{i\in[\ell]} \prod_{j\in[2]} e(T_j, \hat{Z}_{ij})$$
$$\bigwedge_{j\in[2]\wedge i\in[\ell]} e(T_j, N_{ij}) = e(M_{ij}, \hat{P})$$

$\mathsf{ConvertTag}(\mathbf{T}, \mu) \rightarrow \mathbf{T}'$: On input a tag $\mathbf{T}$ and randomness μ, output a randomized tag $\mathbf{T}' = (h^{\rho_1\mu}, h^{\rho_2\mu})$.

$\mathsf{ChangRep}(\sigma, (\mathbf{M}, \mathbf{N}), \mathbf{T}, (\mu, \upsilon))$: On input a representative $(\mathbf{M}, \mathbf{N}) \in [(\mathbf{M}, \mathbf{N})]_{\mathcal{R}_{\mathsf{TDH}}}$, $\mathbf{T} \in [\mathbf{T}]_{\mathcal{R}_\tau}$, signature $\sigma = (h, b, s)$, and $(\mu, \upsilon) \in (\mathbb{Z}_p^*)^2$, output:

$$\sigma' = \left(h' \leftarrow h^{\mu\upsilon}, b' \leftarrow b^{\mu}, s' \leftarrow s^{\mu\upsilon}, \mathbf{T}' \leftarrow \mathsf{ConvertTag}(\mathbf{T}, \mu)\right),$$

which is a valid signature for new representatives $(\mathbf{M}^{\mu\upsilon} = \mathbf{M}', \mathbf{N}^{\upsilon} = \mathbf{N}') \in [(\mathbf{M}, \mathbf{N})]_{\mathcal{R}_{\mathsf{TDH}}}$ and $\mathbf{T}' = (h^{\rho_1\mu}, h^{\rho_2\mu}) \in [\mathbf{T}]_{\mathcal{R}_\tau}$.

$\mathsf{ConvertSK}(\mathsf{sk}, \omega) \rightarrow \mathsf{sk}'$: On input a sk and key converter $\omega \in \mathbb{Z}_p^*$, output a new secret key as $\mathsf{sk}' = \mathsf{sk} \cdot \omega$.

$\mathsf{ConvertVK}(\mathsf{vk}, \omega) \rightarrow \mathsf{vk}'$: On input a vk and key converter $\omega \in \mathbb{Z}_p^*$, output $\mathsf{vk}' = \mathsf{vk}^{\omega} = (\hat{X}^{\omega}, \hat{Y}_1^{\omega}, \hat{Y}_2^{\omega}, \hat{Z}_1^{\omega}, \hat{Z}_2^{\omega})$.

$\mathsf{ConvertSig}(\mathsf{vk}, (\mathbf{M}, \mathbf{N}), \sigma, \mathbf{T}, \omega) \rightarrow \sigma'$: On input a vk, message $(\mathbf{M}, \mathbf{N})$, signature σ with tag $\mathbf{T}$, and key converter $\omega \in \mathbb{Z}_p^*$, returns a new signature σ' as: $\sigma' = (h, b^{\omega}, s^{\omega})$.

Note that one can reduce the number of paring operations in VerifyAggr by using batching verification techniques (cf. [34]).

Theorem 17 (Privacy). *Our construction is origin-hiding of ConvertSig (Def. 5), public key class-hiding (Def. 3), and provides perfect adaption of signatures (Def. 15).*

Theorem 18 (Unforgeability). *Our construction is EUF-CMA secure regarding the definition 13 in the generic group model for Type-III bilinear groups.*

The proofs of Theorem 18 and Theorem 17 are provided in the full version [47].

5 APPLICATION TO AC

As our core application we present Issuer-Hiding Multi-Authority Anonymous Credentials (IhMA). In a multi-authority setting [40], credentials come from ℓ-different credential issuers. Naively, the showing of credentials requires ℓ-independent credentials to be shown. This can be overcome [40] by leveraging aggregate signatures, obtaining a compact AC system with compact-size credentials, and showing costs. However, verifying a user's credentials needs knowledge of all issuers' verification keys, which might violate user privacy. Thus, in the vein of [6] we introduce the issuer-hiding property for multi-authority credentials. We recall that here the verifier can define a set of acceptable issuers in an ad-hoc manner. Then a user can prove that the subset of credentials shown were issued by acceptable issuers without revealing which credential corresponds to which issuer. This is an important feature, especially in multi-authority settings where disclosing issuer keys can reveal too much information compared to a single issuer setting and already lead to identification of the user.

5.1 Formal Definition

Our definition supports multiple users $(u_j)_{j\in[\ell]}$ and multiple credential issuers $(\mathsf{CI}_j)_{j\in[\ell]}$. An issuer can generate a key pair of secret and verification keys ($i$sk, $i$vk) via IKeyGen(). Similarly, users runs UKeyGen() to generate a user key pair ($u$sk, $u$vk). Each issuer can then issue a credential (cred) on an attribute (a) or attribute-set (A) to a user who can verify the received credential locally. Indeed, when we use AtoSa, we consider an attribute a (i.e., the attribute set includes only one attribute); when we use ATMS, we consider an attribute set, A. We use the notation A, to define security and formal definitions for consistency of definitions.

Users can then use the CredAggr algorithm to aggregate all credentials and create a single credential valid for all attributes and verification keys. To define the set of accepted issuers, a verifier generates a key-policy *pol* using GenPolicies (it is known as Presentation policies in [6]), which can be checked for well-formedness by

everyone. Finally, with an aggregate credential (disclosing a subset attributes D) and some key-policy pol from the verifier, a user uses Show to derive a proof, which a verifier can verify.

*Definition 19 (Issuer-Hiding Multi-Authority Credentials (*IhMA*)).* An IhMA is defined by the following algorithms/protocols:

- Setup: On input a security parameter λ, output public parameters pp (implicit input to all algorithms) .
- IKeyGen: Generate a key pair $(i\mathsf{sk}, i\mathsf{vk})$ for an issuer i.
- UKeyGen: Take a message-key set S, generate a user key pair $(u\mathsf{sk}, u\mathsf{vk})$ which acts as user's identity and auxiliary data aux.
- Issuance: In this protocol, an issuer i associated to $(i\mathsf{sk}, i\mathsf{vk})$ creates a credential cred on an attributes-set A to a user u associated to $(u\mathsf{sk}, u\mathsf{vk})$ as follows:

$$[\mathsf{CredObtain}(u\mathsf{sk}, i\mathsf{vk}, \mathsf{A}) \leftrightarrow \mathsf{CredIssue}(i\mathsf{sk}, u\mathsf{vk}, \mathsf{A})] \rightarrow \mathsf{cred}$$

- CredAggr: Take as input a $u$sk of user and a list of credentials $(i\mathsf{vk}, \mathsf{A}_i, \mathsf{cred}_i)$ for $i \in [\ell]$ and output an aggregated credential cred of attributes-set $\{\mathsf{A}_i\}_{i\in[\ell]}$:

$$\mathsf{CredAggr}\left(u\mathsf{sk}, \{(i\mathsf{vk}, \mathsf{A}_i, \mathsf{cred}_i)\}_{i\in[\ell]}\right) \rightarrow \mathsf{cred}$$

- GenPolicies: A verifier with the secret key $v$sk can define policies defining sets of issuers $\{\mathsf{ivk}\}_{i\in[n]}$ they are willing to accept for certain Show sessions, we have:

$$\mathsf{GenPolicy}(v\mathsf{sk}, \{\mathsf{ivk}\}_{i\in[n]}) \rightarrow pol, \text{ where } n \leq \ell$$

  Note that pol defines the sets of accepted issuers by a verifier, but not which attributes a verifier needs to disclose. Thus, pol can be reused for multiple contexts, reducing the number of policies.
- Show: In this protocol, a user u with $(u\mathsf{sk}, u\mathsf{vk})$ runs CredShow and interacts with a verifier running CredVerify to prove that she owns a valid credential cred on disclosed attribute sets $D \subseteq \{\mathsf{A}_i\}_{i\in[\ell]}$ issued respectively by one or some credential issuers in pol:

$$\begin{bmatrix}\mathsf{CredShow}(u\mathsf{sk}, pol, \{(i\mathsf{vk}, \mathsf{A}_i)\}_{i\in[\ell]}, \mathsf{cred}, D) \leftrightarrow \\ \mathsf{CredVerify}(pol, (i\mathsf{vk}_i)_{i\in[\ell]}, D)\end{bmatrix} \rightarrow (0, 1)$$

Due to the lack of space we refer to the full version [47] for our security model.

5.2 Constructions

Now we are ready to describes our two constructions of IhMA, the first being based on AtoSa (Def. 1) and SPSEQ [35] and the second based on ATMS (Def. 12), a set commitment scheme SC [35, 48], and SPSEQ. To enhance users' privacy and prevent issuers from learning attributes issued by other issuers, we change how aux for the signatures is computed. In particular, we commit to the attributes (messages) instead of including them in plaintext. For example, this can be achieved using a hash-based commitment scheme, where a commitment value c is generated by computing $c := H'(a, r)$ with H' being a hash function modeled as a random oracle, a being the attributing being committed to, and r a sufficiently large random value. When issuing a credential, users can reveal the relevant message (attribute) a, the opening o, and the commitment value c. The signer then verifies if the c is correct for a and o before issuing the corresponding credential. We modify GenAuxTag(S) and VerifyAux in AtoSa and ATMS as follows:

- GenAuxTag(S): Given $S = \{(m_j, \mathsf{vk}_j)_{j\in[\ell]}\}$, choose $(\rho_1, \rho_2) \xleftarrow{\$} \mathbb{Z}_p$, set $c = P^{\rho_1}||P^{\rho_2}||(c_{m_j}||\mathsf{vk}_j)_{j\in[\ell]}$, where c_{m_j} is a hash commitment to j'th message and all vk are distinct. Output aux $= (c, o_j)$ and tag $\tau = ((\rho_1, \rho_2), (T_1 = h^{\rho_1}, T_2 = h^{\rho_2}))$ with $h = H(c)$.
- VerifyAux(sk, aux, τ, m_j) Parse aux as (c, o). Check that $\tau \in t$ (i.e., that c has the form: $P^{\rho_1}||P^{\rho_2}||...$) check that c_j exists such that $(c_j, \mathsf{vk}) \in t$ and $\mathsf{Open}(c_j, o, m_j) = 1$ where vk is a verification key related to sk (in the same equivalence class). Also check that no other vk in aux has the same equivalence class as sk.

In our IhMA schemes, tags are user identities and are used to verify the user before issuing attributes.

5.2.1 AtoSa *based* IhMA *Construction in Fig. 4.* Here, every issuer creates a credential (signature) σ_{1i} on an attribute a_i for the user u with tag τ (and the respective aux) verified with $i$vk by the AtoSa scheme. We cannot reveal the secret part of the tag to signers (issuers) as this would violate the security of the user. To obtain a credential through the Issuing protocols, a user is required to disclose the public parts of tag as identity to the issuer and then authenticate their identity via a ZKPOK.

Interactive signing. We can adapt the signing in a way that signers (issuers) don't learn (ρ_1, ρ_2) as follows:

- u sends $(\mathsf{aux}, (h, \mathbf{T}), \pi)$, where $\mathsf{aux} = P^{\rho_1}||P^{\rho_2}||(c_{m_i}, \mathsf{vk}_i)_{i\in[n]}$ and
  $\pi = \mathsf{ZKPOK}\{(\rho_1, \rho_2) : T_1 = h^{\rho_1} \wedge T_2 = h^{\rho_2} \wedge u_1 = P^{\rho_1} \wedge u_2 = P^{\rho_2}\}$.
- Signer (issuer) checks if proof π is valid and if so outputs $(h' = h^{\rho_1}, s = (h^{\rho_1})^{x_j + y_{1j}\cdot m_j} \cdot (h^{\rho_2})^{y_{2j}})$

We note that this interactive signing outputs signatures that are identical to that output by Sign and this is used in Issuance. For the Show protocol, we assume that verifier(s) have signed all accepted issuer keys using an SPSEQ scheme [35]. A user u can take pol and the set of disclosed credentials D, aggregates the respective credentials (signatures) and randomizes the aggregated signature and tag. We note that alternatively, a user could already after Issuance aggregate all credentials to a constant-size (single) credential and then in Show protocol can provide a ZKPOK of the signature and selectively disclose the required attributes (as originally done for PS signatures in [52]). This also yields constant size credentials as noted in Table 1. We stick with the former approach here as it is more efficient for showing credentials, but one can easily switch to the other option. Moreover, In $\mathsf{IhMA}_{\mathsf{AtoSa}}$, only one attribute per vk can be issued. However, if an issuer needs to issue multiple attributes, they can easily generate multiple vks.

To hide the issuer's keys, u randomizes them using a random ω and adapts the signature for these randomized keys using ConvertSig. So far, we have created a compact randomized credential (proof) for attributes in D where issuer verification keys of this signature are hidden. The next step is to show that these random verification keys correspond to those keys signed by the verifier (using SPSEQ signatures) in pol. In this direction, u first collects signatures in pol according to issuer keys that are needed in the proof. Then u runs ChangRep of SPSEQ to randomize messages (which are issuer public keys) and signatures with the same randomness ω used in convert, i.e., randomized keys. Randomized issuer keys in a credential match with the messages signed by verifier in pol.

Finally, u uses the randomized tag as a pseudonym for communication and provides a ZKPOK of secret part of tag (secret keys and randomness) used in the credentials.

5.2.2 ATMS *based* IhMA *Construction in Fig. 5.* We use the framework in [35] in which one can combine mercurial or SPSEQ with a set commitment such that a credential is a signature on set commitment SC. One can then open a subset of messages from this commitment while randomizing both set commitment and signature together. This provides unlinkability and selective disclosure at the same time (see [35]). Unlike the previous construction, we can aggregate credentials immediately after receiving them and have a constant-size credential but still avoid zero-knowledge proof of a signature in showing protocol (because of compatibility of EQ message relation of ATMS and SC randomization).

In the Show protocol, similar to the previous construction, u first collects the signatures required to prove the attributes D from pol. Then, for issuer-hiding similar to AtoSa it randomizes these SPSEQ signatures using ChangRep of SPSEQ with ω. For preparing a proof for D, a user (u) randomizes issuer verification keys in credentials using ConvertVK and converts the ATMS signature using ConvertSig with ω. Subsequently, u randomizes the signature with a tag using ChangRep. Finally, u opens a subset of attributes D from the set commitments. Now a verifier can check if these attributes are in the set commitments signed by some issuers in pol. Same as in the first construction, since all issuer keys are randomized due to the SPSEQ signature the issuers are hidden. We run a ZKPOK to prove that u knows all secret values related to the randomized tag like before. The only point left is the signing of set commitments, which is defined in one source group in [35], but we need both groups. Subsequently, we show how one can combine set commitments with a tag-based DH message space.

Set commitments for $\mathcal{M}^H_{\mathsf{TDH}}$. The main point here is that we need to convert the set commitments space to $\mathcal{M}^H_{\mathsf{TDH}}$, which can be smoothly done as follows: In addition to credentials issuers, we also define a Trusted Authority TA who holds the trapdoor α of the set commitment scheme and can create commitments for the attributes of users who want to register in the system. WLOG, let us for simplicity assume only one attribute set $\mathsf{A} = (\mathsf{A}, \eta)$, where we have a fixed constant η which is never opened in practice and it is the same for all (it is just required for anonymity). It works as:

- The user sends a tag $\mathbf{T}$ and aux to TA.
- TA computes a set commitment in both groups ($\mathbf{C} = (C_1, C_2), \hat{\mathbf{C}} = (\hat{C}_1, \hat{C}_2)$) (i.e., $(\mathbf{M}, \mathbf{N})$) with tag, where $(C_2, \hat{C}_2)$ are dummy commitments for a fixed constant η and the other one for the (real) attribute set A. More precisely: TA computes the commitment in $\mathbb{G}_1$ to base h^{ρ_i} and the one in $\mathbb{G}_2$ in base $\hat{P}$: $C_1 = (h^{f_{\mathsf{A}}(\alpha)})^{\rho_1}$, $\hat{C}_1 = \hat{P}^{f_{\mathsf{A}}(\alpha)}$, $C_2 = (h^{\eta})^{\rho_2}$ and $\hat{C}_2 = \hat{P}^{\eta}$ such that such that we have $\bigwedge_{i\in[2]} e(T_i, \hat{C}_i) = e(C_i, \hat{P})$, where $h = H(c)$, aux $= (c, o)$, $c = P^{\rho_1}||P^{\rho_2}||(c_{\mathsf{A}_i}||\mathsf{vk}_j)_{j\in[2]}$, returns $(\mathbf{C}, \hat{\mathbf{C}})$. Note that $c_{\mathsf{A}} := H'(\mathsf{A}, r)$.

Note that α is a trapdoor kept by TA, but TA does not need to know (ρ_1, ρ_2) (e.g., C_i be computed as $(T_1)^{f_{\mathsf{A}}(\alpha)}$). A multiparty computation protocol can also be used to hide other user details from TA. A user can first randomize set commitment exactly like our tag-based message with (μ, υ) as $(\mathbf{C}^{\mu\upsilon}, \hat{\mathbf{C}}^{\upsilon})$ and use υ as opening information to open any subset values from $\hat{C}_1$ and still verify as follows: verifying the OpenSubset works $e(P, \hat{C}_1) = e(P^{f_D(\alpha)}, W)$. Consequently, we do not need any fundamental change on SC construction, and it works as stated in [48]. In our construction, we make it explicit as:

- $\mathsf{SC.Commit}_3(\mathsf{A}, \alpha, \mathbf{T}, h) \rightarrow ((\mathbf{C}, \hat{\mathbf{C}}), O)$: On input a set $\mathsf{A} = (\mathsf{A}, \eta)$, $\mathbf{T}$ and h, compute a commitment: $C_1 = (T_1^{f_{\mathsf{A}}(\alpha)})$, $\hat{C}_1 = \hat{P}^{f_{\mathsf{A}}(\alpha)}$, $C_2 = (T_2^{\eta})$ and $\hat{C}_2 = \hat{P}^{\eta}$, output $((\mathbf{C}, \hat{\mathbf{C}}), O)$ with $O \leftarrow \perp$.

Now, we can use the same technique as AtoSa to not reveal (ρ_1, ρ_2) to issuers when signing the above commitments $(\mathbf{C}, \hat{\mathbf{C}})$ as follows:

Interactive signing. We can adapt the signing in a way that signers (issuers) don't learn (ρ_1, ρ_2) as follows:

- u sends $(\mathsf{aux}, \mathbf{T}, (\mathbf{C}, \hat{\mathbf{C}}), \pi)$, where $\pi = \mathsf{ZKPOK}\{(\rho_1, \rho_2) : T_1 = h^{\rho_1} \wedge T_2 = h^{\rho_2} \wedge u_1 = P^{\rho_1} \wedge u_2 = P^{\rho_2}\}$, where P^{ρ_1} and P^{ρ_2} are in aux.
- Signer (issuer) checks if proof π is valid and if so outputs $(h = H(c), b = \prod T_i^{z_i}, s = (h^x \cdot \prod_{i\in[2]} (C_i)^{y_i}))$.

Again we note that this interactive signing outputs signatures that are identical to that output by Sign and this is used in Issuance.

Achieving constant-size credentials. This can be achieved by following these steps: 1) Users can obtain the (h^{α_i}) values from the TA instead of the commitments. 2) During the issuing phase, users can aggregate all the credentials received from issuers. 3) The commitments can then be recomputed using randomness and the obtained information, eliminating the need to store them. Note that in this case the size of the |**Show**| operation will become linear with respect to N instead of K.

Theorem 20. *The above* IhMA *constructions in Fig. 5 and in Fig. 4 are correct, unforgeable, anonymous, and issuer-hiding.*

To prove the anonymity of ATMS, we need to define a variant of the uber assumption, which we present in the full version [47] along with the proof of Theorem 20. Moreover, in the full version [47] we discuss how additional features can be obtained via slight modifications of the so far presented approaches.

6 IMPLEMENTATION AND EVALUATION

In the following we present our evaluation based on a Python library in which we implement our primitives ATMS and AtoSa as well as our IhMA protocols (Fig. 5 and Fig. 4). Our implementation is based upon the `bplib` library[12] and petlib [13] with `OpenSSL` bindings[14]. We use the popular pairing friendly curve BN256 which provides efficient type 3 bilinear groups at a security level of around 100 bits. Our measurements have been performed on an Intel Core i5-6200U CPU at 2.30 GHz, 16 GB RAM running Ubuntu 20.04.3.

Benchmark of Primitives. Table 2 shows the mean of the execution time of each algorithm over 500 runs such that AggrSign and VerifyAggr are computed assuming two signers ($n = 2$); the other algorithms are independent of n. ChR/Rnd stands for ChangRep and signature randomization (RandSign) for the ATMS and AtoSa, respectively. PC stands for Pre-Computation, and in ATMS it includes

[12] https://github.com/gdanezis/bplib
[13] https://github.com/gdanezis/petlib
[14] https://github.com/dfaranha/OpenPairing

- $\mathsf{Setup}(1^\lambda)$: Run $\mathsf{pp}_{\mathsf{AtoSa}} \leftarrow \Sigma_1.\mathsf{Setup}(1^\lambda) \wedge \mathsf{pp}_{\mathsf{SPSEQ}} \leftarrow \Sigma_2.\mathsf{Setup}(1^\lambda)$, output $\mathsf{pp} = (\mathsf{pp}_{\mathsf{AtoSa}}, \mathsf{pp}_{\mathsf{SPSEQ}})$. The attribute space is $\mathbb{Z}_p$.
- $\mathsf{UKeyGen}(\mathsf{pp}, \mathsf{S})$: Run $(\{\mathsf{aux}_j\}, (\tau, \mathbf{T})) \leftarrow \mathsf{GenAuxTag}(\mathsf{pp}, S)$, and return $(usk = \tau, uvk = \mathbf{T}, \{\mathsf{aux}_j\})$ to u.
- $\mathsf{IKeyGen}(\mathsf{pp})$: Generate $(\mathsf{sk}, \mathsf{vk}) \xleftarrow{\$} \Sigma_1.\mathsf{KeyGen}(\mathsf{pp})$, return $(isk = \mathsf{sk}, ivk = \mathsf{vk})$ to an issuer i.
- Issuance: On input $(\mathbf{T}, \mathsf{aux}_i, a_i)$, u and each issuer i act as follows for an attribute a_i:
  - u sends $(\mathbf{T}, \mathsf{aux}_i, \pi)$, to an issuer i, where π is a zero knowledge proof that the user knows the secret part of the given tag.
  - Issuer checks π is valid and runs $\sigma_i \leftarrow \Sigma_1.\mathsf{Sign}(isk, \mathbf{T}, \mathsf{aux}_i, a_i)$ and outputs (σ_i, a_i) to u or aborts if Sign outputs $\perp$.
  - u takes $(ivk, \mathsf{cred}_i = (a_i, \sigma_i))_{i\in[\ell]}$, checks $\Sigma_1.\mathsf{Verify}(ivk, a_i, \mathsf{cred}_i)_{i\in[\ell]}$, and saves $\mathsf{cred} = \{\mathsf{cred}_i = (\sigma_i, \tau), \mathsf{A}\}_{i\in[\ell]}$, where $\mathsf{A} = (a_i)_{i\in[\ell]}$.
- Gen-Policies: Generate a key pair $(vsk, vpk) \leftarrow \Sigma_2.\mathsf{KeyGen}(\mathsf{pp})$, run $\sigma_{2i} \leftarrow \Sigma_2.\mathsf{Sign}(vsk, ivk)$ for $i \in I$ where ivk is a message vector for SPSEQ, return $pol = (vvk, (ivk, \sigma_{2i})_{i\in[I]})$.
- Show: On input $\mathsf{cred} = \{(\sigma_i, \tau, \mathsf{A})_{i\in[\ell]}\}$, $pol = (vvk, (ivk, \sigma_{2i})_{i\in[I]})$, an D (a set of attributes) from $n \subseteq I$ issuers ($|D| = n$), u prepares a proof for D as:
  (1) Run $\sigma \leftarrow \Sigma_1.\mathsf{AggrSign}(\mathbf{T}, (ivk, a_i, \sigma_i))_{i\in[D]}$ with $\mathsf{avk} = \{ivk\}_{i\in[D]}$. For $\omega \in \mathbb{Z}_p^*$, run $\mathsf{avk}' \leftarrow \Sigma_1.\mathsf{ConvertVK}(\mathsf{avk}, \omega)$, $\sigma' \leftarrow \Sigma_1.\mathsf{ConvertSig}(\mathsf{avk}, D, \mathbf{T}, \sigma, \omega)$, and randomize $(\sigma'', \mathbf{T}') \leftarrow \Sigma_1.\mathsf{RandSign}(\mathsf{vk}, \mathbf{T}, m, \sigma', v)$ for $v \in \mathbb{Z}_p^*$.
  (2) Run $(\sigma'_{2i}, \mathsf{avk}') \xleftarrow{\$} \Sigma_2.\mathsf{ChangRep}(\mathbf{M}_i = \mathsf{vk}_i, \sigma_{2i}, \omega)_{i\in[n]}$ where avk' is the same as $\mathsf{avk}' \leftarrow \Sigma_1.\mathsf{ConvertVK}$.
  (3) Prove in zero knowledge that the user knows the secret key for the tag $\mathbf{T}'$, yielding π, send $(\sigma'', \mathsf{Nym} = \mathbf{T}', \sigma'_{2i}, \pi)_{i\in[n]}$ to a verifier V.
- CredVerify: Output 1, if $\pi \wedge \Sigma_1.\mathsf{VerifyAggr}(\mathsf{avk}', \mathbf{T}', D, \sigma') \wedge \Sigma_2.\mathsf{Verify}(vvk, \mathbf{M}, \sigma'_2) = 1$, where $\mathbf{M} = \mathsf{avk}'$ and $\mathbf{T}' = \mathsf{Nym}$. Output 0 if this check fails.

Figure 4: Our IhMA scheme (Σ_1 and Σ_2 denote AtoSa and SPSEQ [35], respectively)

- $\mathsf{Setup}(1^\lambda)$: Run $\mathsf{pp}_{\mathsf{ATMS}} \leftarrow \Sigma_1.\mathsf{Setup}(1^\lambda) \wedge \mathsf{pp}_{\mathsf{SPSEQ}} \leftarrow \Sigma_2.\mathsf{Setup}(1^\lambda) \wedge \mathsf{pp}_{\mathsf{SC}} \leftarrow \mathsf{SC.Setup}$, output $\mathsf{pp} = (\mathsf{pp}_{\mathsf{ATMS}}, \mathsf{pp}_{\mathsf{SPSEQ}}, \mathsf{pp}_{\mathsf{SC}})$.
- $\mathsf{IKeyGen}(\mathsf{pp})$: Generate $(\mathsf{sk}, \mathsf{vk}) \xleftarrow{\$} \Sigma_1.\mathsf{KeyGen}(\mathsf{pp})$, return $(isk = \mathsf{sk}, ivk = \mathsf{vk})$ to an issuer i.
- $\mathsf{UKeyGen}(\mathsf{pp}, S)$: Run $((\tau, \mathbf{T}), \mathsf{aux}) \leftarrow \mathsf{GenAuxTag}(S)$, and return $(usk = \tau, uvk = \mathbf{T})$ to u.
  Then, TA and u interact to computes $((\hat{\mathbf{C}}_i, \mathbf{C}_i)_{i\in[\ell]}) \leftarrow \mathsf{SC.Commit}_3(\mathsf{A}_i, \alpha, \mathbf{T})$, for all attribute sets.
- Issuance: The interaction between an issuer i and a user u for one attribute-set $\mathsf{A} \in \mathbb{Z}_p$ and $(\mathbf{C}, \hat{\mathbf{C}})$ acts as follows:
  - u hands over $(\mathbf{T}, (\mathbf{C}, \hat{\mathbf{C}}), \mathsf{aux}_i, \pi)$ to an issuer i, where π is zero knowledge proof the secret parts of the tag.
  - An issuer i checks that the proof is correct, then runs $\sigma \leftarrow \Sigma_1.\mathsf{Sign}(isk, \mathbf{T}, \mathsf{aux}_i, (\mathbf{C}, \hat{\mathbf{C}}))$, and outputs $(\mathsf{A}, \mathbf{T}, \sigma) = \mathsf{cred}_i$.
  - u takes (ivk, cred_i) for $i \in [\ell]$, checks $\Sigma_1.\mathsf{Verify}(ivk, \mathbf{T}, (\mathbf{C}_i, \hat{\mathbf{C}}_i), \sigma_i)_{i\in[\ell]} = 1$, and outputs $\{\mathsf{cred} = (\sigma_i, \tau), (\mathsf{A}_i, \mathbf{C}_i, \hat{\mathbf{C}}_i)_{i\in[\ell]}\}$.
- Gen-Policies: Generate a key pair $(vsk, vpk) \leftarrow \Sigma_2.\mathsf{KeyGen}(\mathsf{pp})$, run $\sigma_{2i} \leftarrow \Sigma_2.\mathsf{Sign}(vsk, ivk)$ for $i \in I$, return $pol = (vvk, (ivk, \sigma_{2i})_{i\in[I]})$.
- Show: On input $\mathsf{cred} = \{(\sigma_i, usk, \mathsf{A}_i)_{i\in[\ell]}\}$, $pol = (vvk, (ivk, \sigma_{2i})_{i\in[I]})$, and $D \subseteq \mathsf{A}$ from $n \subseteq I$ issuers, u prepares a proof for D as:
  (1) Run $(\sigma'_{2i}, \mathsf{avk}') \leftarrow \Sigma_2.\mathsf{ChangRep}(\mathbf{M}_i = \mathsf{vk}_i, \sigma_{2i}, \omega)_{i\in[n]}$ for $\omega \in \mathbb{Z}_p^*$.
  (2) Run $\sigma \leftarrow \Sigma_1.\mathsf{AggrSign}(\mathbf{T}, (ivk, (\mathbf{C}_i, \hat{\mathbf{C}}_i), \sigma_i))_{i\in[n]}$. Convert credentials and issuer keys $\mathsf{avk}' \leftarrow \Sigma_1.\mathsf{ConvertVK}(\mathsf{avk}, \omega)$ and $\sigma' \leftarrow \Sigma_1.\mathsf{ConvertSig}(\mathsf{avk}, (\mathbf{C}, \hat{\mathbf{C}}), \sigma, \mathbf{T}, \omega)$.
  (3) Run $(\sigma', \mathbf{T}') \xleftarrow{\$} \Sigma_1.\mathsf{ChangRep}(\sigma, (\mathbf{M}_i, \mathbf{N}_i)_{i\in[n]}, \mathbf{T}, (\mu, v))$ for (μ, v), where $(\mathbf{M}_i, \mathbf{N}_i) = (\mathbf{C}_i, \hat{\mathbf{C}}_i)$, and σ' is valid for $(\mathbf{C}'_i = \mathbf{C}_i^{\mu v}, \hat{\mathbf{C}}'_i = \hat{\mathbf{C}}_i^{v})_{i\in[n]}$. Create witnesses for attributes $W_j \leftarrow \mathsf{SC.OpenSubset}(\hat{C}_{1j}, A_j, O_j, d_j)$ for $j \wedge d_j \in D$. Aggregate witness $W \leftarrow \mathsf{SC.AggregateAcross}(\{\hat{C}_{1j}, d_j, W_j\}_{j\in[\ell]})$, randomize $W' \leftarrow W^{\mu v}$.
  (4) Prove in zero knowledge that the user knows the secret key for the tag $\mathbf{T}'$, yielding π, send $(\sigma', W', \mathbf{T}', \sigma'_{2i}, \pi, \mathbb{M} = \{(\mathbf{C}'_i, \hat{\mathbf{C}}'_i)\})_{i\in[n]}$ to V.
- CredVerify: Output 1, if $\pi \wedge \Sigma_1.\mathsf{VerifyAggr}(\mathsf{avk}', \mathbf{T}', \mathbb{M}, \sigma') \wedge \Sigma_2.\mathsf{Verify}(vvk, \mathbf{M}, \sigma'_2) \wedge \mathsf{SC.VerifySubset}(\mathbf{C}', D, W') = 1$, where $\mathbf{M} = \mathsf{avk}'$ is verified by vvk.

Figure 5: Our IhMA scheme (Σ_1 and Σ_2 denote ATMS and SPSEQ [35], respectively)

converting messages to the $\mathcal{M}^H_{\mathsf{TDH}}$ message space and generating tags. While in AtoSa, PC includes generating tags and aux using Pedersen commitment, but note that one could also use a hash based commitment instead. We can observe that signing is faster than verifying the signature – due to the pairing operation in the latter. Moreover, verification of ATMS is slower than AtoSa because of additional pairing operations that are needed to check if messages are in $\mathcal{M}^H_{\mathsf{TDH}}$. We increase the number (n) of signers from 2 to 10

Table 2: Running times for ATMS and AtoSa (ms)

	PC	Sign	Verify	Convert	ChR/Rnd	AggrSign	VerifyAggr
AtoSa	6	2,5	8,4	4	2,7	0.005	9
ATMS	8.6	3	33	5,4	7,4	0.01	72

and show the running time in Fig. 6. Since aggregation is almost free (for $n = 10$ is 0.05 ms), we omit it. We should also note that the result are stated without considering VerifyAux algorithm.

IhMA **Benchmarks**. IhMA is based upon Schnorr-style discrete logarithm ZKPOK. Our library supports Damgård's technique [28]

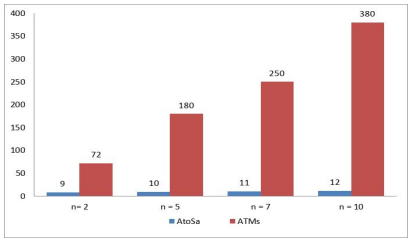


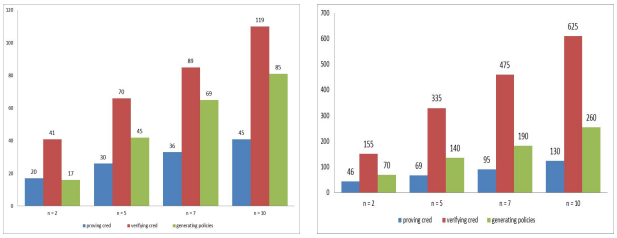
Figure 6: Running times of VerifyAggr in ATMS & AtoSa (ms)

for obtaining malicious-verifier interactive zero-knowledge proofs of knowledge during the showing and also NIZK obtained via the Fiat-Shamir heuristic. We interpret signers as issuers here and also show n as a number of issuers involved in Showing. For example, $n = 2$ means showing two credentials from 2 different issuers.

Issuing. This protocol does not depend on n, and results are as follows: 1) For IhMA based on AtoSa, including generation of signature, tag, user keys, and aux, it takes 8 ms. 2) For IhMA based

on AtoSa, including generation of tag and encoding messages to $\mathcal{M}^H_{\mathsf{TDH}}$, with two attributes in each credential it takes 10 ms.

Showing. Fig. 7a shows the runtime of showing for IhMA based on AtoSa. In this experiment, we increase the number of issuers n from 2 to 10 and assume that all attributes are disclosed during verification (the worst-case scenario). Each issuer issues only one attribute, giving a total of n attributes. Fig. 7b shows the time for showing a credentials of IhMA based on ATMS. Here, we have a different setting; we can encode a set of attributes in a credential as we use set commitments. For our evaluation, we have the following parameters: n represents the number of the issuer, t the number of attributes in each set (each credential issued), $d < t$ is the number of disclosed attributes from each attribute set A in the respective commitment C. Here we increase n from 2 to 10, set $t = 2$, and $d = 1$. The total disclosed attributes length $|D| = d \cdot n$ and the total attribute $|A| = n \cdot t$ range from 2 to 10 and 4 to 20, respectively.



(a) Running times of $\mathsf{IhMA}_{\mathsf{AtoSa}}$ **(b) Running times of** $\mathsf{IhMA}_{\mathsf{ATMS}}$

Figure 7: Running times of IhMA (ms)

7 CONCLUSION AND OPEN QUESTIONS

This paper introduces the Issuer-Hiding Multi-Authority Anonymous Credentials (IhMA). **MA** means proving possession of attributes from multiple independent credential issuers requires the presentation of independent credentials. Meanwhile, **Ih** means verifying a user's credential does not require disclosing multiple issuers' public keys. Our proposed solution involves the development of two new signature primitives with versatile randomization features which are independent of interest: 1) Aggregate Signatures with Randomizable Tags and Public Keys (AtoSa) and 2) Aggregate Mercurial Signatures (ATMS), which extend the functionality of AtoSa to support the randomization of messages additionally.

Open Questions and Future Work. Finally, we still have several open questions that merit further investigation. 1) An interesting open question is whether it is possible to present constructions in a fully dynamic setting, i.e., there are no assumptions about prior knowledge of messages and verification keys, nor requirement for a stateful issuer to keep track of the signed information aux. 2) Revocation is another intriguing avenue. While issuer revocation in our scheme is straightforward, as revoked issuers can be excluded from the key policy, user revocation poses greater challenges. The user revocation within our framework, and for issuer-hiding anonymous credentials in general, are an interesting future work.

Acknowledgements. This work has in part been carried out within the scope of Digidow, the Christian Doppler Laboratory for Private Digital Authentication in the Physical World. Omid Mir acknowledge financial support by the Austrian Federal Ministry for Digital and Economic Affairs, the National Foundation for Research, Technology and Development, the Christian Doppler Research Association, 3 Banken IT GmbH, ekey biometric systems GmbH, Kepler Universitätsklinikum GmbH, NXP Semiconductors Austria GmbH and Co KG, Österreichische 24 Staatsdruckerei GmbH, and the State of Upper Austria. Daniel Slamanig was supported by the European Commission through ECSEL Joint Undertaking (JU) under grant agreement n°826610 (Comp4Drones), the European Union through the Horizon Europe research programme under grant agreement n°101073821 (Sunrise) and by the Austrian Science Fund (FWF) and netidee SCIENCE under grant agreement P31621-N38 (Profet). Anna Lysyanskaya and Scott Griffy are supported by NSF Awards 2247305, 2154941 and 2154170, as well as funding from the Peter G. Peterson Foundation and Meta.

REFERENCES

[1] Jae Hyun Ahn, Matthew Green, and Susan Hohenberger. 2010. Synchronized aggregate signatures: new definitions, constructions and applications. In *ACM CCS 2010*, Ehab Al-Shaer, Angelos D. Keromytis, and Vitaly Shmatikov (Eds.). ACM Press, 473–484. https://doi.org/10.1145/1866307.1866360

[2] Man Ho Au, Willy Susilo, and Yi Mu. 2006. Constant-Size Dynamic k-TAA. In *SCN 06 (LNCS, Vol. 4116)*, Roberto De Prisco and Moti Yung (Eds.). Springer, Heidelberg, 111–125. https://doi.org/10.1007/11832072_8

[3] Foteini Baldimtsi and Anna Lysyanskaya. 2013. Anonymous credentials light. In *ACM CCS 2013*, Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung (Eds.). ACM Press, 1087–1098. https://doi.org/10.1145/2508859.2516687

[4] Mihir Bellare, Chanathip Namprempre, and Gregory Neven. 2007. Unrestricted Aggregate Signatures. In *ICALP 2007 (LNCS, Vol. 4596)*, Lars Arge, Christian Cachin, Tomasz Jurdzinski, and Andrzej Tarlecki (Eds.). Springer, Heidelberg, 411–422. https://doi.org/10.1007/978-3-540-73420-8_37

[5] Johannes Blömer and Jan Bobolz. 2018. Delegatable Attribute-Based Anonymous Credentials from Dynamically Malleable Signatures. In *ACNS 18 (LNCS, Vol. 10892)*, Bart Preneel and Frederik Vercauteren (Eds.). Springer, Heidelberg, 221–239. https://doi.org/10.1007/978-3-319-93387-0_12

[6] Jan Bobolz, Fabian Eidens, Stephan Krenn, Sebastian Ramacher, and Kai Samelin. 2021. Issuer-Hiding Attribute-Based Credentials. In *International Conference on Cryptology and Network Security*. Springer, 158–178.

[7] Jan Bobolz, Fabian Eidens, Stephan Krenn, Sebastian Ramacher, and Kai Samelin. 2022. Issuer-Hiding Attribute-Based Credentials. Cryptology ePrint Archive, Report 2022/213. https://eprint.iacr.org/2022/213.

[8] Alexandra Boldyreva, Craig Gentry, Adam O'Neill, and Dae Hyun Yum. 2007. Ordered Multisignatures and Identity-Based Sequential Aggregate Signatures, with Applications to Secure Routing. Cryptology ePrint Archive, Report 2007/438. https://eprint.iacr.org/2007/438.

[9] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. 2003. Aggregate and Verifiably Encrypted Signatures from Bilinear Maps. In *EUROCRYPT 2003 (LNCS, Vol. 2656)*, Eli Biham (Ed.). Springer, Heidelberg, 416–432. https://doi.org/10.1007/3-540-39200-9_26

[10] Daniel Bosk, Davide Frey, Mathieu Gestin, and Guillaume Piolle. 2022. Hidden Issuer Anonymous Credential. *Proc. Priv. Enhancing Technol.* 2022, 4 (2022), 571–607. https://doi.org/10.56553/popets-2022-0123

[11] Xavier Boyen. 2008. The Uber-Assumption Family (Invited Talk). In *PAIRING 2008 (LNCS, Vol. 5209)*, Steven D. Galbraith and Kenneth G. Paterson (Eds.). Springer, Heidelberg, 39–56. https://doi.org/10.1007/978-3-540-85538-5_3

[12] Stefan Brands. 2000. *Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy*. MIT Press, Cambridge-London. http://www.credentica.com/the_mit_pressbook.html

[13] Ernie Brickell and Jiangtao Li. 2012. Enhanced Privacy ID: A Direct Anonymous Attestation Scheme with Enhanced Revocation Capabilities. *IEEE Trans. Dependable Secur. Comput.* 9, 3 (2012), 345–360. https://doi.org/10.1109/TDSC.2011.63

[14] Ernest F. Brickell, Jan Camenisch, and Liqun Chen. 2004. Direct Anonymous Attestation. In *ACM CCS 2004*, Vijayalakshmi Atluri, Birgit Pfitzmann, and Patrick McDaniel (Eds.). ACM Press, 132–145. https://doi.org/10.1145/1030083.1030103

[15] Jan Camenisch, Liqun Chen, Manu Drijvers, Anja Lehmann, David Novick, and Rainer Urian. 2017. One TPM to Bind Them All: Fixing TPM 2.0 for Provably Secure Anonymous Attestation. In *2017 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, 901–920. https://doi.org/10.1109/SP.2017.22

[16] Jan Camenisch, Manu Drijvers, Anja Lehmann, Gregory Neven, and Patrick Towa. 2020. Short Threshold Dynamic Group Signatures. In *SCN 20 (LNCS, Vol. 12238)*,

Clemente Galdi and Vladimir Kolesnikov (Eds.). Springer, Heidelberg, 401–423. https://doi.org/10.1007/978-3-030-57990-6_20

[17] Jan Camenisch, Maria Dubovitskaya, Kristiyan Haralambiev, and Markulf Kohlweiss. 2015. Composable and Modular Anonymous Credentials: Definitions and Practical Constructions. In *ASIACRYPT 2015, Part II (LNCS, Vol. 9453)*, Tetsu Iwata and Jung Hee Cheon (Eds.). Springer, Heidelberg, 262–288. https://doi.org/10.1007/978-3-662-48800-3_11

[18] Jan Camenisch and Anna Lysyanskaya. 2001. An Efficient System for Non-transferable Anonymous Credentials with Optional Anonymity Revocation. In *EUROCRYPT 2001 (LNCS, Vol. 2045)*, Birgit Pfitzmann (Ed.). Springer, Heidelberg, 93–118. https://doi.org/10.1007/3-540-44987-6_7

[19] Jan Camenisch and Anna Lysyanskaya. 2003. A Signature Scheme with Efficient Protocols. In *SCN 02 (LNCS, Vol. 2576)*, Stelvio Cimato, Clemente Galdi, and Giuseppe Persiano (Eds.). Springer, Heidelberg, 268–289. https://doi.org/10.1007/3-540-36413-7_20

[20] Jan Camenisch and Anna Lysyanskaya. 2004. Signature Schemes and Anonymous Credentials from Bilinear Maps. In *CRYPTO 2004 (LNCS, Vol. 3152)*, Matthew Franklin (Ed.). Springer, Heidelberg, 56–72. https://doi.org/10.1007/978-3-540-28628-8_4

[21] Jan Camenisch and Markus Stadler. 1997. Efficient Group Signature Schemes for Large Groups (Extended Abstract). In *CRYPTO'97 (LNCS, Vol. 1294)*, Burton S. Kaliski Jr. (Ed.). Springer, Heidelberg, 410–424. https://doi.org/10.1007/BFb0052252

[22] Jan Camenisch and Els Van Herreweghen. 2002. Design and Implementation of The Idemix Anonymous Credential System. In *ACM CCS 2002*, Vijayalakshmi Atluri (Ed.). ACM Press, 21–30. https://doi.org/10.1145/586110.586114

[23] Melissa Chase, Trevor Perrin, and Greg Zaverucha. 2020. The Signal Private Group System and Anonymous Credentials Supporting Efficient Verifiable Encryption. In *ACM CCS 2020*, Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna (Eds.). ACM Press, 1445–1459. https://doi.org/10.1145/3372297.3417887

[24] David Chaum. 1985. Security Without Identification: Transaction Systems to Make Big Brother Obsolete. *Commun. ACM* 28, 10 (1985), 1030–1044. https://doi.org/10.1145/4372.4373

[25] David Chaum. 1986. Showing Credentials Without Identification: Signatures Transferred Between Unconditionally Unlinkable Pseudonyms. In *EUROCRYPT'85 (LNCS, Vol. 219)*, Franz Pichler (Ed.). Springer, Heidelberg, 241–244. https://doi.org/10.1007/3-540-39805-8_28

[26] Valerio Cini, Sebastian Ramacher, Daniel Slamanig, Christoph Striecks, and Erkan Tairi. 2021. Updatable Signatures and Message Authentication Codes. In *PKC 2021, Part I (LNCS, Vol. 12710)*, Juan Garay (Ed.). Springer, Heidelberg, 691–723. https://doi.org/10.1007/978-3-030-75245-3_25

[27] Aisling Connolly, Pascal Lafourcade, and Octavio Perez Kempner. 2022. Improved constructions of anonymous credentials from structure-preserving signatures on equivalence classes. In *IACR International Conference on Public-Key Cryptography*. Springer, 409–438.

[28] Ronald Cramer, Ivan Damgård, and Philip D. MacKenzie. 2000. Efficient Zero-Knowledge Proofs of Knowledge Without Intractability Assumptions. In *PKC 2000 (LNCS, Vol. 1751)*, Hideki Imai and Yuliang Zheng (Eds.). Springer, Heidelberg, 354–372. https://doi.org/10.1007/978-3-540-46588-1_24

[29] Elizabeth Crites, Markulf Kohlweiss, Bart Preneel, Mahdi Sedaghat, and Daniel Slamanig. 2022. Threshold Structure-Preserving Signatures. Cryptology ePrint Archive, Paper 2022/839. https://eprint.iacr.org/2022/839 https://eprint.iacr.org/2022/839.

[30] Elizabeth C. Crites and Anna Lysyanskaya. 2019. Delegatable Anonymous Credentials from Mercurial Signatures. In *CT-RSA 2019 (LNCS, Vol. 11405)*, Mitsuru Matsui (Ed.). Springer, Heidelberg, 535–555. https://doi.org/10.1007/978-3-030-12612-4_27

[31] Alex Davidson, Ian Goldberg, Nick Sullivan, George Tankersley, and Filippo Valsorda. 2018. Privacy Pass: Bypassing Internet Challenges Anonymously. *PoPETs* 2018, 3 (2018), 164–180. https://doi.org/10.1515/popets-2018-0026

[32] Dominic Deuber, Matteo Maffei, Giulio Malavolta, Max Rabkin, Dominique Schröder, and Mark Simkin. 2018. Functional Credentials. *PoPETs* 2018, 2 (April 2018), 64–84. https://doi.org/10.1515/popets-2018-0013

[33] Jack Doerner, Yashvanth Kondi, Eysa Lee, abhi shelat, and LaKyah Tyner. 2023. Threshold BBS+ Signatures for Distributed Anonymous Credential Issuance. Cryptology ePrint Archive, Paper 2023/602. https://doi.org/10.1109/SP46215.2023.00120 https://eprint.iacr.org/2023/602.

[34] Anna Lisa Ferrara, Matthew Green, Susan Hohenberger, and Michael Østergaard Pedersen. 2009. Practical Short Signature Batch Verification. In *CT-RSA 2009 (LNCS, Vol. 5473)*, Marc Fischlin (Ed.). Springer, Heidelberg, 309–324. https://doi.org/10.1007/978-3-642-00862-7_21

[35] Georg Fuchsbauer, Christian Hanser, and Daniel Slamanig. 2019. Structure-Preserving Signatures on Equivalence Classes and Constant-Size Anonymous Credentials. *Journal of Cryptology* 32, 2 (April 2019), 498–546. https://doi.org/10.1007/s00145-018-9281-4

[36] Christina Garman, Matthew Green, and Ian Miers. 2014. Decentralized Anonymous Credentials. In *NDSS 2014*. The Internet Society.

[37] Essam Ghadafi. 2016. Short Structure-Preserving Signatures. In *CT-RSA 2016 (LNCS, Vol. 9610)*, Kazue Sako (Ed.). Springer, Heidelberg, 305–321. https://doi.org/10.1007/978-3-319-29485-8_18

[38] Rishab Goyal and Vinod Vaikuntanathan. 2022. Locally Verifiable Signature and Key Aggregation. Cryptology ePrint Archive, Report 2022/179. https://eprint.iacr.org/2022/179.

[39] Lucjan Hanzlik and Daniel Slamanig. 2021. With a Little Help from My Friends: Constructing Practical Anonymous Credentials. In *ACM CCS 2021*, Giovanni Vigna and Elaine Shi (Eds.). ACM Press, 2004–2023. https://doi.org/10.1145/3460120.3484582

[40] Chloé Hébant and David Pointcheval. 2022. Traceable Constant-Size Multi-authority Credentials. In *Security and Cryptography for Networks - 13th International Conference, SCN 2022, Amalfi, Italy, September 12-14, 2022, Proceedings (Lecture Notes in Computer Science, Vol. 13409)*, Clemente Galdi and Stanislaw Jarecki (Eds.). Springer, 411–434. https://doi.org/10.1007/978-3-031-14791-3_18

[41] Susan Hohenberger and Brent Waters. 2018. Synchronized Aggregate Signatures from the RSA Assumption. In *EUROCRYPT 2018, Part II (LNCS, Vol. 10821)*, Jesper Buus Nielsen and Vincent Rijmen (Eds.). Springer, Heidelberg, 197–229. https://doi.org/10.1007/978-3-319-78375-8_7

[42] Ben Kreuter, Tancrède Lepoint, Michele Orrù, and Mariana Raykova. 2020. Anonymous Tokens with Private Metadata Bit. In *CRYPTO 2020, Part I (LNCS, Vol. 12170)*, Daniele Micciancio and Thomas Ristenpart (Eds.). Springer, Heidelberg, 308–336. https://doi.org/10.1007/978-3-030-56784-2_11

[43] Kwangsu Lee, Dong Hoon Lee, and Moti Yung. 2013. Aggregating CL-Signatures Revisited: Extended Functionality and Better Efficiency. In *FC 2013 (LNCS, Vol. 7859)*, Ahmad-Reza Sadeghi (Ed.). Springer, Heidelberg, 171–188. https://doi.org/10.1007/978-3-642-39884-1_14

[44] Steve Lu, Rafail Ostrovsky, Amit Sahai, Hovav Shacham, and Brent Waters. 2006. Sequential Aggregate Signatures and Multisignatures Without Random Oracles. In *EUROCRYPT 2006 (LNCS, Vol. 4004)*, Serge Vaudenay (Ed.). Springer, Heidelberg, 465–485. https://doi.org/10.1007/11761679_28

[45] Anna Lysyanskaya. 2022. Security Analysis of RSA-BSSA. Cryptology ePrint Archive, Report 2022/895. https://eprint.iacr.org/2022/895.

[46] Anna Lysyanskaya, Silvio Micali, Leonid Reyzin, and Hovav Shacham. 2004. Sequential Aggregate Signatures from Trapdoor Permutations. In *EUROCRYPT 2004 (LNCS, Vol. 3027)*, Christian Cachin and Jan Camenisch (Eds.). Springer, Heidelberg, 74–90. https://doi.org/10.1007/978-3-540-24676-3_5

[47] Omid Mir, Balthazar Bauer, Scott Griffy, Anna Lysyanskaya, and Daniel Slamanig. 2023. Aggregate Signatures with Versatile Randomization and Issuer-Hiding Multi-Authority Anonymous Credentials. *Cryptology ePrint Archive* (2023).

[48] Omid Mir, Daniel Slamanig, Balthazar Bauer, and René Mayrhofer. 2023. Practical Delegatable Anonymous Credentials From Equivalence Class Signatures. *Proc. Priv. Enhancing Technol.* 2023, 3 (2023), 488–513. https://doi.org/10.56553/popets-2023-0093

[49] Omid Mir, Daniel Slamanig, and René Mayrhofer. 2023. Threshold Delegatable Anonymous Credentials with Controlled and Fine-Grained Delegation. *IEEE Transactions on Dependable and Secure Computing* (2023).

[50] Gregory Neven. 2008. Efficient Sequential Aggregate Signed Data. In *EUROCRYPT 2008 (LNCS, Vol. 4965)*, Nigel P. Smart (Ed.). Springer, Heidelberg, 52–69. https://doi.org/10.1007/978-3-540-78967-3_4

[51] Christian Paquin and Greg Zaverucha. 2013. U-Prove Cryptographic Specification V1.1 (Revision 3). https://www.microsoft.com/en-us/research/publication/u-prove-cryptographic-specification-v1-1-revision-3/

[52] David Pointcheval and Olivier Sanders. 2016. Short Randomizable Signatures. In *CT-RSA 2016 (LNCS, Vol. 9610)*, Kazue Sako (Ed.). Springer, Heidelberg, 111–126. https://doi.org/10.1007/978-3-319-29485-8_7

[53] Michael Rosenberg, Jacob White, Christina Garman, and Ian Miers. 2022. `zk-creds`: Flexible Anonymous Credentials from zkSNARKs and Existing Identity Infrastructure. Cryptology ePrint Archive, Report 2022/878. https://eprint.iacr.org/2022/878.

[54] Olivier Sanders. 2020. Efficient Redactable Signature and Application to Anonymous Credentials. In *PKC 2020, Part II (LNCS, Vol. 12111)*, Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas (Eds.). Springer, Heidelberg, 628–656. https://doi.org/10.1007/978-3-030-45388-6_22

[55] Olivier Sanders. 2021. Improving Revocation for Group Signature with Redactable Signature. In *PKC 2021, Part I (LNCS, Vol. 12710)*, Juan Garay (Ed.). Springer, Heidelberg, 301–330. https://doi.org/10.1007/978-3-030-75245-3_12

[56] Tjerand Silde and Martin Strand. 2022. Anonymous tokens with public metadata and applications to private contact tracing. In *International Conference on Financial Cryptography and Data Security*. Springer, 179–199.

[57] Alberto Sonnino, Mustafa Al-Bassam, Shehar Bano, Sarah Meiklejohn, and George Danezis. 2019. Coconut: Threshold Issuance Selective Disclosure Credentials with Applications to Distributed Ledgers. In *NDSS 2019*. The Internet Society.

[58] Stefano Tessaro and Chenzhi Zhu. 2023. Revisiting BBS Signatures. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 691–721.