Threshold Anonymous Credentials with Silent Setup

Preshtha Garg
UC Berkeley

Sanjam Garg
UC Berkeley and Exponential Science Foundation

Guru-Vamsi Policharla
UC Berkeley

Bhaskar Roberts
UC Berkeley

Abstract

Anonymous credentials allow users to authenticate themselves in an anonymous and unlinkable fashion. By the end of 2026, EU member states will be required to issue digital identity wallets to their residents that enable authentication in this manner. In decentralized settings, we desire schemes with additional properties: schemes that allow multiple authorities to issue credentials, hide the identities of the issuers, and allow verifiers to dynamically choose their policies.

We present the first construction of issuer-hiding anonymous credentials with constant-sized showing, threshold issuance, and no requirement of interactive setup. Silent (non-interactive) setup is crucial as the various issuers may be slow-moving, independent organizations that are unwilling to coordinate in a distributed key generation protocol beforehand. Our construction also supports dynamic verifier policies. This is useful if different verifiers disagree about which issuers they trust or what threshold they accept.

At the heart of our scheme, we construct threshold structure-preserving signatures with silent setup and prove security in the generic group model. We also provide a NIZK for anonymous showing that is more efficient than a standard application of Groth-Sahai proofs. Finally, we provide an implementation of our scheme in Rust, along with concrete efficiency metrics.

1 Introduction

Anonymous credentials [16, 17] allow users to reveal certain attributes about themselves (attested to by an issuer) without leaking any additional information. A long line of research has focused on designing anonymous credentials [3, 4, 12, 14–17, 21, 25, 27, 33, 45, 46, 48, 49]. They have also been built by Microsoft [44], IBM [18], and deployed by Signal to realize group chats [20]. More recently, the eIDAS regulation from the European Commission entered into force in May 2024 which demands that all member states are required to offer an EU Digital Identity Wallet (EUDI). An EUDI allows EU citizens and residents to identify themselves to public and private online services in a *privacy preserving* manner — guaranteeing properties such as selective disclosure, unlinkability, unobservability and the right to pseudonymous authentication. Furthermore, even weaker forms of the primitive without attributes, typically referred to as anonymous tokens [24, 40, 51] have been deployed to protect against DDoS prevention in Cloudflare's Privacy Pass [24] (a variant is also used by Apple[1]), Google's Private State Tokens[2], and by Meta for private telemetry[3]. A recent draft at the IETF aims to standardize anonymous tokens[4].

Decentralized Issuance. In most of the literature and deployments of anonymous tokens/credentials, the issuer (in charge of providing user's with credentials) is assumed to be

[1] https://developer.apple.com/news/?id=huqjyh7k
[2] https://privacysandbox.google.com/protections/private-state-tokens
[3] https://engineering.fb.com/2022/03/30/security/de-identified-authentication-at-scale/
[4] https://datatracker.ietf.org/wg/privacypass/about/

a single entity. However, this may not be a reasonable assumption. A single issuer constitutes a single point of failure — which may be very difficult to detect given that showings of anonymous credentials come with strong privacy guarantees. Hence organizations may want to *distribute* trust amongst multiple operators of the credential system.

In fact, in certain situations the verifier may require that credentials be endorsed by *multiple* issuers in order to be valid. A great example is when members of a Decentralized Autonomous Organization (DAO) wish to issue credentials to external members — either for voting or perhaps to grant access to certain resources controlled by the DAO. The policy may demand that at least a certain (potentially weighted) fraction of the DAO needs to endorse the credential before it is considered valid.

There has been prior work that targets this issue by using threshold signatures to issue the credential [26, 48, 52, 53] but they crucially require issuers to coordinate in an initial interactive setup phase, which is not only expensive, but also has to be repeated every time an issuer joins/leaves the system. This is completely impractical in a fully decentralized system such as a DAO where there may be too many issuers (in the thousands) to coordinate between and the set of issuers may change frequently. Another strongly desirable property is the support for *dynamic* verification policies — in particular not all issuers may have the same requirements. For instance, different verifiers may want to specify a different threshold for the credential to be valid. Or some verifiers may be unwilling to accept credentials for a certain subset of issuers — possibly due to geo-political tensions. If using a threshold signature scheme, each verification policy would require a different setup phase.

Decentralized Issuance: Limitations of Prior Work. Indeed we are not the first ones to identify this problem — Coconut [52] explicitly states that the interactive setup is a limitation of their work. They also acknowledge that credentials issued by two different committees of issuers (after someone leaves/joins) are distinguishable, thereby reducing the privacy provided.

There have been attempts to address this issue, somewhat unsatisfactorily, by foregoing the threshold property. For instance [36] builds anonymous tokens with decentralized issuance (i.e. avoids the interactive setup) by leveraging the aggregation property of the BLS signature scheme coupled with blind signing. But this approach requires that *all* issuers in the set need to participate in the issuance of a credential which may not be feasible in practice — especially in a DAO. If one wanted to allow for certain issuers to go offline, then the exact set of issuers who participated would need to be revealed to the verifier — this will significantly reduce the anonymity set size, and may even act as a unique fingerprint in the extreme case. In a similar vein [34] use a similar strategy to build anonymous credentials from aggregate signatures with randomizable tags and signatures but they still suffer from similar issues.

In order to address the limitations in the above works, a new line of research has emerged that aims to provide an *Issuer-Hiding* property [7,13,21,41]. Here the idea is to allow the verifier to specify an acceptable set of N issuers, and the user can show that they have a credential from at least K of them. The state-of-the-art solution [41] requires $O(K)$ communication during the showing — effectively growing with the number of issuers. Although this is an improvement over needing a linear amount of communication, it is still impractical in the many-issuer setting. They also require either a trusted party holding a trapdoor to a set commitment scheme, or suffer from a worse verifier time growing as $O(k \cdot K)$, where k is the number of disclosed attributes.

The Dream Goal. It's clear that supporting threshold issuance without interactive setup is an important goal, but the prior results are unsatisfactory. An ideal solution in this setting would:

- Not require an interactive setup between issuers.
- Have a constant-sized showing and constant-time verification irrespective of the number of issuers.
- Not leak the exact set of parties that participated in issuance.

A general-purpose zkSNARK proving knowledge of sufficiently many signatures offers an ideal solution to this solution, but at the cost of a large prover time. In fact, this was the approach taken by [48] and they showed that a credential supporting additional properties such as revocation, auditing, and traceability has a showing size of approximately 0.5 KB, and costs less than 300k gas to verify on Ethereum. The two main downsides to this approach are:

- Proof generation takes a long time, on the order of tens of seconds on weak devices.
- They verify EdDSA signatures inside the SNARK circuit, which requires that the random oracle is explicitly instantiated and proved inside the circuit, resulting in heuristic security. Recent work [38] has shown that proving the Fiat-Shamir hash function inside the circuit can lead to attacks where a malicious prover can generate proofs for false statements. Although there isn't an explicit attack against [48], it is nonetheless desirable to have a scheme based on well-founded assumptions.

1.1 Our Contributions

A Blackbox zkSNARK. When using a zkSNARK, the main source of inefficiency stems from the non-blackbox use of cryptographic operations like hashing and elliptic curve operations. If we are able to avoid this overhead, with a fully *blackbox* zkSNARK, then one can hope to build a concretely efficient prover alongside all of the desirable properties mentioned above.

In this work, we deviate from prior approaches and tackle the problem using a fundamentally different strategy. We build a *special-purpose* blackbox zkSNARK that does not suffer from the same drawbacks as [48]. Concretely, we are able to *efficiently* prove a relation of the form:

> *I know signatures* $\{\ \sigma_i\ \}_{i\in[t]}$ *from at least* t *different issuers* $\{\ I_i\ \in \mathcal{I}\}_{i\in[t]}$ *on a commitment* com *to attributes* attrs *such that the* j*-th attribute is* attrs_j*,*

where the highlighted elements denote the witness. Importantly, we achieve a silent setup phase, where the issuers do not need to coordinate and in fact do not even need to be aware of each other. Every issuer will issue credentials (signatures on a commitment) independently. This immediately yields an anonymous credential with silent setup.

We now outline the path to building our construction and the main challenges we face.

Threshold SPS with Silent Setup. We build the first threshold structure preserving signature scheme with silent setup. An important requirement when designing this scheme is to ensure that it has a *fully algebraic* verification equation — that is the verifier only needs to check pairing product equations. This will ensure that we can subsequently design an efficient proof system proving knowledge of the signature.

Our approach is inspired by [23, 29] which builds a BLS [11] based threshold signature scheme with silent setup. The main idea is to use the univariate sumcheck [6,47] to (1) prove that t issuer keys have been correctly aggregated into an aggregate verification key aVK, and (2) prove knowledge of a signature under the aggregate verification key.

Their construction can be used directly to build anonymous tokens with silent setup by obtaining blind signatures from issuers as done in [36]. However, there are two main barriers to overcome in order to build full-fledged anonymous credentials with silent setup:

- First, in BLS signatures the message is hashed using a random oracle, which means that if we wish to build a selective-disclosure anonymous credential, then we need to prove the execution of the random oracle inside a zero-knowledge proof.
- Second, they use the Fiat-Shamir transform to transform an interactive verification procedure into a non-interactive publicly verifiable signature scheme.

A natural step to overcome the first issue is to use a structure preserving signature scheme [1] as the verification is fully algebraic. We note that this is not a trivial step as the signature scheme needs to be carefully selected to support aggregation, allow for the message to be a commitment to the attributes, and also support an efficient proof of knowledge of the signature. But the latter issue still persists. Our approach uses techniques from [30] to avoid the Fiat-Shamir transform altogether. This is done by changing the strategy used to prove that public keys have been aggregated correctly (Section 3).

Optimized Proof of Knowledge of Threshold SPS. The next step is to prove knowledge of a threshold structure-preserving signature on some commitment to the attributes and an opening of that commitment at the relevant positions. Since the verification only involves pairing product equations, one can use the Groth-Sahai [32] proof system. This typically comes with a large concrete overhead, which we are able to reduce by leveraging the *linear* verification property of our threshold structure-preserving signature scheme. Our techniques extend to any set of pairing product equations with linear verification and are of independent interest (Section 4.4).

Implementation and Evaluation. We implement our construction in rust and evaluate the performance of our threshold structure preserving signature scheme with silent setup and our anonymous credential scheme with silent setup. Our benchmarks show that our scheme is practical and can easily scale to thousands of issuers, outperforming prior work where the showing and verification complexity grows with the number of issuers. (Section 5)

2 Preliminaries

2.1 Bilinear maps

We briefly recall the definition of an asymmetric bilinear map [10, 35]. Let $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ be distinct groups, all of prime order p, and let $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a mapping from $\mathbb{G}_1 \times \mathbb{G}_2$ onto the target group $\mathbb{G}_T$. Let g_1, g_2 be generators for $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. We say that $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e, g, \hat{g})$ are the parameters of an asymmetric bilinear group if the following conditions are met:

- (Efficiency) The group operations in $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ as well as the mapping $e(\cdot, \cdot)$ are all efficiently computable.
- (Non-degeneracy) $e(g_1, g_2) = g_T$, where g_T is a generator of $\mathbb{G}_T$.
- (Bilinearity) $e(g_1^a, g_2^b) = g_T^{ab}$ for all $a, b \in \mathbb{Z}_q$.

We will also use $h \circ \hat{h}$ to represent the pairing operation. Finally, we will use the additive notation for group operations on $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$. For matrices of group elements, we use the notation from [42]. For example, given two matrices of field elements, $\mathbf{A} = (a_{i,j}) \in \mathbb{Z}_p^{n \times n}$ and $\mathbf{B} = (b_{i,j}) \in \mathbb{Z}_p^{n \times n}$, then $[\mathbf{A}]_1 = ([a_{i,j}]_1)$, $[\mathbf{B}]_2 = ([b_{i,j}]_2)$, and $[\mathbf{A}]_1 \circ [\mathbf{B}]_2 = [\mathbf{A} \cdot \mathbf{B}]_T$. Additionally, given a scalar $c \in \mathbb{Z}_p$, $[c]_1 \circ [\mathbf{B}]_2 = [c \cdot \mathbf{B}]_T$.

2.2 Generic bilinear group model.

We will use an extension of the generic group model [43, 50] adapted to bilinear groups. We follow the presentation in [8], in particular defining the label sets as known sets of size equal to the size of the group. Moreover, we define the Add interface to directly support arbitrary linear combinations of group elements, which is without loss of generality due to the double-and-add algorithm.

Definition 2.1 (Generic Bilinear Group Oracle)**.** A *generic bilinear group oracle* is a stateful oracle GGM, parameterized by a prime $p \in \mathbb{Z}$, and three sets S_1, S_2, S_T of size p.

- On a query $\mathsf{GGM.Init}()$, the oracle samples and stores three uniformly random injections $\tau_i : \mathbb{Z}_p \to S_i$ for $i \in \{1, 2, T\}$. Future queries to $\mathsf{GGM.Init}$ are ignored.
- On a query $\mathsf{GGM.GetGenerator}(i)$, the oracle returns $\tau_i(1)$.
- On a query $\mathsf{GGM.Add}(i, g_1, g_2, c_1, c_2)$, return $\perp$ unless $g_1, g_2 \in S_i$ and $c_1, c_2 \in \mathbb{Z}_p$. Otherwise, return $g = \tau_i(c_1\tau_i^{-1}(g_1) + c_2\tau_i^{-1}(g_2))$.
- On a query $\mathsf{GGM.Pair}(g_1, g_2)$, return $\perp$ unless $g_1 \in S_1, g_2 \in S_2$. Otherwise, return $\tau_T(\tau_1^{-1}(g_1) \cdot \tau_2^{-1}(g_2))$.

Modeling a group-output random oracle. We will also model the scenario in which parties have access to a cryptographic hash function that outputs group elements, which is needed for example to support the signature scheme in [42]. Since the sets S_1, S_2, S_T are public, this is straightforward to implement via a random oracle defined from $\mathbb{Z}_p \to S_i$.

Let $\mathbf{X}$ be a list of variables. Every group element given to the adversary is represented as a variable in $\mathbf{X}$ or a polynomial in $\mathbf{X}$. The adversary can only output linear combinations of the polynomials given to it as input.

We say that a polynomial $P(\mathbf{X})$ *varies with* a given variable $V \in \mathbf{X}$ if the degree in V of P is ≥ 1. Given a polynomial $P(\mathbf{X})$ and a list of polynomials $\mathbf{L}(\mathbf{X}) = (Q_1(\mathbf{X}), \ldots, Q_{|\mathbf{L}|}(\mathbf{X}))$, we say that $P(\mathbf{X})$ is *dependent* on $\mathbf{L}(\mathbf{X})$ if $P(\mathbf{X})$ can be written as a linear combination of the polynomials in $\mathbf{L}(\mathbf{X})$. i.e.

$$P(\mathbf{X}) = \sum_{i \in [|\mathbf{L}|]} \alpha_i \cdot Q_i(\mathbf{X})$$

for some $\alpha_1, \ldots, \alpha_{|\mathbf{L}|} \in \mathbb{F}_p$. Every polynomial output by the adversary is dependent on the polynomials given to the adversary as input.

Let $\mathbf{L}_1(\mathbf{X})$ and $\mathbf{L}_2(\mathbf{X})$ list the polynomials in $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively, that are given to the adversary as input. Because this is a type-III pairing, any output of the adversary in $\mathbb{G}_1$ must be dependent on $\mathbf{L}_1(\mathbf{X})$, and any output in $\mathbb{G}_2$ must be dependent on $\mathbf{L}_2(\mathbf{X})$.

Furthermore, any pairing-product equation check is implemented as a polynomial identity check in the GGM. For instance, if a, b, c, d are group elements represented by the polynomials $A(\mathbf{X})$, $B(\mathbf{X})$, $C(\mathbf{X})$, $D(\mathbf{X})$, then the check $a \circ b = c \circ d$ is implemented as the polynomial identity check $A(\mathbf{X}) \cdot B(\mathbf{X}) = C(\mathbf{X}) \cdot D(\mathbf{X})$.

Polynomials: Let $\omega \in \mathbb{Z}_p$ be an N-th primitive root of unity. Let $\mathbb{H} = \{\omega^1, \ldots, \omega^N\}$ be the N-th roots of unity. Let $Z(T) = T^N - 1$ be the vanishing polynomial on $\mathbb{H}$.

For each $i \in [N]$, let $L_i(T)$ be the Lagrange polynomial of degree $\leq N-1$ such that $L_i(\omega^i) = 1$, and $L_i(\omega^j) = 0$ for all $j \in [N] \backslash \{i\}$. Furthermore, lagrange polynomials over the roots of unity additionally have the following properties:

- $L_i(X) = \frac{\omega^i}{N} \cdot \frac{X^N - 1}{X - \omega^i}$
- $L_i(0) = \frac{1}{N}$

Univariate Sumcheck. Our construction relies on a univariate sumcheck protocol [6, 47], slightly modified to work for inner products [19]. In particular, we use the following lemma.

Lemma 2.1 (Univariate Sumcheck [6, 47])**.** Let $A(x) = \sum_{i=1}^{|\mathbb{H}|} a_i \cdot L_i(x)$, $B(x) = \sum_{i=1}^{|\mathbb{H}|} b_i \cdot L_i(x)$. It holds that

$$A(x) \cdot B(x) = \frac{\sum_i a_i b_i}{|\mathbb{H}|} + Q_x(x) \cdot x + Q_Z(x) \cdot Z(x),$$

where both Q_x and Q_Z are polynomials with degree $\leq |\mathbb{H}| - 2$ defined as

$$Q_x(x) = \sum_i a_i b_i \frac{L_i(x) - L_i(0)}{x},$$
$$Q_Z(x) = \sum_i a_i b_i \frac{L_i^2(x) - L_i(x)}{Z(x)} + \sum_{i \neq j} a_i b_j \frac{L_i(x) L_j(x)}{Z(x)}$$

We note that the original sumcheck is concretely stated for $b_i = 1$. In our case, we treat general inner products, which is a straightforward generalization (see [19]).

For space reasons, we defer the preliminary on KZG commitment schemes to Appendix A

3 Silent Threshold Structure-Preserving Signatures

We construct a threshold signature scheme that forms the core of our anonymous credentials construction (Section 4). The signature scheme supports signature aggregation with a dynamic threshold, allows independent signers to generate their keys without interactive setup, produces constant-sized signatures with constant-time verification, and is structure-preserving.

3.1 Overview

Picking a base SPS. Our construction chooses a structure-preserving signature scheme (SPS) with certain properties and aggregates the signatures and verification keys using the hinTS technique of [29, 30]. We chose the SPS in [42] as our base in order to *enable aggregation* and to *enable signing of KZG commitments*.

As discussed in [42], the nonlinear operations in most SPS schemes make them unsuited to aggregation, since the resulting signature cannot be reconstructed using Lagrange interpolation. In order to use hinTS-style aggregation, the verifier's pairing product equations should not pair together two aggregated group elements. Recent works on threshold SPSs identified two SPSs that have this property: the IMM-SPS scheme presented in [22] and the TSPS presented in [42].

Next, we would like our signature scheme to allow signing of a KZG commitment, and only the SPS of [42] makes this possible. Our anonymous credentials scheme should support selective disclosure of attributes with a constant-sized credential. The standard way to achieve this is to have the issuers sign a length-reducing commitment to the list of attributes. However, [22] uses a special indexed message space comprising pairs in $\mathbb{G}_1 \times \mathbb{G}_2$ of correlated group elements. This message space is not compatible with most standard commitment schemes, since they usually produce an element in either $\mathbb{G}_1$ or $\mathbb{G}_2$. To convert a KZG commitment to the indexed Diffie-Hellman message space, one needs access to the commitment trapdoor τ. This is the approach used in [41], where the anonymous credential scheme can either sign just one attribute per credential, or it needs a trusted authority to hold the trapdoor to the commitment scheme and create commitments to the attributes.

To avoid these tradeoffs, we choose to use the SPS from [42] as our base. This scheme has a message space that is simply $\mathbb{G}_1$.

Signature Aggregation. The technique of signature aggregation with hints [29, 30] is a powerful tool that allows us to lift signature schemes of a certain format[5] to threshold signature schemes with silent setup. We will start with a structure-preserving signature scheme based on [39, 42], which verifies the signature with the following equations:

$$\begin{aligned} \sigma_1 \circ [\mathbf{A}]_2 &= \begin{pmatrix} [1]_1 & m \end{pmatrix} \circ \mathsf{vk} + \sigma_2 \circ [\mathbf{UA}]_2 + \sigma_3 \circ [\mathbf{VA}]_2 \\ \sigma_2 \circ \sigma_4 &= \sigma_3 \circ [1]_2 \end{aligned} \tag{1}$$

[5] We need the property that the verifier's pairing product equations do not pair together two aggregated group elements. The aggregated group elements take the place of the highlighted elements in Eq. (1).

The highlighted components are different for each signature or signer and will need to be aggregated. The remaining components are public constants or (in the case of m and σ_4) are uniquely determined by the message.

In order to support multiple signers, we will have each signer $i \in [N]$ sample their own keys $(\mathsf{sk}_i, \mathsf{vk}_i)$ and sign the message individually, producing a partial signature σ^i. Then the user that receives the partial signatures will aggregate them into a constant-sized signature and prove that they have done the aggregation correctly.

To aggregate the partial signatures $\left(\sigma^i\right)_{i\in S}$, the aggregator chooses non-zero coefficients $(b_i)_{i\in S}$ and computes:

$$\hat{\sigma}_1 = \frac{1}{N} \cdot \sum_{i\in S} b_i \cdot \sigma_1^i$$
$$\hat{\sigma}_2 = \frac{1}{N} \cdot \sum_{i\in S} b_i \cdot \sigma_2^i$$
$$\hat{\sigma}_3 = \frac{1}{N} \cdot \sum_{i\in S} b_i \cdot \sigma_3^i$$
$$\text{aVK} = \frac{1}{N} \cdot \sum_{i\in S} b_i \cdot \mathsf{vk}_i$$

Assuming that aVK was aggregated honestly, the verifier can verify all $|S|$ partial signatures at once by checking Eq. (1) with $(\hat{\sigma}_1, \hat{\sigma}_2, \hat{\sigma}_3, \text{aVK})$ taking the place of $(\sigma_1, \sigma_2, \sigma_3, \mathsf{vk})$.

This construction has several nice properties. First, the scheme has silent setup. Since the signers sample their keys independently, they do not need to engage in a distributed key generation procedure. Second, the verification runs in constant time.

Third, this scheme supports threshold signing, where the threshold t is chosen dynamically at aggregation time. The aggregator just needs to prove that at least t signatures were aggregated. To do so, they choose $(b_i)_{i\in S}$ to lie on a polynomial $B(T)$ of degree $\leq N - t$ such that $B(0) = 1$. The aggregator also provides a KZG-style proof that B has these properties. Any non-zero polynomial of degree $\leq N - t$ must be non-zero on at least t out of N points, so this proves that at least t values in $(b_i)_{i\in N}$ are non-zero.

Sumcheck proof of correct aggregation: The aggregator will prove that aVK was computed honestly using KZG-style polynomial commitments and openings [37]. We will assume the reader is familiar with KZG techniques.

aVK can be expressed as the inner product of two vectors: $(b_i)_{i\in N}$ and $(\mathsf{vk}_i)_{i\in N}$, multiplied by $\frac{1}{N}$. We will express these vectors as polynomials $B(T)$ and $\mathsf{VK}(T)$ and multiply them together as a rough proxy for the inner product.

Let ω be an N-th primitive root of unity, and for each $i \in [N]$, let $L_i(T)$ be the Langrange polynomial such that $L_i(\omega^i) = 1$ and $L_i(\omega^j) = 0$ for all $j \neq i$. Next, let $B(T) = \sum_{i\in[N]} L_i(T) \cdot b_i$ and $\mathsf{VK}(T) = \sum_{i\in[N]} L_i(T) \cdot \mathsf{vk}_i$.

Next, the univariate sumcheck lemma (Lemma 2.1) says that:

$$B(T) \cdot \mathsf{VK}(T) = \underbrace{\frac{1}{N} \cdot \sum_{i\in[N]} b_i \cdot \mathsf{vk}_i}_{\text{honest aVK}} + Q_x(T) \cdot T + Q_Z(T) \cdot \left(T^N - 1\right) \tag{2}$$

for some polynomials Q_x, Q_Z of degree $\leq T-2$. Intuitively, the honestly computed aVK is the inner product of the B and VK vectors (multiplied by $\frac{1}{N}$), and $Q_x(T) \cdot T + Q_Z(T) \cdot \left(T^N - 1\right)$ are cross terms.

To prove that $\text{aVK} = \frac{1}{N} \cdot \sum_{i \in [N]} b_i \cdot \mathsf{vk}_i$, the aggregator proves that Eq. (2) still holds when $\frac{1}{N} \cdot \sum_{i \in [N]} b_i \cdot \mathsf{vk}_i$ (the honest aVK) is replaced with the actual aVK.

The prover commits to polynomials B, VK, Q_x, Q_Z (and a few others) by evaluating these polynomials at a random input τ. Then the verifier performs the following checks:

1. $B \circ \mathsf{VK} = [1]_1 \circ \text{aVK} + [\tau]_1 \circ Q_x + \left[\tau^N - 1\right]_1 \circ Q_Z$
2. $B \circ [\tau^t]_2 = [1]_1 \circ \hat{B}$
3. $B \circ [1]_2 = [\tau]_1 \circ Q_0 + [1]_T$
4. $[1]_1 \circ \hat{\text{aVK}} = [\tau^N]_1 \circ \text{aVK}$
5. $[\tau^2]_1 \circ Q_x = [1]_1 \circ \hat{Q}_x$

Item 1 checks the sumcheck relation Eq. (2), with aVK taking the place of $\frac{1}{N} \cdot \sum_{i \in [N]} b_i \cdot \mathsf{vk}_i$. Items 2 and 3 prove that B is well-formed. They prove that $\deg_T(B) \leq N - t$ and $B(0) = 1$. Then for at least t values of $i \in [N]$, $B(\omega^i) \neq 0$. Item 4 proves that $\deg_T(\text{aVK}) = 0$. This is a new check that does not appear in [30]. It ensures that aVK does not depend on τ. This is necessary for the security of our scheme and makes our proof of security simpler than that of [30]. Finally, Item 5 proves that $\deg_T(Q_x) \leq N - 2$. This is needed to argue that the terms of $Q_x(T) \cdot T$ and $Q_Z(T) \cdot \left(T^N - 1\right)$ do not cancel out in Eq. (2).

Hints: In order to compute the commitment to $\mathsf{VK}(T)$, the prover needs to know the values $(L_i(\tau) \cdot \mathsf{vk}_i)_{i \in [N]}$ as elements of $\mathbb{G}_2$. They cannot compute them for themselves because τ is only given in the form of group elements such as $[\tau]_1$, and $[\tau]_2$, and vk_i is an element of $\mathbb{G}_2$. To aid the prover, each signer i will compute $L_i(\tau) \cdot \mathsf{vk}_i$ as $\mathsf{sk}_i \cdot [L_i(\tau)\mathbf{A}]_2$ and publish it as a hint. This works because $\mathsf{vk}_i = \mathsf{sk}_i \cdot [\mathbf{A}]_2$.

Additional hints are required to compute commitments to $Q_x(T)$, $\hat{Q}_x(T)$, and $Q_Z(T)$. In total, each signer must output $N + 4$ group elements as hints, and N of those hints are used to commit to Q_Z. Intuitively, we need each of N signers to generate $\sim N$ hints because Eq. (2) produces $\sim N^2$ cross terms.

Improvements over prior work: Our aggregation procedure makes several improvements to the aggregation techniques of prior work [29, 30].

First, [30] required a dummy party to be included in every aggregation to ensure that $B(T) \neq 0$. We do not need a dummy party because we instead check that $B \circ [1]_2 = [\tau]_1 \circ Q_0 + [1]_T$, which ensures that $B(0) = 1$. This reduces the size of the crs and simplifies the notation of our aggregation procedure.

Second, we give a simple and modular proof that aVK was computed honestly. We do so by introducing a new check – $[1]_1 \circ \hat{\text{aVK}} = [\tau^N]_1 \circ \text{aVK}$ – which ensures that aVK does not vary with τ.

This check is necessary for our scheme. Without this check, there is an attack that would break the unforgeability of our signature scheme by mauling an aggregate signature on one message to an aggregate signature on a new message by canceling out new terms in the verification equations with some malicious terms in aVK.

[29, 30] did not include this check, so they had to use properties of the underlying signature scheme's verification procedure to prove that aVK does not depend on τ. In contrast, we can prove that aVK was computed honestly without using such properties. Our proof is more or less the same for any signature scheme. This makes our security proof more modular. After proving that aVK was computed honestly, we can directly reduce to the security of the underlying (aggregate) signature scheme. For these reasons, we believe that the degree check on aVK is an essential addition to any future work using hints-based aggregation.

Third, we present the following optimization that achieves constant-sized hints for the case of multisignatures. Then our multisignature scheme has the same asymptotic efficiency as classic multisignature constructions, such as [9].

Optimization for Multisignatures: When the signing threshold is fixed to $t = N$, we have a multisignature scheme. In this case, we can use hints that each comprise 4 group elements, rather than $N + 4$ group elements.

For any aggregate signature with $t = N$, we must set $B(T) = 1$ and $Q_Z(T) = 0$. Then the only hints that party i needs to provide to enable aggregation are:

$$\mathsf{hint}_i = \left(L_i(\tau) \cdot \mathsf{vk}_i, (L_i(\tau) - L_i(0))\, \tau \cdot \mathsf{vk}_i, \tau^N \cdot \mathsf{vk}_i, \frac{L_i(\tau) - L_i(0)}{\tau} \cdot \mathsf{vk}_i\right)$$

3.2 Definitions

Let us define the syntax of our signature scheme, as well as the correctness and security properties. The following definition of syntax is based on [29] definition 1.

Definition 3.1 (STS Syntax)**.** A *silent threshold signature* scheme (STS) comprises the following algorithms:

- $\mathsf{Setup}(1^\lambda, 1^N) \to (\mathsf{pp}, \mathsf{crs})$: Takes as inputs the security parameter λ and an upper bound N on the number of signers in the universe, both expressed in unary. Then Setup generates the public parameters pp and the common reference string crs. All other algorithms implicitly take pp as input.
- $\mathsf{KGen}(\mathsf{crs}) \to (\mathsf{sk}, \mathsf{vk})$: For a given signer $i \in [N]$, KGen generates that signer's secret key sk and public verification key vk.
- $\mathsf{HintGen}(\mathsf{crs}, \mathsf{sk}, i, N) \to \mathsf{hint}$: For a given signer $i \in [N]$, HintGen generates that signer's aggregation hint hint.
- $\mathsf{Preprocess}(\mathsf{crs}, (\mathsf{hint}_i, \mathsf{vk}_i)_{i\in[N]}) \to (\mathsf{VK}, \mathsf{ak})$: Takes as input all the signers' hints and verification keys $(\mathsf{hint}_i, \mathsf{vk}_i)_{i\in[N]}$ and compiles them to generate an aggregate verification key VK and an aggregation key ak.
- $\mathsf{PartialSign}(\mathsf{crs}, \mathsf{sk}, m) \to \sigma$: For a given signer $i \in [N]$ with signing key sk, PartialSign takes as input a message m and outputs a partial signature σ on that message.
- $\mathsf{PartialVerify}(\mathsf{crs}, \mathsf{vk}, m, \sigma) \to \{0, 1\}$: For a given signer $i \in [N]$ with verification key vk, PartialVerify takes as input a message m and partial signature σ generated by party i and outputs 1 if the signature is accepted and 0 otherwise.
- $\mathsf{AggrSign}(\mathsf{crs}, \mathsf{ak}, t, m, S, (\sigma^{i_1}, \ldots, \sigma^{i_n})) \to \sigma$: Takes as input a threshold $t \in [N]$, a subset of signers $S = \{i_1, \ldots, i_n\} \subseteq [N]$ of size $n \geq t$, and the partial signatures generated by those signers $(\sigma^{i_1}, \ldots, \sigma^{i_n})$. Then AggrSign outputs a succinct aggregate signature σ.
- $\mathsf{AggrVerify}(\mathsf{crs}, \mathsf{VK}, t, m, \sigma) \to \{0, 1\}$: Takes as input the aggregate verification key VK, a threshold t, a message m, and an aggregate signature σ. Then AggrVerify outputs 1 if the signature is accepted and 0 otherwise.

Efficiency: An STS satisfies the following efficiency properties:

- VK (output by Preprocess) has size constant in (N, t).
- σ (output by AggrSign) has size constant in (N, t).

- AggrVerify runs in time constant in (N, t).[6]

Structure-Preserving: The STS is structure-preserving if crs, VK, m, and σ are tuples of group elements in $\mathbb{G}_1 \cup \mathbb{G}_2$, and AggrVerify consists only of checking pairing product equations. The abbreviation STSPS refers to a silent threshold structure-preserving signature scheme.

Correctness says that an aggregate signature will pass verification if the partial signatures pass verification and aggregation was performed correctly. In more detail, the adversary can corrupt a subset of parties and even generate partial signatures on behalf of those parties. The challenger will honestly aggregate the partial signatures from the corrupted parties (as long as they pass PartialVerify) as well as the ones from the honest parties. In the end, the aggregated signature should be accepted by AggrVerify with overwhelming probability for any threshold less than or equal to the number of partial signatures that were aggregated.

Our correctness definition is similar to [29], definition 4. In Definition B.1 and Fig. 1, we define correctness formally, and in Theorem B.1, we prove that our STSPS scheme constructed in Definition 3.2 satisfies correctness.

Security We allow the adversary to corrupt a subset of the parties Cor $\subset [N]$. Then the adversary can query PartialSign to any honest parties on any messages of their choice. In the end, the adversary outputs a forgery (t, m, σ^*). Let Q be the subset of the honest parties to which the adversary made a PartialSign query on message m. The adversary wins the security game if $\mathsf{AggrVerify}(\mathsf{crs}, \mathsf{VK}, t, m, \sigma^*) = 1$, and $t > |Q \cup \mathsf{Cor}|$.

The security definition we use here is similar to [29], definition 5. In Definition B.2 and Fig. 2, we define security formally, and in Theorem B.2, we prove that our STSPS scheme constructed in Definition 3.2 is secure.

3.3 Construction

Here, we will construct a threshold structure-preserving signature scheme with silent setup.[7] Our construction is based on the threshold structure-preserving signature scheme of [42] and uses hinTS-style aggregation [29, 30].

Notation and Variables: Let $\lambda \in \mathbb{N}$ be the security parameter. Let $N \in \mathbb{N}$ be the maximum number of signers in the universe. Let the threshold $t \in \{0\} \cup [N]$ be the minimum number of signers needed to produce a valid signature. It can be chosen at aggregation-time and can be different for each aggregate signature. Let the message space be $\mathcal{M} = \mathbb{G}_1$.

We let $\mathsf{crs}_{\mathsf{TSPS}} = \left([\mathbf{UA}]_2, [\mathbf{VA}]_2, [\mathbf{B}]_1, [\mathbf{B}^\top \mathbf{U}]_1, [\mathbf{B}^\top \mathbf{V}]_1\right)$ be the common reference string used in the Threshold Structure-Preserving Signature scheme of [42].

Definition 3.2 (Construction of the STSPS Scheme)**.**

- $\mathsf{Setup}(1^\lambda, 1^N)$:
  1. Sample the parameters of a type-III bilinear group $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e, g, \hat{g})$, and sample collision-resistant hash function $\mathcal{H} : \{0,1\}^* \to \mathbb{Z}_p$. Compute $\mathsf{pp} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e, g, \hat{g}, \mathcal{H})$.
  2. Compute $\mathsf{crs}_{\mathsf{TSPS}}$.
     - Sample $\mathbf{A}, \mathbf{B} \xleftarrow{\$} \left(\begin{smallmatrix} 1 \\ \mathbb{Z}_p \end{smallmatrix}\right)$ and $\mathbf{U}, \mathbf{V} \xleftarrow{\$} \mathbb{Z}_p^{2\times 2}$.
     - Set $\mathsf{crs}_{\mathsf{TSPS}} = \left([\mathbf{UA}]_2, [\mathbf{VA}]_2, [\mathbf{B}]_1, [\mathbf{B}^\top \mathbf{U}]_1, [\mathbf{B}^\top \mathbf{V}]_1\right)$.

[6] We model AggrVerify as having random access to crs without having to read in all of crs as input. Additionally, cryptographic group operations and pairings are counted as constant-time operations.

[7] We note that our Rust implementation of this construction swaps the roles of $\mathbb{G}_1$ and $\mathbb{G}_2$ because group operations in $\mathbb{G}_1$ are less expensive.

3. Compute $\mathsf{crs}_{\mathsf{hints}}$.[8]
   - Sample $\tau \xleftarrow{\$} \mathbb{Z}_p$.
   - Set

$$\mathsf{crs}_{\mathsf{hints}} = \left(\left([1]_1, [\tau]_1, \ldots, \left[\tau^N\right]_1\right), \left([\mathbf{A}]_2, [\mathbf{A}\cdot\tau]_2, \ldots, \left[\mathbf{A}\cdot\tau^N\right]_2\right) \right)$$

4. Set $\mathbf{crs} = (\mathsf{crs}_{\mathsf{TSPS}}, \mathsf{crs}_{\mathsf{hints}})$ and output $(\mathsf{pp}, \mathbf{crs})$.

- $\mathsf{KGen}(\mathbf{crs})$: Generate the keys for the underlying TSPS.
  1. Sample $\mathbf{K} \xleftarrow{\$} \mathbb{Z}_p^{2\times 2}$
  2. Compute $\mathbf{sk} = \mathbf{K}$, $\mathbf{vk} = [\mathbf{KA}]_2$. Return $(\mathbf{sk}, \mathbf{vk})$
- $\mathsf{HintGen}(\mathbf{crs}, \mathbf{sk}, i, N)$:
  1. Parse $\mathbf{sk}$ as $\mathbf{K}$
  2. Compute and output

$$\mathbf{hint} = \Bigg([\mathbf{KA}\cdot L_i(\tau)]_2, [\mathbf{KA}\cdot\tau\left(L_i(\tau)-L_i(0)\right)]_2, \left[\mathbf{KA}\cdot\tau^N\right]_2, \left[\mathbf{KA}\cdot\frac{L_i^2(\tau)-L_i(\tau)}{Z(\tau)}\right]_2, \left[\mathbf{KA}\cdot\frac{L_i(\tau)-L_i(0)}{\tau}\right]_2 \left\{\left[\mathbf{KA}\cdot\frac{L_i(\tau)\cdot L_j(\tau)}{Z(\tau)}\right]_2\right\}_{j\in[N]\setminus\{i\}} \Bigg)$$

- $\mathsf{IsValid}(\mathbf{crs}, \mathbf{hint}, \mathbf{vk}, i, N)$:
  1. Parse $\mathbf{hint}$ as

$$\mathbf{hint} = \left(\mathbf{h}_1, \mathbf{h}_2, \mathbf{h}_3, \mathbf{h}_4, \mathbf{h}_5, \{\mathbf{h}_{6,j}\}_{j\in[N]\setminus\{i\}}\right)$$

  2. Output 1 if the following checks pass, and output 0 otherwise:
     (a) $[1]_1 \circ \mathbf{h}_1 = [L_i(\tau)]_1 \circ \mathbf{vk}$
     (b) $[1]_1 \circ \mathbf{h}_2 = [\tau\left(L_i(\tau)-L_i(0)\right)]_1 \circ \mathbf{vk}$
     (c) $[1]_1 \circ \mathbf{h}_3 = \left[\tau^N\right]_1 \circ \mathbf{vk}$
     (d) $[1]_1 \circ \mathbf{h}_4 = \left[\frac{L_i^2(\tau)-L_i(\tau)}{Z(\tau)}\right]_1 \circ \mathbf{vk}$
     (e) $[1]_1 \circ \mathbf{h}_5 = \left[\frac{L_i(\tau)-L_i(0)}{\tau}\right]_1 \circ \mathbf{vk}$
     (f) $[1]_1 \circ \mathbf{h}_{6,j} = \left[\frac{L_i(\tau)\cdot L_j(\tau)}{Z(\tau)}\right]_1 \circ \mathbf{vk}$ for all $j \in [N]\setminus\{i\}$
- $\mathsf{Preprocess}(\mathbf{crs}, (\mathbf{hint}_i, \mathbf{vk}_i)_{i\in[N]})$:
  1. For each $i \in [N]$, compute $\mathsf{IsValid}(\mathbf{crs}, \mathbf{hint}_i, \mathbf{vk}_i, i, N)$. Let $V \subseteq [N]$ be the set of signers for which $\mathsf{IsValid}$ returns 1.
  2. For each $i \in V$, compute:

$$\mathbf{ak}_i = \Bigg(\mathbf{vk}_i, \mathbf{vk}_i\cdot\tau\left(L_i(\tau)-L_i(0)\right), \mathbf{vk}_i\cdot\tau^N, \mathbf{vk}_i\cdot\frac{L_i^2(\tau)-L_i(\tau)}{Z(\tau)}, \mathbf{vk}_i\cdot\frac{L_i(\tau)-L_i(0)}{\tau}, \sum_{j\in V\setminus\{i\}} \mathbf{vk}_j\cdot\frac{L_i(\tau)\cdot L_j(\tau)}{Z(\tau)} \Bigg)$$

[8] Since the first entry of $\mathbf{A}$ is 1, then $\mathbf{crs}$ contains $\left([1]_2, [\tau]_2, \ldots, [\tau^N]_2\right)$.

Then compute the aggregation key:

$$\mathbf{ak} = (V, (\mathbf{ak}_i)_{i \in V})$$

3. Compute the verification key $\mathbf{VK}$ as follows:

$$\mathbf{VK} = \sum_{i \in V} \mathbf{vk}_i \cdot L_i(\tau)$$

4. Output $(\mathbf{VK}, \mathbf{ak})$

- $\mathsf{PartialSign}(\mathbf{crs}, \mathbf{sk}, m)$: Run the PartialSign algorithm from the underlying TSPS.

1. Parse $\mathbf{sk} = \mathbf{K}$. Check that $m \in \mathbb{G}_1$.
2. Compute $h = \mathcal{H}(m)$
3. Sample $r \xleftarrow{\$} \mathbb{Z}_p$
4. Compute:

$$\begin{aligned}
\sigma_1 &= \begin{pmatrix}[1]_1 & m\end{pmatrix} \cdot \mathbf{K} + [r\mathbf{B}^\top(\mathbf{U} + h\mathbf{V})]_1 \\
\sigma_2 &= [r\mathbf{B}^\top]_1 \\
\sigma_3 &= [hr\mathbf{B}^\top]_1 \\
\sigma_4 &= [h]_2
\end{aligned}$$

5. Output $\sigma \coloneqq (\sigma_1, \sigma_2, \sigma_3, \sigma_4)$

- $\mathsf{PartialVerify}(\mathbf{crs}, \mathbf{vk}, m, \sigma)$: Run the PartialVerify algorithm from the underlying TSPS.

1. Parse σ as $(\sigma_1, \sigma_2, \sigma_3, \sigma_4)$.
2. Perform the following checks:
   (a) $\sigma_1 \circ [\mathbf{A}]_2 = \begin{pmatrix}[1]_1 & m\end{pmatrix} \circ \mathbf{vk} + \sigma_2 \circ [\mathbf{UA}]_2 + \sigma_3 \circ [\mathbf{VA}]_2$
   (b) $\sigma_2 \circ \sigma_4 = \sigma_3 \circ [1]_2$

   If either of the checks fail, then output 0 (reject). Otherwise output 1 (accept).

- $\mathsf{AggrSign}\left(\mathbf{crs}, \mathbf{ak}, t, m, S, \left(\sigma^{i_1}, ..., \sigma^{i_n}\right)\right)$:

1. Let $S = \{i_1, \ldots, i_n\} \subseteq [N]$ be the indices of the n signers whose signatures we will aggregate. Let $S_v = S \cap V$ be the signatures with valid hints. Abort if $|S_v| < t$.
   For each $i \in S_v$, parse $\sigma^i \coloneqq (\sigma_1^i, \sigma_2^i, \sigma_3^i, \sigma_4^i)$. If there exists $i \in S_v$ and $j \in S_v$, such that $\sigma_4^i \neq \sigma_4^j$, then abort and output $\perp$.
2. Interpolate a polynomial $B(T)$ of degree $\leq N - |S_v|$ that passes through the points $\{(0, 1), (\omega^i, 0)_{i \in [N] \setminus S_v}\}$.
3. Aggregate the signature components:

$$\begin{aligned}
\hat{\sigma}_1 &= \frac{1}{N} \cdot \left(\sum_{i \in S_v} B(\omega^i) \cdot \sigma_1^i \right) \\
\hat{\sigma}_2 &= \frac{1}{N} \cdot \left(\sum_{i \in S_v} B(\omega^i) \cdot \sigma_2^i \right) \\
\hat{\sigma}_3 &= \frac{1}{N} \cdot \left(\sum_{i \in S_v} B(\omega^i) \cdot \sigma_3^i \right) \\
\hat{\sigma}_4 &= \sigma_4^{i_1}
\end{aligned}$$

4. Aggregate the verification keys:

$$\mathbf{aVK} = \frac{1}{N} \cdot \left(\sum_{i \in S_v} B(\omega^i) \cdot \mathbf{vk}_i \right)$$

5. Define the following polynomials:

$$\begin{aligned}
\hat{B}(T) &= B(T) \cdot T^t \\
\hat{\mathbf{aVK}}(T) &= \mathbf{aVK} \cdot T^N \\
Q_0(T) &= \frac{B(T) - 1}{T} \\
\mathbf{Q_Z}(T) &= \left(\sum_{i \in V} B(\omega^i) \cdot \mathbf{vk}_i \cdot \frac{L_i^2(T) - L_i(T)}{Z(T)} \right) + \left(\sum_{i \in V} B(\omega^i) \cdot \sum_{j \in V \setminus \{i\}} \mathbf{vk}_j \cdot \frac{L_i(T) \cdot L_j(T)}{Z(T)} \right) \\
\mathbf{Q_x}(T) &= \sum_{i \in V} B(\omega^i) \cdot \mathbf{vk}_i \cdot \frac{L_i(T) - L_i(0)}{T} \\
\hat{\mathbf{Q}}_\mathbf{x}(T) &= \sum_{i \in V} B(\omega^i) \cdot \mathbf{vk}_i \cdot T \left(L_i(T) - L_i(0) \right)
\end{aligned}$$

6. Compute $B = [B(\tau)]_1$, $\hat{B} = [\hat{B}(\tau)]_2$, $\hat{\mathbf{aVK}} = [\hat{\mathbf{aVK}}(\tau)]_2$, $Q_0 = [Q_0(\tau)]_2$, $\mathbf{Q_Z} = [\mathbf{Q_Z}(\tau)]_2$, $\mathbf{Q_x} = [\mathbf{Q_x}(\tau)]_2$, $\hat{\mathbf{Q}}_\mathbf{x} = [\hat{\mathbf{Q}}_\mathbf{x}(\tau)]_2$.
7. Output $\sigma = (\hat{\sigma}_1, \hat{\sigma}_2, \hat{\sigma}_3, \hat{\sigma}_4, \mathbf{aVK}, \hat{\mathbf{aVK}}, B, \hat{B}, Q_0, \mathbf{Q_Z}, \mathbf{Q_x}, \hat{\mathbf{Q}}_\mathbf{x})$

- AggrVerify $(\mathbf{crs}, \mathbf{VK}, t, m, \sigma)$:

1. Parse $\sigma = (\hat{\sigma}_1, \hat{\sigma}_2, \hat{\sigma}_3, \hat{\sigma}_4, \mathbf{aVK}, \hat{\mathbf{aVK}}, B, \hat{B}, Q_0, \mathbf{Q_Z}, \mathbf{Q_x}, \hat{\mathbf{Q}}_\mathbf{x})$
2. Perform the following checks:
   (a) $B \circ \mathbf{VK} = [1]_1 \circ \mathbf{aVK} + [\tau]_1 \circ \mathbf{Q_x} + \left[\tau^N - 1\right]_1 \circ \mathbf{Q_Z}$
   (b) $B \circ [\tau^t]_2 = [1]_1 \circ \hat{B}$
   (c) $B \circ [1]_2 = [\tau]_1 \circ Q_0 + [1]_T$
   (d) $[1]_1 \circ \hat{\mathbf{aVK}} = [\tau^N]_1 \circ \mathbf{aVK}$
   (e) $[\tau^2]_1 \circ \mathbf{Q_x} = [1]_1 \circ \hat{\mathbf{Q}}_\mathbf{x}$
   (f) $\hat{\sigma}_1 \circ [\mathbf{A}]_2 = \begin{pmatrix} [1]_1 & m \end{pmatrix} \circ \mathbf{aVK} + \hat{\sigma}_2 \circ [\mathbf{UA}]_2 + \hat{\sigma}_3 \circ [\mathbf{VA}]_2$
   (g) $\hat{\sigma}_2 \circ \hat{\sigma}_4 = \hat{\sigma}_3 \circ [1]_2$

   If any of the checks fail, then output 0 (reject). Else output 1 (accept).

We prove in Appendix B that the threshold signature scheme constructed in Definition 3.2 satisfies correctness (Definition B.1) and security (Definition B.2).

4 Threshold Anonymous Credentials with Silent Setup

We will use the STSPS scheme from Section 3 to build an anonymous credential scheme that supports silent setup and threshold issuance with dynamic thresholds. At a high level, the user requests partial signatures on a KZG commitment to a list of attributes, and generates a zero-knowledge Groth-Sahai proof that they have an valid aggregate signature for the given attributes.

4.1 Syntax

Our definitions are based on [12], section 4.

Definition 4.1 (Syntax)**.** A threshold anonymous credential scheme with silent setup is a tuple of algorithms defined as follows:

- $\mathsf{Init}(1^\lambda, 1^N, 1^l) \to \mathsf{pp}$: Takes as inputs the security parameter (λ), the maximum size of the "committee" the verifier can request credentials from (N), and the total length of the attributes (l), all expressed in unary. Then Init generates the public parameters pp.
- $\mathsf{IssuerRegister}(\mathsf{pp}) \to (\mathsf{vk}_I, \mathsf{hint}_I, \mathsf{sk}_I)$: Run by each issuer individually when they register to the system for the first time.
- $\mathsf{Preprocess}(\mathsf{pp}, \mathcal{I}, (\mathsf{hint}_I, \mathsf{vk}_I)_{I\in\mathcal{I}}) \to (\mathsf{VK}, \mathsf{ak})$: Takes the set $\mathcal{I}$ of issuers the verifier will accept credentials from, as well as all their public keys and hints. Returns an aggregated verification key VK that the verifier can use instead of the individual public keys, and an aggregation key ak that is used to aggregate partial credentials.
- $\mathsf{Issue}(U(\mathsf{vk}_I, \mathsf{attrs}, \mathsf{idx}), I(\mathsf{sk}_I, \mathsf{attrs}', \mathsf{idx})) \to \mathsf{cred}$: The issuing protocol between the user U and the issuer I. The user recieves the issuer public key vk_I, an array $\mathsf{attrs} := (\mathsf{a}_j)_{j\in[l]}$ of attributes $\mathsf{a}_j \in \{0,1\}^*$, and a set $\mathsf{idx} \subset [l]$ of revealed indices. The issuer I has as inputs its own secret key sk_I, an array of disclosed attributes $\mathsf{attrs}' := (\mathsf{attrs}[\mathsf{idx}[j]])_{j\in[\mathsf{idx}]}$, and indices idx. The user outputs a partial credential cred_k over attributes attrs if $\forall j \in [\mathsf{idx}], \mathsf{attrs}'[j] = \mathsf{attrs}[\mathsf{idx}[j]]$, or $\perp$ otherwise.
- $\mathsf{AggrCred}(\mathsf{pp}, \mathsf{attrs}, S, t, \mathsf{ak}, \{\mathsf{cred}_k\}_{k\in S}) \to \mathsf{cred}$: The aggregation protocol run by the user. Takes as input the array of attributes attrs, the set of parties $S \subset [N]$ that issued the partial credentials, the threshold t, the aggregation key ak, and the partial credentials $\mathsf{creds} := [\mathsf{cred}_j]_{j\in S}$ that the user has acquired over attrs. It outputs a credential cred over attrs with threshold t if $|S| = t$, and S has no duplicates.
- $\mathsf{Show}(\mathsf{cred}, \mathsf{VK}, \mathsf{attrs}, \mathsf{idx}, t) \to (\mathsf{attrs}', \mathsf{idx}, \pi)$: Run by the user to show their credential to the verifier. attrs' is the array of disclosed attributes, with indices in $\mathsf{idx} \subset [l]$. π is a proof that the user possesses t signatures on attrs.
- $\mathsf{Verify}(\mathsf{attrs}', \mathsf{idx}, \pi, t, \mathsf{VK}) \to \{0,1\}$: Verifies the credential. Inputs are an array of disclosed attributes attrs', indices $\mathsf{idx} \subset [l]$, and the threshold t. The Verifier outputs 1 if cred is a valid credential issued by at least t issuers over attributes attrs such that $\forall j \in [\mathsf{idx}], \mathsf{attrs}'[j] = \mathsf{attrs}[\mathsf{idx}[j]]$, or 0 otherwise.

Efficiency We require that IssuerRegister does not require any prior coordination on the part of the issuers; it can be run individually as each issuer comes online. We also require that the output size of Show and the running time for Verify be constant in N (they do not grow with the number of issuers).

To achieve the latter property, we use Preprocess to aggregate the public keys offline, without knowing the threshold or the user.

4.2 Construction

We give the following construction for our silent threshold anonymous credentials system.

Notation. GS is the Groth-Sahai proof system. GS.Prove takes an input a set of relations and a witness, and outputs a GS proof of knowledge.

Construction 4.1 (Silent-Setup Threshold Anonymous Credentials)**.** The system has the following algorithms:

- $\mathsf{Init}(1^\lambda, 1^N, 1^l) \rightarrow \mathsf{pp}$: Generate the parameters for the setup of the STSPS scheme. $\mathsf{Setup}(1^\lambda, 1^N) \rightarrow \mathsf{pp}_{\mathsf{STSPS}}, \mathbf{crs}_{\mathsf{STSPS}}$, (defined in Section 3). Generate a new $\mathbf{crs}_{\mathsf{KZG}}$ for the KZG commitment scheme for vectors of length l and $\mathbf{crs}_{\mathsf{GS}}$ for Groth-Sahai. Return $\mathsf{pp} = (\mathsf{pp}_{\mathsf{STSPS}}, \mathbf{crs}_{\mathsf{STSPS}}, \mathbf{crs}_{\mathsf{GS}}, \mathbf{crs}_{\mathsf{KZG}})$.
- $\mathsf{IssuerRegister}(\mathsf{pp}) \rightarrow (\mathsf{vk}_I, \mathsf{hint}_I, \mathsf{sk}_I)$: The issuer generates keys and hints for the STSPS. Run $\mathsf{KGen}(\mathsf{pp}_{\mathsf{STSPS}}) \rightarrow (\mathsf{sk}_I, \mathsf{vk}_I)$, and $\mathsf{HintGen}(\mathsf{pp}_{\mathsf{STSPS}}, \mathsf{sk}_I, I, N) \rightarrow \mathsf{hint}_I$. The issuer publishes $(\mathsf{vk}_I, \mathsf{hint}_I)$, and stores sk_I.
- $\mathsf{Preprocess}(\mathsf{pp}, [N], (\mathsf{hint}_k, \mathsf{vk}_k)_{k\in[N]}) \rightarrow (\mathsf{VK}, \mathsf{ak})$: Run $\mathsf{Preprocess}(\mathbf{crs}_{\mathsf{STSPS}}, (\mathsf{hint}_i, \mathsf{vk}_i)_{i\in[N]}) \rightarrow (\mathsf{VK}, \mathsf{ak})$ from the STSPS (Section 3).
- $\mathsf{Issue}(U(\mathsf{vk}_{\mathcal{I}}, \mathsf{attrs}, \mathsf{idx}), I(\mathsf{sk}_{\mathcal{I}}, \mathsf{attrs}', \mathsf{idx})) \rightarrow \mathsf{cred}_j$:
  1. $U(\mathsf{vk}_I, \mathsf{attrs}, \mathsf{idx})$:
     - Set $\mathsf{attrs}' := (\mathsf{attrs}[\mathsf{idx}[j]])_{j\in[\mathsf{idx}]}$.
     - Commit to the attrs array with KZG. Set $\mathsf{com} = \mathsf{KZG.Commit}(\mathbf{crs}_{\mathsf{KZG}}, \mathsf{attrs})$.
     - Create a batch KZG opening proof. Get the remainder polynomial r_{π_0}, the zero polynomial Z_{π_0} and the proof π_0 from calling $\mathsf{KZG.BatchProve}(\mathbf{crs}_{\mathsf{KZG}}, \mathsf{attrs}, \mathsf{idx})$.
     - Send $(\mathsf{com}, \pi_0, r_{\pi_0}, Z_{\pi_0}, \mathsf{attrs}', \mathsf{idx})$ to the Issuer I.
  2. $I(\mathsf{sk}_I, \mathsf{attrs}', \mathsf{idx})$:
     - Check π_0 by verifying $\mathsf{KZG.BatchVerify}(\mathbf{crs}_{\mathsf{KZG}}, \mathsf{com}, \mathsf{idx}, \mathsf{attrs}', r_{\pi_0}, Z_{\pi_0}\pi_0)$. If it does not verify, return $\perp$.
     - Create a partial signature on the commitment. Return $\mathsf{PartialSign}(\mathsf{pp}_{\mathsf{STSPS}}, \mathsf{sk}_I, \mathsf{com})$, which is either $\Sigma_I = (\sigma_{I,1}, \sigma_{I,2}, \sigma_{I,3}, \sigma_4)$ or $\perp$.
- $\mathsf{AggrCred}(\mathsf{pp}, \mathsf{attrs}, S, t, \mathsf{ak}, \{\mathsf{cred}_k\}_{k\in S}) \rightarrow \mathsf{cred}$:
  - Check that $\forall k \in S, \mathsf{PartialVerify}(\mathbf{crs}_{\mathsf{STSPS}}, \mathsf{vk}_i, \mathsf{com}, \mathsf{cred}_K) = 1$
  - Run $\mathsf{AggrSign}(\mathsf{pp}_{\mathsf{STSPS}}, \mathsf{ak}, t, S, \{\mathsf{cred}_j\}_{j\in S}) \rightarrow \sigma$ and return $\mathsf{cred} = \sigma$.
- $\mathsf{Show}(\mathsf{cred}, \mathsf{VK}, \mathsf{attrs}, \mathsf{idx}, t) \rightarrow (\mathsf{attrs}', \mathsf{idx}, \pi)$:
  - For showing a credential, parse $\mathsf{cred} = \sigma = (\hat{\sigma}_1, \hat{\sigma}_2, \hat{\sigma}_3, \hat{\sigma}_4, \mathbf{aVK}, \hat{\mathbf{aVK}}, B, \hat{B}, Q_0, \mathbf{Q_Z}, \mathbf{Q_x}, \hat{\mathbf{Q}}_\mathbf{x})$
  - Set $\mathsf{attrs}' := (\mathsf{attrs}[\mathsf{idx}[j]])_{j\in[\mathsf{idx}]}$
  - Create a batch KZG opening proof. Get the remainder polynomial r_{π_1}, the zero polynomial Z_{π_1} and the proof π_1 from calling $\mathsf{KZG.BatchProve}(\mathbf{crs}_{\mathsf{KZG}}, \mathsf{com}, \mathsf{attrs}', \mathsf{idx})$.
  - Prove the following relations using Groth-Sahai, where the highlighted terms are witness variables.
    Let $\pi_2 = \mathsf{GS.Prove}$:
    1. $B \circ \mathsf{VK} = [1]_1 \circ \mathbf{aVK} + [\tau]_1 \circ \mathbf{Q_x} + \left[\tau^N - 1\right]_1 \circ \mathbf{Q_Z}$
    2. $B \circ [\tau^t]_2 = [1]_1 \circ \hat{B}$
    3. $B \circ [1]_2 = [\tau]_1 \circ Q_0 + [1]_T$
    4. $[1]_1 \circ \hat{\mathbf{aVK}} = [\tau^N]_1 \circ \mathbf{aVK}$
    5. $[\tau^2]_1 \circ \mathbf{Q_x} = [1]_1 \circ \hat{\mathbf{Q}}_\mathbf{x}$
    6. $\mathsf{com} \circ [1]_2 = \pi_1 \circ Z_{\pi_1} + r_{\pi_1} \circ [1]_2$
    7. $\hat{\sigma}_2 \circ \hat{\sigma}_4 = \hat{\sigma}_3 \circ [1]_2$
    8. $\hat{\sigma}_1 \circ [\mathbf{A}]_2 = [[1]_1, \mathsf{com}] \circ \mathbf{aVK} + \hat{\sigma}_2 \circ [\mathbf{UA}]_2 + \hat{\sigma}_3 \circ [\mathbf{VA}]_2$

- Return $(\mathsf{attrs}', \mathsf{idx}, \pi_2)$.

- $\mathsf{Verify}(\mathsf{attrs}', \mathsf{idx}, \pi, t, \mathsf{VK}) \to \{0, 1\}$:
  - Check $\mathsf{GS.Verify}(\pi_2)$. Return 1 if the proof verifies, else return 0.

4.3 Security

Our scheme satisfies several security properties, which are inspired by section 5.2 of [41]. At showing time, the user should not have to reveal which issuers issued partial credentials (*issuer-hiding*). Furthermore, multiple showings of the same credential should be unlinkable to each other and to the issuing sessions that produced the aggregate credential (*unlinkability*). It should be hard to forge a valid showing unless t issuers issued partial credentials on the revealed attributes (*unforgeability*). Finally, the issuers should not learn anything about the attributes that were not revealed during the issuing protocol (*blind issuing*).

Unlinkability says that the verifier should not know which attributes underlie the credential, beyond the subset of revealed attributes. In the unlinkability security game, Fig. 5, the adversary corrupts all the issuers and chooses two sets of attributes $\mathsf{attrs}_0, \mathsf{attrs}_1$ that agree on some indices idx. The adversary can issue credentials on these attributes on behalf of any issuers. Finally, the challenger generates a showing based on attrs_0 or attrs_1, and the adversary should not be able to guess which attributes were used in the showing, except with negligible advantage. Definition C.1 and Fig. 5 give the definition of unlinkability, and Theorem C.1 proves that our construction satisfies unlinkability.

Unforgeability says that the adversary should not be able to generate valid a showing on some attrs^* unless at least t issuers issued credentials that are consistent with attrs^* on the revealed indices. In the unforgeability security game (Fig. 6), the adversary can corrupt some of the issuers and make issuing queries to the honest issuers on any attributes of their choice. In the end, the adversary outputs a forged credential showing for some attributes attrs^* revealed on indices idx^*. Let M be the number of honest issuers who issued a partial credential on some attrs that agree with attrs^* on indices idx^*. The adversary wins if the showing passes verification, but $M + |\mathsf{Cor}| < t$. Finally, note that during issuance, the user reveals all attributes to the issuer. Definition C.2 and Fig. 6 give the definition of unforgeability, and Theorem C.2 proves that our construction satisfies unforgeability.

Issuer-hiding says that a credential showing should not reveal which issuers issued the partial credentials for the showing, except what is revealed by the attributes. In the issuer-hiding security game (Fig. 7), the adversary chooses two sets of issuers I_0, I_1 of the same size and two sets of attributes $\mathsf{attrs}_0, \mathsf{attrs}_1$ that agree on the revealed indices idx. Then for each $b \in \{0, 1\}$, the challenger requests partial credentials on attrs_b from each issuer in I_b. Then the challenger samples $b' \xleftarrow{\$} \{0, 1\}$ and generates a showing on the credentials obtained for $\mathsf{attrs}_{b'}$ that reveals only the attributes at indices idx. The adversary should not be able to guess b' except with negligible advantage. Definition C.3 and Fig. 7 give the definition of issuer-hiding, and Theorem C.3 proves that our construction satisfies issuer-hiding.

Blind-Issuing says that a malicious issuer should not be able to link a set of attributes to a particular signing session, even if they know the attributes in advance. Intuitively, this means that issuers do not learn anything about the hidden attributes. This property can be added straightforwardly by enforcing that the last attribute in attrs be a uniformly random element r that is chosen by the user. This ensures that the KZG commitment scheme is perfectly hiding. In addition, we must allow the user to only reveal a subset of attributes during Issue.

4.4 Optimizing Groth-Sahai Proofs

In order to show a credential, it is sufficient to provide a zero-knowledge proof of the relation from Construction 4.1, where the highlighted terms are witness variables. The zero-knowledge property guarantees unlinkability across showings. As stated, one can use a Groth-Sahai [32] proof, but the concrete overhead of the proof system can be quite high, especially when trying to prove many different pairing product equations (PPEs). In this section we show how we can compress pairing product equations that are *linear*, thereby reducing both the prover and verifier cost.

Linear Relations. Consider "linear" relations of the form $[A]_1 \circ [w]_2 = [b]_T$, where $[A]_1 \in \mathbb{G}_1^{m \times n}$, $[w]_2 \in \mathbb{G}_2^n$, and $[b]_T \in \mathbb{G}_T^m$. Instead of proving m different PPEs, we show that in the GGM, it sufficient to prove a single PPE.

- Verifier samples $s \leftarrow\!\!\$\ \mathbb{F}^m$ and sends
  - $\mathcal{K} = ([c_1]_1, \ldots, [c_n]_1) \leftarrow s^T \cdot [A]_1$,
  - $[d]_T \leftarrow s^T \cdot [b]_T$.

  to the prover. (Note that this step is reusable across provers, so this message can be sampled as part of the common reference string (with trusted setup).)
- Prove generates a Groth-Sahai NIZK for the relation

$$\{[w]_2 \mid \mathcal{K} \circ [w]_2 = [d]_T\}$$

- The verifier accepts if the Groth-Sahai proof passes verification.

Completeness and Zero-knowledge follow from the completeness and zero-knowledge property of the Groth-Sahai proof system.

Knowledge Soundness. In the GGM, we treat the randomness s that is sampled by the prover as formal variables. Now observe that from knowledge soundness we extract $[w]_2$ such that:

$$\mathcal{K} \circ [w]_2 = [d]_T.$$

Note that the right hand-side of the equation only has degree-1 terms in s, and all terms in $\mathcal{K}$ have degree-1 in s. Hence $[w]_2$ can only contain constant terms (in the formal variable s).

Now observe that each $c_j = \sum_{i \in [m]} s_i \cdot A_{i,j}$. We can thus rewrite the above equation as

$$\sum_{j \in [n]} \left(\left(\sum_{i \in [m]} s_i \cdot A_{i,j} \right) \circ [w_j]_2 \right) = \sum_{i \in [m]} s_i \cdot [b_i]_T$$
$$\implies \left\{ \sum_{j \in [n]} A_{i,j} \circ [w_j]_2 = [b_i]_T \right\}_{i \in [m]}$$

where the implication comes because the coefficients of s_i on the left and right hand sides of the equality must be the same. Hence the same witness $[w_j]_2$ will satisfy $[A]_1 \circ [w]_2 = [b]_T$.

Thus, it is now sufficient to prove just three (albeit larger) pairing product equations.

1. $$\begin{bmatrix} B \\ \mathbf{aVK}[0] \\ \mathbf{aVK}[1] \\ \mathbf{Q_x}[0] \\ \mathbf{Q_x}[1] \\ \mathbf{Qz}[0] \\ \mathbf{Qz}[1] \\ \hat{B} \\ Q_0 \\ \mathbf{a\hat{V}K}[0] \\ \mathbf{a\hat{V}K}[1] \\ \mathbf{\hat{Q}_x}[0] \\ \mathbf{\hat{Q}_x}[1] \\ \mathsf{com} \\ \pi_1 \end{bmatrix} \circ \mathcal{K} = (s_3 \cdot [1]_T) + (s_9 \cdot [r_{\pi_1}]_T)$$
2. $\hat{\sigma}_2 \circ \hat{\sigma}_4 = \hat{\sigma}_3 \circ [1]_2$
3. $\hat{\sigma}_1 \circ [\mathbf{A}]_2 = [[1]_1, \mathsf{com}] \circ \mathbf{aVK} + \hat{\sigma}_2 \circ [\mathbf{UA}]_2 + \hat{\sigma}_3 \circ [\mathbf{VA}]_2$

where

1. $\mathcal{K}_1 = s_0 \cdot \mathsf{VK}[0] + s_1 \cdot \mathsf{VK}[1] + s_2 \cdot [\tau^t]_2 + s_3 \cdot [1]_2$
2. $\mathcal{K}_2 = -(s_0 \cdot [1]_1 + s_4 \cdot [\tau^N])$
3. $\mathcal{K}_3 = -(s_1 \cdot [1]_1 + s_5 \cdot [\tau^N])$
4. $\mathcal{K}_4 = -(s_0 \cdot [\tau]_1 + s_6 \cdot [\tau^2]_1)$
5. $\mathcal{K}_5 = -(s_1 \cdot [\tau]_1 + s_7 \cdot [\tau^2]_1)$
6. $\mathcal{K}_6 = -(s_0 \cdot \left[\tau^N - 1\right]_1)$
7. $\mathcal{K}_7 = -(s_1 \cdot \left[\tau^N - 1\right]_1)$
8. $\mathcal{K}_8 = -(s_2 \cdot [1]_1)$
9. $\mathcal{K}_9 = -(s_3 \cdot [\tau]_1)$
10. $\mathcal{K}_{10} = s_4 \cdot [1]_1$
11. $\mathcal{K}_{11} = s_5 \cdot [1]_1$
12. $\mathcal{K}_{12} = s_6 \cdot [1]_1$
13. $\mathcal{K}_{13} = s_7 \cdot [1]_1$
14. $\mathcal{K}_{14} = s_9 \cdot [1]_1$
15. $\mathcal{K}_{15} = -(s_9 \cdot Z_{\pi_1})$

5 Implementation and Evaluation

To evaluate the concrete performance of our scheme, we implemented our scheme in Rust using `arkworks` [2] for implementations of pairing-friendly curves. Our implementation can be found at https://github.com/guruvamsi-policharla/silent-anonymous-credentials.

Algorithm	Time (ms)
PartialSign	8.6
PartialVerify	8.6
AggrSign($n = 2^6$)	71.2
AggrSign($n = 2^8$)	232.6
AggrSign($n = 2^{10}$)	519.1
AggrVerify	27.3

Table 1: Timings of our threshold SPS scheme with silent setup using $t = n/2$. All reported numbers are in single threaded mode.

Algorithm	—Show—	Silent Setup	Blackbox
[52]	$\mathbb{G}_1 + 3\,\mathbb{G}_2$ $+(m+2)\,\mathbb{F}$	✗	✓
[48]	$\mathbb{G}_1 + 2\,\mathbb{G}_2$	✓	✗
[41]	$(2t+4)\,\mathbb{G}_1 + 4t\,\mathbb{G}_2$	✓	✓
Ours	$44\,\mathbb{G}_1 + 32\,\mathbb{G}_2$	✓	✓

Table 2: Comparison of the sizes of showings with n issuers, with a threshold of t, and m attributes. We compare against the scheme with a smaller showing size from [41] and ignore the cost of an additional zero-knowledge proof the user needs to send. [48] does not implement decentralized issuance but the showing communication does not change as the SNARK used is Groth16 [31] which always has a constant sized proof.

Setup. All of our experiments were run on a 2019 MacBook Pro with a 2.4 GHz Intel Core i9 processor and 16 GB of DDR4 RAM in single-threaded mode. We use BLS12-381 [5] as our pairing friendly curve. Compared to the construction in Definition 3.2, our implementation swaps the roles of $\mathbb{G}_1$ and $\mathbb{G}_2$. This change of notation leads to efficiency gains since group operations in $\mathbb{G}_1$ are less expensive than operations in $\mathbb{G}_2$.

Dealer. We assume a trusted setup for powers-of-tau common reference string, but this can be carried out in a distributed manner, such that as long as at least one party is honest, the overall protocol will be secure. Alternatively, one can use a common reference string that has already been set up in a *decentralized* manner with contributions from many parties.[9]

[9]For instance, https://ceremony.ethereum.org has 141,416 contributions. The ZCash foundation also carried it out for polynomials of degree $< 2^{21}$ https://zfnd.org/conclusion-of-the-powers-of-tau-ceremony.

Algorithm	Ours (BLS12-381)	Ours (BN254)	[41] $n = 2$	[41] $n = 10$	[52]
Show	71.6	35.8	20	45	1.5
Verify	76.2	47.5	41	119	10.5

Table 3: Timings of our anonymous credential scheme with silent setup compared against prior work with n issuers. All times in milliseconds. Show and Verify timings remain the same irrespective of the number of issuers. All reported numbers are in single threaded mode. We additionally report numbers over BN254 (which only offers 100 bits of security) as prior work [41, 52] use it to report their performance. [52] does not support silent setup but we provide numbers here to understand the overhead incurred due to the silent setup in our scheme.

Benchmarks. We benchmark the time taken by various algorithms in our threshold SPS with silent setup Definition 3.2 in Table 1. We also provide benchmarks of our threshold issuance anonymous credential scheme with silent setup in Table 3. All algorithms run in constant time except for AggrSign which grows with the number of issuers as expected. We emphasize here that the AggrSign method only needs to be run once per verifier policy. In the anonymous credential benchmarks we only provide Show and Verify timings, as timings of Issue and AggrCred are essentially the same as the timings of PartialSign and AggrSign respectively. The size of the crs and the public key of each party grows with the maximum possible size of the verification policy – i.e. the size of the set of acceptable issuers. Concretely, for a maximum verifier policy size of 1024, with $30,000$ possible issuers to choose from amounts to about 3 GB of public keys and hints that are needed to produce a showing. For context, the Uniswap DAO, currently the largest one [10], has about 2800 actively participating members.

In Table 3 we note that [41] did not have a link to their implementation and hence performance numbers are taken directly from their paper for $n = 10$ issuers. Our implementation uses a different pairing library and different machines to benchmark. Hence these benchmarks only offer a rough comparison against prior work. But it is clear that our work will outperform the state of the art [41] as the number of issuers grows.

From Table 2 it is clear that when the number of issuers is larger than 10, our scheme is favorable in terms of showing size. We expect that the concrete costs of our scheme can be reduced by optimizing the Groth-Sahai proofs further, and by designing a more efficient aggregatable structure preserving signature scheme in the GGM.

6 Acknowledgments

The research was supported in part by the AFOSR Award FA9550-24-1-0156 and research grants from the Bakar Fund, Supra Inc., and the Stellar Development Foundation.

References

[1] Masayuki Abe, Georg Fuchsbauer, Jens Groth, Kristiyan Haralambiev, and Miyako Ohkubo. Structure-preserving signatures and commitments to group elements. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 209–236. Springer, Berlin, Heidelberg, August 2010.

[2] arkworks contributors. `arkworks` zksnark ecosystem. https://arkworks.rs, 2022.

[10] https://coinspaidmedia.com/xdao/special-xdao-famous/

[3] Man Ho Au, Willy Susilo, and Yi Mu. Constant-size dynamic k-TAA. In Roberto De Prisco and Moti Yung, editors, *SCN 06*, volume 4116 of *LNCS*, pages 111–125. Springer, Berlin, Heidelberg, September 2006.

[4] Foteini Baldimtsi and Anna Lysyanskaya. Anonymous credentials light. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 2013*, pages 1087–1098. ACM Press, November 2013.

[5] Paulo S. L. M. Barreto, Ben Lynn, and Michael Scott. Constructing elliptic curves with prescribed embedding degrees. In Stelvio Cimato, Clemente Galdi, and Giuseppe Persiano, editors, *SCN 02*, volume 2576 of *LNCS*, pages 257–267. Springer, Berlin, Heidelberg, September 2003.

[6] Eli Ben-Sasson, Alessandro Chiesa, Michael Riabzev, Nicholas Spooner, Madars Virza, and Nicholas P. Ward. Aurora: Transparent succinct arguments for R1CS. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part I*, volume 11476 of *LNCS*, pages 103–128. Springer, Cham, May 2019.

[7] Jan Bobolz, Fabian Eidens, Stephan Krenn, Sebastian Ramacher, and Kai Samelin. Issuer-hiding attribute-based credentials. In Mauro Conti, Marc Stevens, and Stephan Krenn, editors, *CANS 21*, volume 13099 of *LNCS*, pages 158–178. Springer, Cham, December 2021.

[8] Jan Bobolz, Pooya Farshim, Markulf Kohlweiss, and Akira Takahashi. The brave new world of global generic groups and uc-secure zero-overhead snarks. In Elette Boyle and Mohammad Mahmoody, editors, *Theory of Cryptography*, pages 90–124, Cham, 2025. Springer Nature Switzerland.

[9] Dan Boneh, Manu Drijvers, and Gregory Neven. Compact multi-signatures for smaller blockchains. In Thomas Peyrin and Steven Galbraith, editors, *ASIACRYPT 2018, Part II*, volume 11273 of *LNCS*, pages 435–464. Springer, Cham, December 2018.

[10] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 213–229. Springer, Berlin, Heidelberg, August 2001.

[11] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. In Colin Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 514–532. Springer, Berlin, Heidelberg, December 2001.

[12] Jonathan Bootle, Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Alessandro Sorniotti. A framework for practical anonymous credentials from lattices. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part II*, volume 14082 of *LNCS*, pages 384–417. Springer, Cham, August 2023.

[13] Daniel Bosk, Davide Frey, Mathieu Gestin, and Guillaume Piolle. Hidden issuer anonymous credential. *PoPETs*, 2022(4):571–607, October 2022.

[14] Jan Camenisch, Maria Dubovitskaya, Kristiyan Haralambiev, and Markulf Kohlweiss. Composable and modular anonymous credentials: Definitions and practical constructions. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015, Part II*, volume 9453 of *LNCS*, pages 262–288. Springer, Berlin, Heidelberg, November / December 2015.

[15] Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 93–118. Springer, Berlin, Heidelberg, May 2001.

[16] Jan Camenisch and Anna Lysyanskaya. A signature scheme with efficient protocols. In Stelvio Cimato, Clemente Galdi, and Giuseppe Persiano, editors, *SCN 02*, volume 2576 of *LNCS*, pages 268–289. Springer, Berlin, Heidelberg, September 2003.

[17] Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 56–72. Springer, Berlin, Heidelberg, August 2004.

[18] Jan Camenisch and Els Van Herreweghen. Design and implementation of the idemix anonymous credential system. In Vijayalakshmi Atluri, editor, *ACM CCS 2002*, pages 21–30. ACM Press, November 2002.

[19] Matteo Campanelli, Anca Nitulescu, Carla Ràfols, Alexandros Zacharakis, and Arantxa Zapico. Linear-map vector commitments and their practical applications. In Shweta Agrawal and Dongdai Lin, editors, *ASIACRYPT 2022, Part IV*, volume 13794 of *LNCS*, pages 189–219. Springer, Cham, December 2022.

[20] Melissa Chase, Trevor Perrin, and Greg Zaverucha. The Signal private group system and anonymous credentials supporting efficient verifiable encryption. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 2020*, pages 1445–1459. ACM Press, November 2020.

[21] Aisling Connolly, Pascal Lafourcade, and Octavio Perez-Kempner. Improved constructions of anonymous credentials from structure-preserving signatures on equivalence classes. In Goichiro Hanaoka, Junji Shikata, and Yohei Watanabe, editors, *PKC 2022, Part I*, volume 13177 of *LNCS*, pages 409–438. Springer, Cham, March 2022.

[22] Elizabeth C. Crites, Markulf Kohlweiss, Bart Preneel, Mahdi Sedaghat, and Daniel Slamanig. Threshold structure-preserving signatures. In Jian Guo and Ron Steinfeld, editors, *ASIACRYPT 2023, Part II*, volume 14439 of *LNCS*, pages 348–382. Springer, Singapore, December 2023.

[23] Sourav Das, Philippe Camacho, Zhuolun Xiang, Javier Nieto, Benedikt Bünz, and Ling Ren. Threshold signatures from inner product argument: Succinct, weighted, and multi-threshold. In Weizhi Meng, Christian Damsgaard Jensen, Cas Cremers, and Engin Kirda, editors, *ACM CCS 2023*, pages 356–370. ACM Press, November 2023.

[24] Alex Davidson, Ian Goldberg, Nick Sullivan, George Tankersley, and Filippo Valsorda. Privacy pass: Bypassing internet challenges anonymously. *PoPETs*, 2018(3):164–180, July 2018.

[25] Dominic Deuber, Matteo Maffei, Giulio Malavolta, Max Rabkin, Dominique Schröder, and Mark Simkin. Functional credentials. *PoPETs*, 2018(2):64–84, April 2018.

[26] Jack Doerner, Yashvanth Kondi, Eysa Lee, abhi shelat, and LaKyah Tyner. Threshold BBS+ signatures for distributed anonymous credential issuance. In *2023 IEEE Symposium on Security and Privacy*, pages 773–789. IEEE Computer Society Press, May 2023.

[27] Georg Fuchsbauer, Christian Hanser, and Daniel Slamanig. Structure-preserving signatures on equivalence classes and constant-size anonymous credentials. *Journal of Cryptology*, 32(2):498–546, April 2019.

[28] Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 33–62. Springer, Cham, August 2018.

[29] Sanjam Garg, Abhishek Jain, Pratyay Mukherjee, Rohit Sinha, Mingyuan Wang, and Yinuo Zhang. hinTS: Threshold signatures with silent setup. In *2024 IEEE Symposium on Security and Privacy*, pages 3034–3052. IEEE Computer Society Press, May 2024.

[30] Sanjam Garg, Dimitris Kolonelos, Guru-Vamsi Policharla, and Mingyuan Wang. Threshold encryption with silent setup. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part VII*, volume 14926 of *LNCS*, pages 352–386. Springer, Cham, August 2024.

[31] Jens Groth. On the size of pairing-based non-interactive arguments. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 305–326. Springer, Berlin, Heidelberg, May 2016.

[32] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 415–432. Springer, Berlin, Heidelberg, April 2008.

[33] Lucjan Hanzlik and Daniel Slamanig. With a little help from my friends: Constructing practical anonymous credentials. In Giovanni Vigna and Elaine Shi, editors, *ACM CCS 2021*, pages 2004–2023. ACM Press, November 2021.

[34] Chloé Hébant and David Pointcheval. Traceable constant-size multi-authority credentials. In Clemente Galdi and Stanislaw Jarecki, editors, *SCN 22*, volume 13409 of *LNCS*, pages 411–434. Springer, Cham, September 2022.

[35] Antoine Joux. Multicollisions in iterated hash functions. Application to cascaded constructions. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 306–316. Springer, Berlin, Heidelberg, August 2004.

[36] Ioanna Karantaidou, Omar Renawi, Foteini Baldimtsi, Nikolaos Kamarinakis, Jonathan Katz, and Julian Loss. Blind multisignatures for anonymous tokens with decentralized issuance. In Bo Luo, Xiaojing Liao, Jun Xu, Engin Kirda, and David Lie, editors, *ACM CCS 2024*, pages 1508–1522. ACM Press, October 2024.

[37] Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. Constant-size commitments to polynomials and their applications. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 177–194. Springer, Berlin, Heidelberg, December 2010.

[38] Dmitry Khovratovich, Ron D. Rothblum, and Lev Soukhanov. How to prove false statements: Practical attacks on fiat-shamir. Cryptology ePrint Archive, Report 2025/118, 2025.

[39] Eike Kiltz, Jiaxin Pan, and Hoeteck Wee. Structure-preserving signatures from standard assumptions, revisited. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 275–295. Springer, Berlin, Heidelberg, August 2015.

[40] Ben Kreuter, Tancrède Lepoint, Michele Orrù, and Mariana Raykova. Anonymous tokens with private metadata bit. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part I*, volume 12170 of *LNCS*, pages 308–336. Springer, Cham, August 2020.

[41] Omid Mir, Balthazar Bauer, Scott Griffy, Anna Lysyanskaya, and Daniel Slamanig. Aggregate signatures with versatile randomization and issuer-hiding multi-authority anonymous credentials. In Weizhi Meng, Christian Damsgaard Jensen, Cas Cremers, and Engin Kirda, editors, *ACM CCS 2023*, pages 30–44. ACM Press, November 2023.

[42] Aikaterini Mitrokotsa, Sayantan Mukherjee, Mahdi Sedaghat, Daniel Slamanig, and Jenit Tomy. Threshold structure-preserving signatures: Strong and adaptive security under standard assumptions. In Qiang Tang and Vanessa Teague, editors, *PKC 2024, Part I*, volume 14601 of *LNCS*, pages 163–195. Springer, Cham, April 2024.

[43] V. I. Nechaev. Complexity of a determinate algorithm for the discrete logarithm. *Mathematical Notes*, 55(2):165–172, February 1994.

[44] Christian Paquin and Greg Zaverucha. U-prove cryptographic specification v1.1 (revision 3), December 2013. Released under the Open Specification Promise (http://www.microsoft.com/openspecifications/en/us/ programs/osp/default.aspx).

[45] David Pointcheval and Olivier Sanders. Short randomizable signatures. In Kazue Sako, editor, *CT-RSA 2016*, volume 9610 of *LNCS*, pages 111–126. Springer, Cham, February / March 2016.

[46] Guru-Vamsi Policharla, Bas Westerbaan, Armando Faz-Hernández, and Christopher A Wood. Post-quantum privacy pass via post-quantum anonymous credentials. Cryptology ePrint Archive, Report 2023/414, 2023.

[47] Carla Ràfols and Arantxa Zapico. An algebraic framework for universal and updatable SNARKs. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part I*, volume 12825 of *LNCS*, pages 774–804, Virtual Event, August 2021. Springer, Cham.

[48] Deevashwer Rathee, Guru Vamsi Policharla, Tiancheng Xie, Ryan Cottone, and Dawn Song. ZEBRA: Anonymous credentials with practical on-chain verification and applications to KYC in DeFi. Cryptology ePrint Archive, Report 2022/1286, 2022.

[49] Olivier Sanders. Efficient redactable signature and application to anonymous credentials. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *PKC 2020, Part II*, volume 12111 of *LNCS*, pages 628–656. Springer, Cham, May 2020.

[50] Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 256–266. Springer, Berlin, Heidelberg, May 1997.

[51] Tjerand Silde and Martin Strand. Anonymous tokens with public metadata and applications to private contact tracing. In Ittay Eyal and Juan A. Garay, editors, *FC 2022*, volume 13411 of *LNCS*, pages 179–199. Springer, Cham, May 2022.

[52] Alberto Sonnino, Mustafa Al-Bassam, Shehar Bano, Sarah Meiklejohn, and George Danezis. Coconut: Threshold issuance selective disclosure credentials with applications to distributed ledgers. In *NDSS 2019*. The Internet Society, February 2019.

[53] Alin Tomescu, Adithya Bhat, Benny Applebaum, Ittai Abraham, Guy Gueta, Benny Pinkas, and Avishay Yanai. UTT: Decentralized ecash with accountable privacy. Cryptology ePrint Archive, Report 2022/452, 2022.

A Extended Preliminaries

A.1 KZG Vector Commitment Scheme

We recall the KZG polynomial commitment scheme [37]. Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a bilinear map with generators $g \in \mathbb{G}_1$ and $h \in \mathbb{G}_2$. The KZG polynomial commitment is also *extractable* in the algebraic group model [28]. Looking ahead, we will be using the KZG polynomial commitment as a vector commitment scheme. For a vector $\mathsf{arr} = (a_0, a_1, \cdots, a_{L-1})$, we interpolate a polynomial $f(X)$ such that $\forall i \in [L-1], f(i) = a_i$, and commit to the polynomial. An opening proof for an index-value pair (i, a_i) is then simply an evaluation proof for $f(i) = a_i$. Similarly, instead of providing $r(X)$ to $\mathsf{BatchVerify}$, we will pass in a set of index-value pairs, which can be interpolated to recover $r(X)$. The KZG vector commitment scheme consists of the following algorithms:

- $\mathsf{Setup}(1^\lambda, 1^L) \to \mathsf{crs}$: A trusted party samples $\tau \leftarrow \mathbb{F}$ and publishes $\mathsf{crs} = (g, g^\tau, \ldots, g^{\tau^{L-1}}, h, h^\tau, \ldots, h^{\tau^{L-1}})$.
- $\mathsf{Commit}(\mathsf{crs}, \mathsf{arr}) \to \mathsf{com}$: Let $f(X)$ be the unique polynomial of degree $< L$ with $f(i) = a_i$ for all $i \in \{0, \ldots, L-1\}$. Output $\mathsf{com} = g^{f(\tau)}$.
- $\mathsf{Open}(\mathsf{crs}, \mathsf{arr}, i^*)$: Let $f(X)$ be as above and $y^* = a_{i^*}$. Define $q(X) = \frac{f(X) - y^*}{X - i^*}$ and output $\pi = g^{q(\tau)}$.
- $\mathsf{Verify}(\mathsf{crs}, \mathsf{com}, i^*, y^*, \pi)$: Accept if and only if $e(\mathsf{com} \cdot g^{-y^*}, h) = e(\pi, h^{\tau - i^*})$.
- $\mathsf{BatchProve}(\mathsf{crs}, \mathsf{arr}, \mathsf{idx}) \to (\mathsf{idx}, \mathsf{arr}', r_\pi, Z_\pi, \pi)$: Let $\mathsf{arr}' = (a_i)_{i \in \mathsf{idx}}$ and $f(X)$ be the unique degree $< L$ polynomial with $f(i) = a_i$ for all i. Let $r(X)$ be the unique degree $< |\mathsf{idx}|$ polynomial with $r(i) = a_i$ for $i \in \mathsf{idx}$, and $Z(X) = \prod_{i \in \mathsf{idx}} (X - i)$. Define $q(X) = \frac{f(X) - r(X)}{Z(X)}$, set $\pi = g^{q(\tau)}$, $r_\pi = g^{r(\tau)}$, and $Z_\pi = h^{Z(\tau)}$.
- $\mathsf{BatchVerify}(\mathsf{crs}, \mathsf{com}, \mathsf{idx}, \mathsf{arr}', r_\pi, Z_\pi, \pi)$: Reconstruct $r(X)$ as the unique degree $< |\mathsf{idx}|$ polynomial with $r(i) = \mathsf{arr}'[i]$ for $i \in \mathsf{idx}$, and compute $Z(X) = \prod_{i \in \mathsf{idx}} (X - i)$. Accept if and only if $r_\pi = g^{r(\tau)}$, $Z_\pi = h^{Z(\tau)}$, and $e(\pi, Z_\pi) \cdot e(r_\pi, h) = e(\mathsf{com}, h)$.

We want KZG commitments that have both *computational binding* and *perfect hiding*. Informally, they guarantee the following properties.

Computational binding states that it is hard for a PPT adversary to compute a commitment c and two opening proofs with revealed indices and values $(\pi_1, \mathsf{idx}_1, \mathsf{arr}'_1)$ and $(\pi_2, \mathsf{idx}_2, \mathsf{arr}'_2)$, such that both π_1 and π_2 verify, and $\exists j$ s.t. $j \in \mathsf{idx}_1$ and $j \in \mathsf{idx}_2$, and $\mathsf{arr}'_1[j] \neq \mathsf{arr}'_2[j]$. The above commitment was shown to be binding in [37] under the q-SDH assumption.

Perfect hiding states that for any arr_1 and arr_2, the distributions of $\mathsf{Commit}(\mathsf{crs}, \mathsf{arr}_1)$ and $\mathsf{Commit}(\mathsf{crs}, \mathsf{arr}_2)$ are identical. To make the commitment hiding, we must sufficiently *randomize* the commitment. Perfect hiding for KZG can be achieved by simply including a uniformly random element r as an extra attribute in arr.

B Threshold Signature Proof of Correctness and Security

We will define and prove notions of correctness and security for our silent threshold structure-preserving signature scheme from Definition 3.2.

B.1 Correctness

Correctness says that an aggregate signature will pass verification if the partial signatures pass verification and aggregation was performed correctly. In more detail, the adversary can corrupt a subset of parties and even generate partial signatures on behalf of those parties. The challenger will honestly aggregate the partial signatures from the corrupted parties (as long as they pass PartialVerify) as well as the ones from the honest parties. In the end, the aggregated signature should be accepted by AggrVerify with overwhelming probability for any threshold less than or equal to the number of partial signatures that were aggregated.

Our correctness definition is similar to [29], definition 4.

Definition B.1 (Correctness)**.** The STS is *correct* if for any non-uniform PPT adversary $\mathcal{A}$, the output of $\mathsf{Correctness\text{-}Exp}_{\mathcal{A}}(1^\lambda)$ (defined in Fig. 1) is 1 with probability $\geq 1 - \mathsf{negl}(\lambda)$.

Theorem B.1. The STSPS scheme constructed in Definition 3.2 satisfies correctness (Definition B.1).

Proof. Every signature σ^i that the challenger aggregates satisfies the following equations:

$$\begin{aligned} \sigma_1 \circ [\mathbf{A}]_2 &= \begin{pmatrix} [1]_1 & m \end{pmatrix} \circ \mathbf{vk} + \sigma_2 \circ [\mathbf{UA}]_2 + \sigma_3 \circ [\mathbf{VA}]_2 \\ \sigma_2 \circ \sigma_4 &= \sigma_3 \circ [1]_2 \end{aligned}$$

These equations are checked by PartialVerify.

Then the aggregate signature components $(\mathbf{aVK}, \hat{\sigma}_1, \hat{\sigma}_2, \hat{\sigma}_3, \hat{\sigma}_4)$ will satisfy the analogous equations of AggrVerify:

$$\begin{aligned} \hat{\sigma}_1 \circ [\mathbf{A}]_2 &= \begin{pmatrix} [1]_1 & m \end{pmatrix} \circ \mathbf{aVK} + \hat{\sigma}_2 \circ [\mathbf{UA}]_2 + \hat{\sigma}_3 \circ [\mathbf{VA}]_2 \\ \hat{\sigma}_2 \circ \hat{\sigma}_4 &= \hat{\sigma}_3 \circ [1]_2 \end{aligned}$$

B.2 Security

We allow the adversary to corrupt a subset of the parties $\mathsf{Cor} \subset [N]$. Then the adversary can query PartialSign to any honest parties on any messages of their choice. In the end, the adversary outputs a forgery (t, m, σ^*). Let Q be the subset of the honest parties to which the adversary made a PartialSign query on message m. The adversary wins the security game if $\mathsf{AggrVerify}(\mathsf{crs}, \mathsf{VK}, t, m, \sigma^*) = 1$, and $t > |Q \cup \mathsf{Cor}|$. The security definition we use here is similar to [29], definition 5.

Definition B.2 (Security)**.** The STS is *secure* (a.k.a. unforgeable) if for any non-uniform PPT adversary $\mathcal{A}$,

$$\Pr\left[\mathsf{Forge\text{-}Exp}_{\mathcal{A}}(1^\lambda) = 1\right] \leq \mathsf{negl}(\lambda)$$

where $\mathsf{Forge\text{-}Exp}_{\mathcal{A}}$ is given in Fig. 2.

Theorem B.2. The threshold signature scheme constructed in Definition 3.2 is secure (Definition B.2) in the generic group model for type-III pairings.

The rest of Appendix B is devoted to proving Theorem B.2.

Variables and Polynomials: Let us define the variables and polynomials that represent the group elements of the unforgeability game.

1. **Setup:** Let $\mathcal{A}$ be the adversary. The adversary is given 1^λ and then picks $N \in \mathbb{N}$. Then the challenger runs $\mathsf{Setup}(1^\lambda, 1^N) \to (\mathsf{pp}, \mathsf{crs})$ and sends $(\mathsf{pp}, \mathsf{crs})$ to the adversary. Then the adversary picks $\mathsf{Cor} \subseteq [N]$, the set of corrupted parties. Let $\mathsf{Hon} = [N] \backslash \mathsf{Cor}$.

2. **Key Generation:**

   (a) For each $i \in \mathsf{Cor}$, the adversary outputs $(\mathsf{vk}_i, \mathsf{hint}_i)$.

   (b) For each $i \in \mathsf{Hon}$, the challenger computes:

   $$\mathsf{KGen}(\mathsf{crs}) \to (\mathsf{sk}_i, \mathsf{vk}_i)$$
   $$\mathsf{HintGen}(\mathsf{crs}, \mathsf{sk}_i, i, N) \to \mathsf{hint}_i$$

   and sends $(\mathsf{vk}_i, \mathsf{hint}_i)$ to the adversary.

   (c) The challenger computes

   $$\mathsf{Preprocess}(\mathsf{crs}, (\mathsf{hint}_i, \mathsf{vk}_i)_{i \in [N]}) \to (\mathsf{VK}, \mathsf{ak})$$

   and sends $(\mathsf{VK}, \mathsf{ak})$ to the adversary.

3. **Partial Signing:**

   (a) The adversary outputs a message m, a (possibly empty) subset of corrupted parties $S_1 \subseteq \mathsf{Cor}$, and a partial signature σ^i for each $i \in S_1$.
   For each $i \in S_1$, the challenger computes

   $$\mathsf{PartialVerify}(\mathsf{crs}, \mathsf{vk}_i, m, \sigma^i)$$

   Let $S_1' \subseteq S_1$ be the subset of parties i for which $\mathsf{PartialVerify}$ accepts σ^i.

   (b) The adversary outputs a (possibly empty) subset of honest parties $S_2 \subseteq \mathsf{Hon}$. For each $i \in S_2$, the challenger computes

   $$\sigma^i = \mathsf{PartialSign}(\mathsf{crs}, \mathsf{sk}_i, m)$$

   sends σ^i to the adversary.

4. **Aggregation and Verification:** The challenger computes the aggregate signature σ as follows:

   $$S = S_1' \cup S_2$$
   $$\sigma = \mathsf{AggrSign}\left(\mathsf{crs}, \mathsf{ak}, |S|, m, S, \left(\sigma^i\right)_{i \in S}\right)$$

   Then for each $t \leq |S|$, the challenger checks that:

   $$1 = \mathsf{AggrVerify}(\mathsf{crs}, \mathsf{VK}, t, m, \sigma)$$

   If all checks pass, then the output of $\mathsf{Correctness\text{-}Exp}_{\mathcal{A}}(1^\lambda)$ is 1. Otherwise, the output is 0.

Figure 1: $\mathsf{Correctness\text{-}Exp}_{\mathcal{A}}(1^\lambda)$

1. **Setup:** Let $\mathcal{A}$ be the adversary. The adversary is given 1^λ and then picks $N \in \mathbb{N}$. Then the challenger runs $\mathsf{Setup}(1^\lambda, 1^N) \to (\mathsf{pp}, \mathsf{crs})$ and sends $(\mathsf{pp}, \mathsf{crs})$ to the adversary. Then the adversary picks $\mathsf{Cor} \subset [N]$, the set of corrupted parties. Let $\mathsf{Hon} = [N] \backslash \mathsf{Cor}$.

2. **Key Generation:**

   (a) For each $i \in \mathsf{Cor}$, the adversary outputs $(\mathsf{vk}_i, \mathsf{hint}_i)$.[a]

   (b) For each $i \in \mathsf{Hon}$, the challenger computes:

   $$\mathsf{KGen}(\mathsf{crs}) \to (\mathsf{sk}_i, \mathsf{vk}_i)$$
   $$\mathsf{HintGen}(\mathsf{crs}, \mathsf{sk}_i, i, N) \to \mathsf{hint}_i$$

   and sends $(\mathsf{vk}_i, \mathsf{hint}_i)$ to the adversary.

   (c) The challenger computes $\mathsf{Preprocess}(\mathsf{crs}, (\mathsf{hint}_i, \mathsf{vk}_i)_{i \in [N]}) \to (\mathsf{VK}, \mathsf{ak})$ and sends them to the adversary.

3. **Signing Queries:** The adversary may query $\mathsf{PartialSign}$ on inputs $(i, m) \in \mathsf{Hon} \times \mathbb{G}_1$. The query's *message* is m, and its *signing party* is i. The challenger records the query, computes $\mathsf{PartialSign}(\mathsf{crs}, \mathsf{sk}_i, m) \to \{\sigma, \perp\}$, and sends the result to the adversary. The adversary can make any number of such signing queries.

4. **Forgery:** The adversary outputs a forgery (t, m^*, σ^*). The challenger verifies the signature by computing $\mathsf{AggrVerify}(\mathsf{crs}, \mathsf{VK}, t, m^*, \sigma^*) \to \{0, 1\}$.

   Let $Q \subseteq \mathsf{Hon}$ be the honest signing parties that were asked to sign message m^* during some signing query. If $t > |Q \cup \mathsf{Cor}|$, and $\mathsf{AggrVerify}(\mathsf{crs}, \mathsf{VK}, t, m^*, \sigma^*) = 1$, then the adversary wins the forgery game, and $\mathsf{Forge\text{-}Exp}_{\mathcal{A}}(1^\lambda)$ outputs 1. Otherwise, $\mathsf{Forge\text{-}Exp}_{\mathcal{A}}(1^\lambda)$ outputs 0.

[a] The corrupted parties choose their vk's before seeing the honest parties' vks. This ensures that the adversary cannot choose their keys based on the honest parties' keys and launch a rogue public key attack. In practice, when we cannot require the adversary to choose their keys first, we can instead require each party to give a proof knowledge of their secret key.

Figure 2: $\mathsf{Forge\text{-}Exp}_{\mathcal{A}}(1^\lambda)$

1. Let T be a variable representing τ.
2. Let A and B be random variables representing the second elements of $\mathbf{A}$ and $\mathbf{B}$ respectively. Such that

$$\mathbf{A} = \begin{bmatrix} 1 \\ A \end{bmatrix}$$
$$\mathbf{B} = \begin{bmatrix} 1 \\ B \end{bmatrix}$$

3. Let $\begin{bmatrix} U_{1,1} & U_{1,2} \\ U_{2,1} & U_{2,2} \end{bmatrix}$ be variables representing corresponding elements of $\mathbf{U}$ and $\begin{bmatrix} V_{1,1} & V_{1,2} \\ V_{2,1} & V_{2,2} \end{bmatrix}$ represent corresponding elements of $\mathbf{V}$.
4. For each $i \in \mathsf{Hon}$, define the following variables

$$\begin{bmatrix} K_{i,1,1} & K_{i,1,2} \\ K_{i,2,1} & K_{i,2,2} \end{bmatrix}$$

   Where each component represents the corresponding element of $\mathbf{K}_i$.
5. **Signing Queries:** Let Q_{Sign} be the number of signing queries the adversary makes. For any signing query $\ell \in [Q_{Sign}]$, let $M_\ell(\mathbf{X})$ represent the message m submitted for signing, and let R_ℓ be a variable that represents the r-value sampled by the signer.

   Therefore, the final list of formal variables is

$$\mathbf{X} = \Big(T, A, B, U_{1,1}, U_{1,2}, U_{2,1}, U_{2,2}, V_{1,1}, V_{1,2}, V_{2,1}, V_{2,2}, \\ (K_{i,1,1}, K_{i,1,2}, K_{i,2,1}, K_{i,2,2})_{i\in\mathsf{Hon}}, (R_\ell)_{\ell\in[Q_{Sign}]}\Big)$$

The Adversary's Inputs $\mathbf{L}_1(\mathbf{X})$ and $\mathbf{L}_2(\mathbf{X})$ The adversary receives input polynomials from the reference string $\mathbf{crs}$, the public keys $(\mathbf{vk}_i)_{i\in\mathsf{Hon}}$, the hints $(\mathbf{hint}_i)_{i\in\mathsf{Hon}}$, and the signing queries. $\mathbf{L}_1(\mathbf{X})$ refers to the input polynomials in $\mathbb{G}_1$, and $\mathbf{L}_2(\mathbf{X})$ refers to the input polynomials in $\mathbb{G}_2$. We list these polynomials in Fig. 3 and Fig. 4 below.

Note that the adversary can make signing queries on any message in $\mathbb{G}_1$, and then the message is multiplied by $\mathbf{K}_i$ as follows:

$$\sigma_{i,1} = \begin{pmatrix} [1]_1 & m \end{pmatrix} \cdot \mathbf{K}_i + [r_i \mathbf{B}^\top (\mathbf{U} + h\mathbf{V})]_1$$

In this way, the adversary can obtain polynomials in $\mathbf{L}_1(\mathbf{X})$ in which any polynomial of their choice is multiplied by $\mathbf{K}_i$. However, this is the only "multiplication" of variables in $\mathbf{L}_1(\mathbf{X})$ that they can perform. For example, they cannot obtain a polynomial in $\mathbf{L}_1(\mathbf{X})$ in which T is multiplied by $V_{1,1}$.

For notational convenience, we work in a hybrid where $\mathbf{hint}_i = (\mathbf{vk}_i \cdot \tau^n)_{n\in[N]}$. Given these values, along with $\mathbf{vk}_i$, the adversary can compute the actual $\mathbf{hint}_i$, so any adversary for the actual security game can operate in this hybrid with the same success probability. This hybrid is referred to later on as Hybrid_1.

Source	Element(s) provided	Polynomials Revealed
crs	$[\mathbf{B}]_1$	B
crs	$[\mathbf{B}^\top \mathbf{U}]_1$	$U_{1,1} + U_{2,1}B, U_{1,2} + U_{2,2}B$
crs	$[\mathbf{B}^\top \mathbf{V}]_1$	$V_{1,1} + V_{2,1}B, V_{1,2} + V_{2,2}B$
crs	$([\tau^n]_1)_{n\in[N]}$	$(T^n)_{n\in[N]}$
Signing Queries $(M_\ell(\mathbf{X}), \Sigma_\ell)_{\ell\in Q_{Sign}}$	$(\sigma_{\ell,1})_{\ell\in Q_{Sign}}$	$K_{i,1,1} + M_\ell(\mathbf{X})K_{i,2,1}$ $+ R_\ell(U_{1,1} + h_\ell V_{1,1}$ $+ B(U_{2,1} + h_\ell V_{2,1}))$ $K_{i,1,2} + M_\ell(\mathbf{X})K_{i,2,2}+$ $R_\ell(U_{1,2} + h_\ell V_{1,2}$ $+ B(U_{2,2} + h_\ell V_{2,2}))$
Signing Queries $(M_\ell(\mathbf{X}), \Sigma_\ell)_{\ell\in Q_{Sign}}$	$(\sigma_{\ell,2})_{\ell\in Q_{Sign}}$	$(R_\ell)_{\ell\in Q_{Sign}}, (R_\ell B)_{\ell\in Q_{Sign}}$

Figure 3: The polynomials in $\mathbf{L}_1(\mathbf{X})$

Source	Element(s) provided	Polynomials Revealed
crs	$[\mathbf{A}]_2$	A
crs	$[\mathbf{UA}]_2$	$U_{1,1} + U_{1,2}A$, $U_{2,1} + U_{2,2}A$
crs	$[\mathbf{VA}]_2$	$V_{1,1} + V_{1,2}A$, $V_{2,1} + V_{2,2}A$
crs	$([\mathbf{A}\tau^n]_2)_{n\in[N]}$	$(T^n)_{n\in[N]}$, $(AT^n)_{n\in[N]}$
$(\mathbf{vk}_i)_{i\in\mathsf{Hon}}$	$[\mathbf{K}_i\mathbf{A}]_2$	$\forall i \in \mathsf{Hon}$, $K_{i,1,1} + K_{i,1,2}A$, $K_{i,2,1} + K_{i,2,2}A$
Hints	$(\mathbf{hint}_i)_{i\in\mathsf{Hon}}$	$\forall i \in \mathsf{Hon}, \forall n \in [N]$, $(K_{i,1,1} + K_{i,1,2}A)T^n$, $(K_{i,2,1} + K_{i,2,2}A)T^n$

Figure 4: The polynomials in $\mathbf{L}_2(\mathbf{X})$

The Adversary's Outputs

1. For the forgery, the adversary outputs a message $M^*(\mathbf{X})$ and a signature:

$$\sigma^*(\mathbf{X}) = \Big(\hat{\sigma}_1(\mathbf{X}), \hat{\sigma}_2(\mathbf{X}), \hat{\sigma}_3(\mathbf{X}), \hat{\sigma}_4(\mathbf{X}), \mathbf{aVK}(\mathbf{X}), \hat{\mathbf{aVK}}(\mathbf{X}),$$
$$B(\mathbf{X}), \hat{B}(\mathbf{X}), Q_0(\mathbf{X}), \mathbf{Q_Z}(\mathbf{X}), \mathbf{Q_x}(\mathbf{X}), \hat{\mathbf{Q}}_\mathbf{x}(\mathbf{X})\Big)$$

2. Let us classify the adversary's outputs into elements of $\mathbb{G}_1$ and $\mathbb{G}_2$. The following polynomials represent elements of $\mathbb{G}_1$ and are dependent on $\mathbf{L}_1(\mathbf{X})$:

$$\hat{\sigma}_1, \hat{\sigma}_2, \hat{\sigma}_3, B,$$

The following polynomials represent elements of $\mathbb{G}_2$ and are dependent on $\mathbf{L}_2(\mathbf{X})$:

$$\hat{\sigma}_4, (\mathbf{K}_i\mathbf{A})_{i\in\mathsf{Cor}}, (M_j(\mathbf{X}))_{j\in[Q_{Sign}]}, M^*(\mathbf{X}), \hat{B}(\mathbf{X}), Q_0(\mathbf{X}), \mathbf{aVK}, \mathbf{Q_Z}, \mathbf{Q_x}, \hat{\mathbf{Q}}_\mathbf{x}$$

The keys are well-formed The following lemma says that if a corrupted party's $\mathbf{vk}_i$ is accepted by $\mathsf{IsValid}$, then the key is well-formed, and the secret ket $\mathbf{K}_i$ can be extracted from $\mathbf{vk}_i$ in the GGM.

Lemma B.1. For each $i \in \mathsf{Cor} \cap V$, $\mathbf{vk}_i(\mathbf{X}) = \mathbf{K}_i\mathbf{A}$ for some scalar matrix $\mathbf{K}_i \in \mathbb{Z}_p^{2\times 2}$.

Proof.

1. $\mathbf{vk}_i(\mathbf{X})$ is a function of A alone.

   The corrupted parties' $\mathbf{vk}_i$'s cannot vary with the honest parties' keys $(\mathbf{K}_j)_{j\in\mathsf{Hon}}$ or with any R_ℓ because the corrupted parties must output their $\mathbf{vk}_i$-values before seeing any polynomials with those variables.

   Additionally, $\mathbf{vk}_i$ does not vary with T, $\mathbf{U}$, or $\mathbf{V}$. $\mathbf{vk}_i$ accepted by $\mathsf{IsValid}(\mathbf{crs}, \mathbf{hint}_i, \mathbf{vk}_i, i, N)$, which checks (among other conditions) that:

   $$\mathbf{h}_3(\mathbf{X}) = T^N \cdot \mathbf{vk}_i(\mathbf{X})$$

   If $\mathbf{vk}_i(\mathbf{X})$ varied with T, $\mathbf{U}$, or $\mathbf{V}$, then one component of $\mathbf{h}_3(\mathbf{X})$ would include a term with a factor of T^{N+1}, $U_{a,b}T^N$, or $V_{a,b}T^N$ for some $(a,b) \in [2]\times[2]$. This is impossible because $\mathbf{h}_3(\mathbf{X})$ is a vector that is dependent on $\mathbf{L}_2(\mathbf{X})$, and no polynomial in $\mathbf{L}_2(\mathbf{X})$ includes such terms.

   The corrupted parties' $\mathbf{vk}_i$'s cannot vary with B because each $\mathbf{vk}_i$ is dependent on $\mathbf{L}_2(\mathbf{X})$, and no polynomials in $\mathbf{L}_2(\mathbf{X})$ vary with B.

   In total, we've shown that $\mathbf{vk}_i(\mathbf{X})$ does not vary with any variable in $\mathbf{X}$, except possibly with A.

2. $\mathbf{vk}_i(\mathbf{X})$ can be expressed as

   $$\mathbf{vk}_i(\mathbf{X}) = \begin{bmatrix} K_{i,1,1} + K_{i,1,2}A \\ K_{i,2,1} + K_{i,2,2}A \end{bmatrix}$$

   for some $\mathbf{K}_i = \begin{bmatrix} K_{i,1,1} & K_{i,1,2} \\ K_{i,2,1} & K_{i,2,2} \end{bmatrix} \in \mathbb{Z}_p^{2\times 2}$. This is because $\mathbf{vk}_i(\mathbf{X})$ is a pair of polynomials that are each a function of A alone. Furthermore, the degree of each polynomial in A is ≤ 1 because $\mathbf{vk}_i(\mathbf{X})$ is dependent on $\mathbf{L}_2(\mathbf{X})$, and every polynomial in $\mathbf{L}_2(\mathbf{X})$ has degree in A that is ≤ 1.

   Finally, this shows that $\mathbf{vk}_i(\mathbf{X}) = \mathbf{K}_i\mathbf{A}$.

Aggregation is correct

Lemma B.2. For any polynomial $P(\mathbf{X})$ that is dependent on $\mathbf{L}_2(\mathbf{X})$, if $P(\mathbf{X})$ varies with $\mathbf{U}, \mathbf{V},$ or $(\mathbf{K}_i)_{i\in\mathsf{Hon}}$, then $P(\mathbf{X})$ also varies with A.

Proof.

1. If $P(\mathbf{X})$ varies with $\mathbf{K}_i$ for some $i \in \mathsf{Hon}$, then $P(\mathbf{X})$ must also vary with A.

   For every polynomial in $\mathbf{L}_2(\mathbf{X})$ that varies with $\mathbf{K}_i$, $\mathbf{K}_i$ appears in terms of the form:

   $$(K_{i,1,1} + K_{i,1,2} \cdot A) \cdot Q(T) \tag{3}$$
   $$\text{or} \quad (K_{i,2,1} + K_{i,2,2} \cdot A) \cdot Q(T) \tag{4}$$

   where $Q(T)$ is some polynomial. Then the terms of $P(\mathbf{X})$ that vary with $\mathbf{K}_i$ must have the form:

   $$\begin{aligned} &c_1 \cdot (K_{i,1,1} + K_{i,1,2}A) + c_2 \cdot (K_{i,2,1} + K_{i,2,2}A) \\ &= c_1K_{i,1,1} + c_1K_{i,1,2}A + c_2K_{i,2,1} + c_2K_{i,2,2}A \end{aligned} \tag{5}$$

   where (c_1, c_2) are constants, and at least one of them is non-zero. Equation (5) has 4 terms, and they do not cancel out with each other. Since at least one of (c_1, c_2) is non-zero, there is a term of $P(\mathbf{X})$ that varies with A.

2. By a similar argument, we can show that if $P(\mathbf{X})$ varies with $\mathbf{U}$ or $\mathbf{V}$, then $P(\mathbf{X})$ varies with A as well.

Lemma B.3. For some scalars $(b_1, \ldots, b_N)$,

$$B(\mathbf{X}) = \sum_{i \in [N]} b_i \cdot L_i(T)$$

Furthermore, for at least t values of $i \in [N]$, $b_i \neq 0$.

Proof.

1. $B(\mathbf{X})$ is a function of T alone.

   AggrVerify checks that

   $$B(\mathbf{X}) \cdot T^t = \hat{B}(\mathbf{X})$$

   Therefore

   $$B(\mathbf{X}) = T^{-t} \cdot \hat{B}(\mathbf{X})$$

   $\hat{B}(\mathbf{X})$ is dependent on $\mathbf{L}_2(\mathbf{X})$, and T^t is a factor of $\hat{B}(\mathbf{X})$. Every polynomial in $\mathbf{L}_2(\mathbf{X})$ either does not vary with T, or it has T as a factor. Therefore, $\hat{B}(\mathbf{X})$ is a linear combination of only the polynomials in $\mathbf{L}_2(\mathbf{X})$ that have T as a factor.

   The polynomials in $\mathbf{L}_2(\mathbf{X})$ that have T as a factor only vary with the following variables: T, A, $(\mathbf{K}_i)_{i \in \mathsf{Hon}}$.

   $B(\mathbf{X})$ is dependent on $\mathbf{L}_1(\mathbf{X})$, so it does not vary with A. Then $\hat{B}(\mathbf{X})$ does not vary with A either.

   $\hat{B}(\mathbf{X})$ does not vary with $\mathbf{K}_i$, for any $i \in \mathsf{Hon}$, because if it did, then it would vary with A Lemma B.2.

   In summary, $\hat{B}(\mathbf{X})$ is a function of T alone, so is $B(\mathbf{X})$ as well.

2. $\deg(B) \leq N - t \leq N - 1$.

   $\hat{B}(\mathbf{X})$ is dependent on $L_2(\mathbf{X})$, so its degree in T is $\leq N$. Since $B(T) = T^{-t} \cdot \hat{B}(\mathbf{X})$, then $\deg(B) \leq N - t$. Furthermore, since $t \geq 1$ for any successful forgery, $N - t \leq N - 1$.

3. Since $B(T)$ is a polynomial in T of degree $\leq N-1$, we can express B as its Lagrange interpolation. For some scalars, $(b_1, \ldots, b_N)$,

   $$B(T) = \sum_{i \in [N]} b_i \cdot L_i(T)$$

4. $B(0) = 1$.

   AggrVerify checks that

   $$B(T) = T \cdot Q_0(\mathbf{X}) + 1$$

   This polynomial identity is true for all values of T, so let us evaluate it at $T = 0$.

   $$\begin{aligned} B(0) &= 0 \cdot Q_0(\mathbf{X})|_{T=0} + 1 \\ &= 1 \end{aligned}$$

5. For at least t values of $i \in [N]$, $b_i \neq 0$.

   $B(T)$ is a non-zero polynomial of degree $\leq N - t$, so it has at most $N - t$ roots. Then for at least t values of $i \in [N]$, $B(\omega^i) = b_i \neq 0$.

Lemma B.4. There is at least one $i^* \in \mathsf{Hon}$ such that $b_{i^*} \neq 0$, and i^* was not asked to sign $M^*(\mathbf{X})$ during any signing query.

Proof.

1. Recall that Q is the set of honest parties that were queried on message $M^*(\mathbf{X})$ during some signing query.

   We have assumed that the adversary wins the forgery game, so $t > |Q \cup \mathsf{Cor}|$. Then any subset of $[N]$ of size t must include at least one element of $[N] \backslash (Q \cup \mathsf{Cor})$.

2. Note that the set $[N] \backslash (Q \cup \mathsf{Cor}) = ([N] \backslash \mathsf{Cor}) \backslash Q = \mathsf{Hon} \backslash Q$. So $[N] \backslash (Q \cup \mathsf{Cor})$ refers to all honest parties that were *not* queried on $M^*(\mathbf{X})$.

3. There are at least t values of $i \in [N]$ for which $b_i \neq 0$ (Lemma B.3). Then there is at least one $i^* \in \mathsf{Hon}$ such that i^* was not asked to sign $M^*(\mathbf{X})$ during any call to PartialSign, and $b_{i^*} \neq 0$.

Lemma B.5. The degree in T of $\mathbf{Q}_x(\mathbf{X})$ is $\leq N - 2$.

Proof. AggrVerify checks that:

$$T^2 \cdot \mathbf{Q_x}(\mathbf{X}) = \hat{\mathbf{Q}}_\mathbf{x}(\mathbf{X})$$

Which implies that:

$$\mathbf{Q_x}(\mathbf{X}) = T^{-2} \cdot \hat{\mathbf{Q}}_\mathbf{x}(\mathbf{X})$$

$\hat{\mathbf{Q}}_\mathbf{x}(\mathbf{X})$ depends on $\mathbf{L}_2(\mathbf{X})$. The degree in T of every polynomial in $\mathbf{L}_2(\mathbf{X})$ is $\leq N$, so the degree in T of $\hat{\mathbf{Q}}_\mathbf{x}(\mathbf{X})$ is $\leq N$. Then the degree in T of $\mathbf{Q_x}(\mathbf{X})$ is $\leq N - 2$.

Lemma B.6.

$$\mathbf{aVK}(\mathbf{X}) = \frac{1}{N} \cdot \sum_{i \in V} b_i \cdot \mathbf{vk}_i(\mathbf{X})$$

Proof.

1. Let us define the polynomials $\mathbf{A}, B, \tilde{Q}_{x,1}$ and $\tilde{Q}_{Z,1}$ that are required for the univariate sumcheck lemma (Lemma 2.1).

   First, we will define $\mathbf{A}(\mathbf{X})$ to be the same as $\mathbf{VK}(\mathbf{X})$, except that the polynomials $\mathbf{vk}_i(\mathbf{X})$ are incorporated into $(a_1, \ldots, a_N)$. For all $i \in V$, let $a_i = \mathbf{vk}_i(\mathbf{X})$, and for all $i \in [N] \backslash V$, let $a_i = \mathbf{0}$. Next, let $\mathbf{A}(\mathbf{X}) = \sum_{i \in [N]} a_i \cdot L_i(T)$. Note that

   $$\mathbf{A}(\mathbf{X}) = \sum_{i \in V} \mathbf{vk}_i(\mathbf{X}) \cdot L_i(T) = \mathbf{VK}(\mathbf{X})$$

   Next, define $B(T)$ as in Lemma B.3: $B(T) = \sum_{i \in [N]} b_i \cdot L_i(T)$. Also, let $\tilde{\mathbf{Q}}_\mathbf{x}$ and $\tilde{\mathbf{Q}}_\mathbf{Z}$ be defined as $\tilde{Q}_x$ and $\tilde{Q}_Z$ were defined in Lemma 2.1:

$$\begin{aligned}
\tilde{\mathbf{Q}}_{\mathbf{x}}(\mathbf{X}) &= \sum_{i\in[N]} a_i \cdot b_i \cdot \frac{L_i(T) - L_i(0)}{T} \\
&= \frac{L_0(T) - L_0(0)}{T} + \sum_{i\in V} \mathbf{vk}_i(\mathbf{X}) \cdot b_i \cdot \frac{L_i(T) - L_i(0)}{T} \\
\tilde{\mathbf{Q}}_{\mathbf{Z}}(\mathbf{X}) &= \sum_{i\in[N]} a_i \cdot b_i \cdot \frac{L_i^2(T) - L_i(T)}{Z(T)} + \sum_{i,j\in[N]:i\neq j} a_i \cdot b_j \cdot \frac{L_i(T) \cdot L_j(T)}{Z(T)} \\
&= \frac{L_0^2(T) - L_0(T)}{Z(T)} + \sum_{i\in V} \mathbf{vk}_i(\mathbf{X}) \cdot b_i \cdot \frac{L_i^2(T) - L_i(T)}{Z(T)} \\
&\quad + \sum_{j\in[N]} b_j \cdot \frac{L_0(T) \cdot L_j(T)}{Z(T)} + \sum_{i\in V} \sum_{j\in[N]\setminus\{i\}} \mathbf{vk}_i(\mathbf{X}) \cdot b_j \cdot \frac{L_i(T) \cdot L_j(T)}{Z(T)}
\end{aligned}$$

Note that $\tilde{\mathbf{Q}}_{\mathbf{x}}$ and $\tilde{\mathbf{Q}}_{\mathbf{Z}}$ are dependent on $\mathbf{L}_2(\mathbf{X})$ because they can be computed using the hints.

2. The degree in T of $\tilde{\mathbf{Q}}_{\mathbf{x}}(\mathbf{X})$ and $\tilde{\mathbf{Q}}_{\mathbf{Z}}$ are each $\leq N-2$. This follows from the following facts:
   (a) For each $i \in V$, the degree in T of $\mathbf{vk}_i(\mathbf{X})$ is 0 (Lemma B.1).
   (b) For each $i \in [N]$, $L_i(T)$ has degree $N-1$.
   (c) $Z(T)$ has degree N.

3. The univariate sumcheck lemma (Lemma 2.1) implies that:

$$\begin{aligned}
\mathbf{A}(\mathbf{X}) \cdot B(T) &= \frac{1}{N} \cdot \left(\sum_{i\in[N]} a_i \cdot b_i \right) + \tilde{\mathbf{Q}}_{\mathbf{x}}(\mathbf{X}) \cdot T + \tilde{\mathbf{Q}}_{\mathbf{Z}}(\mathbf{X}) \cdot Z(T) \\
\mathbf{VK}(\mathbf{X}) \cdot B(T) &= \frac{1}{N} \cdot \left(\sum_{i\in V} \mathbf{vk}_i(\mathbf{X}) \cdot b_i \right) + \tilde{\mathbf{Q}}_{\mathbf{x}}(\mathbf{X}) \cdot T + \tilde{\mathbf{Q}}_{\mathbf{Z}}(\mathbf{X}) \cdot Z(T)
\end{aligned}$$

Furthermore, AggrVerify checks that:

$$\mathbf{VK}(\mathbf{X}) \cdot B(T) = \mathbf{aVK}(\mathbf{X}) + \mathbf{Q}_{\mathbf{x}}(\mathbf{X}) \cdot T + \mathbf{Q}_{\mathbf{Z}}(\mathbf{X}) \cdot Z(T)$$

Then combining the equations above, we have that:

$$\begin{aligned}
\mathbf{VK}(\mathbf{X}) \cdot B(T) &= \mathbf{aVK}(\mathbf{X}) + \mathbf{Q}_{\mathbf{x}}(\mathbf{X}) \cdot T + \mathbf{Q}_{\mathbf{Z}}(\mathbf{X}) \cdot Z(T) \\
&= \frac{1}{N} \cdot \left(\sum_{i\in V} \mathbf{vk}_i(\mathbf{X}) \cdot b_i \right) + \tilde{\mathbf{Q}}_{\mathbf{x}}(\mathbf{X}) \cdot T + \tilde{\mathbf{Q}}_{\mathbf{Z}}(\mathbf{X}) \cdot Z(T) \\
\mathbf{aVK}(\mathbf{X}) &= \frac{1}{N} \cdot \left(\sum_{i\in V} \mathbf{vk}_i(\mathbf{X}) \cdot b_i \right) + \left(\tilde{\mathbf{Q}}_{\mathbf{x}}(\mathbf{X}) - \mathbf{Q}_{\mathbf{x}}(\mathbf{X}) \right) \cdot T + \left(\tilde{\mathbf{Q}}_{\mathbf{Z}}(\mathbf{X}) - \mathbf{Q}_{\mathbf{Z}}(\mathbf{X}) \right) \cdot Z(T)
\end{aligned}$$

4. Let us define polynomials F_1, F_2, G_1, G_2 to simplify the notation.

$$\begin{aligned}
\text{Let} \quad F_1(\mathbf{X}) &= \left(\tilde{\mathbf{Q}}_{\mathbf{x}}(\mathbf{X}) - \mathbf{Q}_{\mathbf{x}}(\mathbf{X}) \right)_1 \\
F_2(\mathbf{X}) &= \left(\tilde{\mathbf{Q}}_{\mathbf{x}}(\mathbf{X}) - \mathbf{Q}_{\mathbf{x}}(\mathbf{X}) \right)_2 \\
G_1(\mathbf{X}) &= \left(\tilde{\mathbf{Q}}_{\mathbf{Z}}(\mathbf{X}) - \mathbf{Q}_{\mathbf{Z}}(\mathbf{X}) \right)_1 \\
G_2(\mathbf{X}) &= \left(\tilde{\mathbf{Q}}_{\mathbf{Z}}(\mathbf{X}) - \mathbf{Q}_{\mathbf{Z}}(\mathbf{X}) \right)_2
\end{aligned}$$

Note that F_1, F_2, G_1, G_2 are dependent on $\mathbf{L}_2(\mathbf{X})$ because $\mathbf{Q_x}$, $\tilde{\mathbf{Q}}_\mathbf{x}$, $\mathbf{Q_Z}$, $\tilde{\mathbf{Q}}_\mathbf{Z}$ are dependent on $\mathbf{L}_2(\mathbf{X})$. Then for each $a \in \{1, 2\}$,

$$\mathbf{aVK}_a(\mathbf{X}) = \frac{1}{N} \cdot \left(\sum_{i \in V} (\mathbf{vk}_i)_a (\mathbf{X}) \cdot b_i \right) + F_a(\mathbf{X}) \cdot T + G_a(\mathbf{X}) \cdot T^N - G_a(\mathbf{X}) \qquad (6)$$

We also replaced $Z(T)$ with $T^N - 1$.

5. The degree in T of F_1 and F_2 is $\leq N - 2$. This is because the degree in T of $\mathbf{Q_x}(\mathbf{X})$ is $\leq N - 2$ (Lemma B.5), and the degree in T of $\tilde{\mathbf{Q}}_\mathbf{x}(\mathbf{X})$ is $\leq N - 2$ (Lemma 2.1).

6. $G_1(\mathbf{X}) = G_2(\mathbf{X}) = 0$.

   Let us assume toward contradiction that for some $a \in \{1, 2\}$, $G_a(\mathbf{X}) \neq 0$. Then we will show that the degree in T of $\mathbf{a\hat{V}K}(\mathbf{X})$ is $\geq 2N$.

   Let $d = \deg_T(G_a)$, and let $t(\mathbf{X})$ be the non-zero terms of $G_a(\mathbf{X})$ with T^d as a factor. Such terms will exist if $G_a(\mathbf{X}) \neq 0$. Then the expression $G_a(\mathbf{X}) \cdot T^N$ will have non-zero terms with T^{N+d} as a factor.

   Furthermore, these terms will not cancel out on the right-hand side of line 6. The degree in T of $\frac{1}{N} \cdot \left(\sum_{i \in V} (\mathbf{vk}_i)_a (\mathbf{X}) \cdot b_i \right)$ is 0 because the degree in T of each $\mathbf{vk}_i(\mathbf{X})$ is 0 (Lemma B.1). The degree in T of $F_a(\mathbf{X}) \cdot T$ is $\leq N - 1$. Finally, the degree in T of $-G_a(\mathbf{X})$ is d. None of these polynomials have any terms of degree $N + d$ in T.

   Therefore, $\mathbf{aVK}(\mathbf{X})$ will include non-zero terms of degree $N + d$ for some $d \geq 0$.

   Next, since

   $$\mathbf{a\hat{V}K}(\mathbf{X}) = \mathbf{aVK}(\mathbf{X}) \cdot T^N$$

   the degree in T of $\mathbf{a\hat{V}K}(\mathbf{X})$ will be $\geq 2N$. This is impossible because $\mathbf{a\hat{V}K}(\mathbf{X})$ is dependent on $\mathbf{L}_2(\mathbf{X})$. We've reached a contradiction, so in fact, $G_a(\mathbf{X}) = 0$ for both $a \in [2]$.

7. $F_1(\mathbf{X}) = F_2(\mathbf{X}) = 0$.

   Assume toward contradiction that $F_a(\mathbf{X}) \neq 0$ for some $a \in [2]$. Then we will show that the degree in T of $\mathbf{a\hat{V}K}(\mathbf{X})$ is $\geq N + 1$.

   If $F_a(\mathbf{X}) \neq 0$, then $F_a(\mathbf{X}) \cdot T$ includes a non-zero term whose degree in T is ≥ 1. This term will not cancel out on the right-hand side of line 6. The degree in T of $\frac{1}{N} \cdot \left(\sum_{i \in V} (\mathbf{vk}_i)_a (\mathbf{X}) \cdot b_i \right)$ is 0, and $G_a(\mathbf{X}) = 0$.

   Then $\mathbf{aVK}_a(\mathbf{X})$ will include a non-zero term whose degree in T is ≥ 1. Since

   $$\mathbf{a\hat{V}K}(\mathbf{X}) = \mathbf{aVK}(\mathbf{X}) \cdot T^N$$

   the degree in T of $\mathbf{a\hat{V}K}(\mathbf{X})$ will be $\geq N + 1$. This is impossible because $\mathbf{a\hat{V}K}(\mathbf{X})$ is dependent on $\mathbf{L}_2(\mathbf{X})$. We've reached a contradiction, so in fact, $F_a(\mathbf{X}) = 0$ for both $a \in [2]$.

8. Finally, since $F_1 = F_2 = G_1 = G_2 = 0$, we have that:

   $$\mathbf{aVK}(\mathbf{X}) = \frac{1}{N} \cdot \sum_{i \in V} b_i \cdot \mathbf{vk}_i(\mathbf{X})$$

Hybrids We proceed using a series of hybrids.

1. Hybrid_0 is Forge-Exp, the unforgeability game for our signature scheme.

2. Hybrid_1 is the same as Hybrid_0, except we change $\mathbf{hint}_i$ from

$$\mathbf{hint}_i = \left([\mathbf{vk}_i \cdot L_i(\tau)]_2, [\mathbf{vk}_i \cdot \tau\,(L_i(\tau) - L_i(0))]_2, \left[\mathbf{vk}_i \cdot \tau^N\right]_2, \left[\mathbf{vk}_i \cdot \frac{L_i^2(\tau) - L_i(\tau)}{Z(\tau)}\right]_2, \left[\mathbf{vk}_i \cdot \frac{L_i(\tau) - L_i(0)}{\tau}\right]_2, \left\{ \left[\mathbf{vk}_i \cdot \frac{L_i(\tau) \cdot L_j(\tau)}{Z(\tau)}\right]_2 \right\}_{j \in [N] \setminus \{i\}} \right)$$

   to

$$\mathbf{hint}_i = (\mathbf{vk}_i \cdot \tau^n)_{n \in [N]}$$

3. Hybrid_2 is the same as Hybrid_1 except when the adversary outputs the forgery (t, m^*, σ^*), they must also output scalars $(b_i)_{i \in [N]}$ such that

$$B(\mathbf{X}) = \sum_{i \in [N]} b_i \cdot L_i(T) \tag{7}$$

   as well as secret keys $(\mathbf{K}_i)_{i \in \mathsf{Cor} \cap V}$ for some of the corrupted parties. Formally, they must output scalar matrices $\mathbf{K}_i \in \mathbb{Z}_p^{2\times 2}$ such that $\mathbf{vk}_i = \mathbf{K}_i \mathbf{A}$ for each $i \in \mathsf{Cor} \cap V$.

4. Hybrid_3 is the same as Hybrid_2 except the challenger, instead of the adversary, computes $\mathbf{aVK}$ as follows:

$$\mathbf{aVK}(\mathbf{X}) = \frac{1}{N} \cdot \sum_{i \in V} b_i \cdot \mathbf{vk}_i(\mathbf{X}) \tag{8}$$

   We also replace the checks of $\mathsf{AggrVerify}$ with the following checks:

   (a) $b_i \neq 0$ for at least t values of $i \in [N]$.

   (b) There exists an i^* s.t. $b_{i^*} \neq 0$ and party i^* was never queried on m^*.

   (c) $\hat{\sigma}_1 \circ [\mathbf{A}]_2 = \begin{pmatrix}[1]_1 & m^*\end{pmatrix} \circ \mathbf{aVK} + \hat{\sigma}_2 \circ [\mathbf{UA}]_2 + \hat{\sigma}_3 \circ [\mathbf{VA}]_2$

   (d) $\hat{\sigma}_2 \circ \hat{\sigma}_4 = \hat{\sigma}_3 \circ [1]_2$

   The first two checks are new, and the last two checks were present in the previous hybrid. Let us call the new function $\mathsf{AggrVerify}'$.

5. Hybrid_4 is the unforgeability game for the basic signature scheme.

   (a) The challenger samples pp, which includes the parameters of the bilinear pairing and the hash function $\mathcal{H}$, as well as the values $\left([\mathbf{A}]_2, [\mathbf{UA}]_2, [\mathbf{VA}]_2, [\mathbf{B}]_1, [\mathbf{B}^T\mathbf{U}]_1, [\mathbf{B}^T\mathbf{V}]_1\right)$ and sends these values to the adversary. The challenger also samples $\mathbf{K} \xleftarrow{\$} \mathbb{Z}_p^{2\times 2}$ and sends $[\mathbf{KA}]_2$ to the adversary.

   (b) The adversary may submit signing queries on messages $m \in \mathbb{G}_1$. The challenger computes $\mathsf{PartialSign}(\mathbf{crs}, \mathbf{K}, m) \to \{\sigma, \perp\}$ and returns the result to the adversary.

   (c) The adversary outputs a forgery (m^*, Σ^*). The challenger checks that m^* was not previously queried, and

$$\mathsf{PartialVerify}(\mathbf{crs}, [\mathbf{KA}]_2, m^*, \Sigma^*) \to 1$$

   If so, then the output of the hybrid is 1. If not, the output is 0.

Lemma B.7. Given any PPT adversary $\mathcal{A}$ for Hybrid_0, there is an adversary $\mathcal{B}$ for Hybrid_1 that wins with the same probability.

Proof. $\mathcal{B}$ uses the Hybrid_1 hints to construct the Hybrid_0 hints. This is possible because in Hybrid_0, $\mathbf{hint}_i$ comprises several polynomials in τ of degree $\leq N$, each multiplied by $\mathbf{vk}_i$.

Then $\mathcal{B}$ gives the Hybrid_0 hints to $\mathcal{A}$, runs $\mathcal{A}$ internally, and outputs whatever $\mathcal{A}$ outputs. $\mathcal{B}$ will win with the same probability as $\mathcal{A}$.

Lemma B.8. Given any adversary $\mathcal{A}$ for Hybrid_1, there is an adversary $\mathcal{B}$ for Hybrid_2 that wins with the same probability.

Proof. $\mathcal{B}$ runs $\mathcal{A}$ internally. When $\mathcal{A}$ outputs the forgery (t, m^*, σ^*), $\mathcal{B}$ verifies the forgery with AggrVerify. If AggrVerify passes verification, then $B(\mathbf{X})$ can be expressed as in Eq. (7) for some scalars $(b_i)_{i\in[N]}$. This follows from Lemma B.3.

Since we are working in the GGM, $\mathcal{B}$ can extract the values of $(b_i)_{i\in[N]}$ from $\mathcal{A}$'s output B. $\mathcal{B}$ can also extract $\mathbf{K}_i$ from $\mathbf{vk}_i$ for each $i \in \mathsf{Cor} \cap V$ due to Lemma B.1. So if AggrVerify accepts, then $\mathcal{B}$ extracts the values $(b_i)_{i\in[N]}$ and $(\mathbf{K}_i)_{i\in\mathsf{Cor}\cap V}$ and outputs them along with (t, m^*, σ^*).

Lemma B.9. Given any adversary $\mathcal{A}$ for Hybrid_2, there is an adversary $\mathcal{B}$ that wins in Hybrid_3 with the same probability or greater.

Proof. $\mathcal{B}$ runs $\mathcal{A}$ internally and outputs everything that $\mathcal{A}$ outputs, except not $\mathbf{aVK}$. It suffices to show that if $\mathcal{A}$'s output would win in Hybrid_2, then $\mathcal{B}$'s output will win in Hybrid_3.

In Hybrid_2, if AggrVerify passes verification, then $\mathbf{aVK}$ is given by the challenger's formula, Eq. (8). This follows from Lemma B.6. $B(\mathbf{X})$ can be expressed as in Eq. (7) for some scalars $(b_i)_{i\in[N]}$ that satisfy the conditions of AggrVerify′. This follows from Lemma B.3 and Lemma B.4.

Lemma B.10. Given an adversary $\mathcal{A}$ that wins the Hybrid_3 with non-negligible probability, there is an adversary $\mathcal{B}$ that wins Hybrid_4 with non-negligible probability.

Proof.
Construction of $\mathcal{B}$:

1. **Setup:**

   (a) $\mathcal{B}$ receives 1^λ, as well as

   $$\mathsf{pp} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e, g, \hat{g}, \mathcal{H})$$
   $$\mathbf{crs} = \left([\mathbf{A}]_2, [\mathbf{UA}]_2, [\mathbf{VA}]_2, [\mathbf{B}]_1, [\mathbf{B}^\top\mathbf{U}]_1, [\mathbf{B}^\top\mathbf{V}]_1\right)$$

   from the challenger.

   (b) $\mathcal{B}$ sends 1^λ to $\mathcal{A}$. $\mathcal{A}$ picks $N \in \mathbb{N}$.

   (c) $\mathcal{B}$ samples $\tau \xleftarrow{\$} \mathbb{Z}_p$ and sets

   $$\mathsf{pp}' = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e, g, \hat{g}, \mathcal{H})$$
   $$\mathbf{crs}' = \Bigg(\left([\mathbf{UA}]_2, [\mathbf{VA}]_2, [\mathbf{B}]_1, [\mathbf{B}^\top\mathbf{U}]_1, [\mathbf{B}^\top\mathbf{V}]_1\right), \left([1]_1, [\tau]_1, \ldots, \left[\tau^N\right]_1\right), \left([\mathbf{A}]_2, [\mathbf{A}\cdot\tau]_2, \ldots, \left[\mathbf{A}\cdot\tau^N\right]_2\right) \Bigg)$$

   and sends $(\mathsf{pp}', \mathbf{crs}')$ to $\mathcal{A}$.

   (d) $\mathcal{A}$ picks $\mathsf{Cor} \subset [N]$, the set of corrupted parties. Let $\mathsf{Hon} = [N]\backslash\mathsf{Cor}$.

   (e) $\mathcal{B}$ picks a random $i^* \in \mathsf{Hon}$. $\mathcal{B}$ sets i^* to be the honest signer from Hybrid_4.

2. **Key Generation:**

   (a) For i^*, $\mathcal{B}$ receives $\mathbf{vk}_{i^*} = [\mathbf{KA}]_2$ from $\mathcal{C}$. Since $\mathcal{B}$ knows τ, it computes

$$\mathbf{hint}_{i^*} = \Bigg([\mathbf{KA} \cdot L_{i^*}(\tau)]_2, [\mathbf{KA} \cdot \tau \left(L_{i^*}(\tau) - L_{i^*}(0)\right)]_2, \left[\mathbf{KA} \cdot \tau^N\right]_2, \left[\mathbf{KA} \cdot \frac{L_{i^*}^2(\tau) - L_{i^*}(\tau)}{Z(\tau)}\right]_2, \left[\mathbf{KA} \cdot \frac{L_{i^*}(\tau) - L_{i^*}(0)}{\tau}\right]_2, \left\{ \left[\mathbf{KA} \cdot \frac{L_{i^*}(\tau) \cdot L_j(\tau)}{Z(\tau)}\right]_2 \right\}_{j \in [N] \setminus \{i^*\}} \Bigg)$$

   (b) For each $i \in \mathsf{Hon} \setminus \{i^*\}$, $\mathcal{B}$ computes:

$$\mathsf{KGen}(\mathbf{crs}') \to (\mathbf{sk}_i, \mathbf{vk}_i)$$
$$\mathsf{HintGen}(\mathbf{crs}', \mathsf{sk}_i, i, N) \to \mathbf{hint}_i$$

   (c) For each $i \in \mathsf{Cor}$, the $\mathcal{A}$ outputs $(\mathbf{vk}_i, \mathbf{hint}_i)$.

   (d) For each $i \in \mathsf{Hon}$, $\mathcal{B}$ sends $(\mathbf{vk}_i, \mathbf{hint}_i)$ to $\mathcal{A}$.

   (e) $\mathcal{B}$ computes $\mathsf{Preprocess}(\mathbf{crs}, (\mathbf{hint}_i, \mathbf{vk}_i)_{i \in [N]}) \to (\mathbf{vk}, \mathbf{ak})$ and sends them to $\mathcal{A}$. Let V be the set of keys for which $\mathsf{IsValid}$ returns 1.

3. **Signing Queries:** $\mathcal{A}$ may query $\mathsf{PartialSign}$ on inputs $(i, m) \in \mathsf{Hon} \times \mathbb{G}_1$. The query's *message* is m, and its *signing party* is i. $\mathcal{B}$ adds (i, m) to Q_{Sign}.

   (a) If $i = i^*$, $\mathcal{B}$ queries $\mathcal{C}$ to get $\mathsf{PartialSign}(\mathbf{crs}', \mathbf{sk}_{i^*}, m) \to \{\sigma, \perp\}$

   (b) Else, $\mathcal{B}$ computes $\mathsf{PartialSign}(\mathbf{crs}', \mathbf{sk}_i, m) \to \{\sigma, \perp\}$

   $\mathcal{B}$ sends the result to the adversary $\mathcal{A}$. $\mathcal{A}$ can make any number of such signing queries.

4. **Forgery:**

(a) $\mathcal{A}$ outputs a forgery $(t, m^*, \sigma^*, (b_i)_{i \in [N]}, (\mathbf{K}_j)_{j \in \mathsf{Cor}})$.

(b) $\mathcal{B}$ aborts if $(i^*, m^*) \in Q_{Sign}$ or if $b_{i^*} = 0$. Otherwise, $\mathcal{B}$ continues.

(c) Parse $\sigma^* = \left(\hat{\sigma}_1, \hat{\sigma}_2, \hat{\sigma}_3, \hat{\sigma}_4, \mathbf{aVK}, \mathbf{a\hat{V}K}, B, \hat{B}, Q_0, \mathbf{Q_Z}, \mathbf{Q_x}, \mathbf{\hat{Q}_x}\right)$
Furthermore, let $S_v = S \cap V$.

(d) $\mathcal{B}$ returns the forgery $(m^*, \Sigma^* = (\hat{\sigma}_1^*, \hat{\sigma}_2^*, \hat{\sigma}_3^*, \hat{\sigma}_4^*))$, where

$$\hat{\sigma}_1^* = \frac{N}{b_{i^*}} \left(\hat{\sigma}_1 - \sum_{i \in S_v, i \neq i^*} \begin{pmatrix} [1]_1 & m^* \end{pmatrix} \cdot \frac{b_i}{N} \cdot \mathbf{K}_i \right)$$
$$\sigma_2^* = \frac{N}{b_{i^*}} (\hat{\sigma}_2)$$
$$\sigma_3^* = \frac{N}{b_{i^*}} (\hat{\sigma}_3)$$
$$\sigma_4^* = \hat{\sigma}_4$$

Analysis: $\mathcal{B}$ acts as an adversary for Hybrid_4 and exactly simulates the Hybrid_3 game for $\mathcal{A}$.

1. If $\mathcal{A}$ wins Hybrid_3, then $\mathcal{B}$ has access to $(b_i)_{i \in [N]}$ such that $b_i \neq 0$ for at least t i-values. This trivially implies Lemma B.3. Then from Lemma B.4, we know $\exists i' \in \mathsf{Hon}$ s.t. $b_{i'} \neq 0$ and $(i', m^*) \notin Q_{Sign}$.

2. $\mathcal{B}$ aborts if $i' \neq i^*$. Since i^* is picked randomly and it is independent of $\mathcal{A}$'s view, the probability of not aborting is $\frac{1}{|\mathsf{Hon}|}$. Then, we have

$$\Pr\left[i' = i^* | \mathsf{Hybrid}_{3\mathcal{A}}(1^\lambda) = 1\right] = \frac{1}{|\mathsf{Hon}|} \tag{9}$$

From now, let us focus on the case where we do not abort.

3. We will show that $\mathsf{PartialVerify}(\mathbf{crs}, [\mathbf{KA}]_2, m^*, \Sigma^*) = 1$ for the forgery $(m^*, \Sigma^* = (\hat{\sigma}_1^*, \hat{\sigma}_2^*, \hat{\sigma}_3^*, \hat{\sigma}_4^*))$. We need to show the following equations hold
   (a) $\hat{\sigma}_1^* \circ [\mathbf{A}]_2 = \begin{pmatrix}[1]_1 & m^*\end{pmatrix} \circ [\mathbf{KA}]_2 + \hat{\sigma}_2^* \circ [\mathbf{UA}]_2 + \hat{\sigma}_3^* \circ [\mathbf{VA}]_2$
   (b) $\hat{\sigma}_2^* \circ \hat{\sigma}_4^* = \hat{\sigma}_3^* \circ [1]_2$

4. Since the forgery given by $\mathcal{A}$ wins Hybrid_3, we have

$$\hat{\sigma}_2 \circ \hat{\sigma}_4 = \hat{\sigma}_3 \circ [1]_2$$

Since $\hat{\sigma}_2^* = \frac{N}{b_{i^*}}(\hat{\sigma}_2), \hat{\sigma}_3^* = \frac{N}{b_{i^*}}(\hat{\sigma}_3), \hat{\sigma}_4^* = \hat{\sigma}_4$, We can multiply both sides by $\frac{N}{b_{i^*}}$ to get

$$\left(\frac{N}{b_{i^*}} \cdot \hat{\sigma}_2\right) \circ \hat{\sigma}_4 = \left(\frac{N}{b_{i^*}} \cdot \hat{\sigma}_3\right) \circ [1]_2$$
$$\hat{\sigma}_2^* \circ \hat{\sigma}_4^* = \hat{\sigma}_3^* \circ [1]_2$$

5. Since the forgery given by $\mathcal{A}$ wins the game, we have

$$\hat{\sigma}_1 \circ [\mathbf{A}]_2 = \begin{pmatrix}[1]_1 & m^*\end{pmatrix} \circ \mathbf{aVK} + \hat{\sigma}_2 \circ [\mathbf{UA}]_2 + \hat{\sigma}_3 \circ [\mathbf{VA}]_2 \tag{10}$$

Let $\mathbf{K}' = \frac{1}{N}\sum_{i \in S_v, i \neq i^*} b_i \mathbf{K}_i$. Then by bilinearity

$$-\begin{pmatrix}[1]_1 & m^*\end{pmatrix} \mathbf{K}' \circ [\mathbf{A}]_2 = \begin{pmatrix}[1]_1 & m^*\end{pmatrix} \circ [-\mathbf{K}'\mathbf{A}]_2 \tag{11}$$

Adding Eq. (11) to Eq. (10),

$$\left(\hat{\sigma}_1 - \begin{pmatrix}[1]_1 m^*\end{pmatrix} \cdot \mathbf{K}'\right) \circ [\mathbf{A}]_2 = \begin{pmatrix}[1]_1 & m^*\end{pmatrix} \circ (\mathbf{aVK} + [-\mathbf{K}'\mathbf{A}]_2) + \hat{\sigma}_2 \circ [\mathbf{UA}]_2 + \hat{\sigma}_3 \circ [\mathbf{VA}]_2$$

Multiplying both sides with $\frac{N}{b_{i^*}}$ we get

$$\hat{\sigma}_1^* \circ [\mathbf{A}]_2 = \begin{pmatrix}[1]_1 m^*\end{pmatrix} \circ \left(\frac{N}{b_{i^*}} \cdot (\mathbf{aVK} + [-\mathbf{K}'\mathbf{A}]_2)\right) + \hat{\sigma}_2^* \circ [\mathbf{UA}]_2 + \hat{\sigma}_3^* \circ [\mathbf{VA}]_2 \tag{12}$$

We know

$$\mathbf{aVK} = \frac{1}{N} \cdot \sum_{i \in S_v} b_i \cdot \mathbf{vk}_i$$

Then we have

$$\mathbf{aVK} + [-\mathbf{K}'\mathbf{A}]_2 = \left(\frac{1}{N} \cdot \sum_{i \in S_v} b_i \cdot \mathbf{vk}_i\right) - \left(\left[\frac{1}{N} \sum_{i \in S_v, i \neq i^*} b_i \mathbf{K}_i \cdot \mathbf{A}\right]_2\right)$$
$$\mathbf{aVK} + [-\mathbf{K}'\mathbf{A}]_2 = \frac{b_{i^*}}{N} \cdot [\mathbf{K}_{i^*}\mathbf{A}]_2$$

Then Eq. (12) becomes

$$\hat{\sigma}_1^* \circ [\mathbf{A}]_2 = \begin{pmatrix}[1]_1 & m^*\end{pmatrix} \circ [\mathbf{K}_{i^*}\mathbf{A}]_2 + \hat{\sigma}_2^* \circ [\mathbf{UA}]_2 + \hat{\sigma}_3^* \circ [\mathbf{VA}]_2 \tag{13}$$

6. We have shown that given $\mathcal{A}$ wins the Hybrid_3 game and $\mathcal{B}$ does not abort, $\mathcal{B}$ wins the TS-UF-1 game. That is,

$$\Pr\left[\mathsf{Hybrid}_{4\mathcal{B}}(1^\lambda) = 1 | i' = i^*, \mathsf{Hybrid}_{3\mathcal{A}}(1^\lambda) = 1\right] = 1 \tag{14}$$

By conditional probability

$$\begin{aligned}\Pr\left[\mathsf{Hybrid}_{4\mathcal{B}}(1^\lambda) = 1\right] = \Pr\left[\mathsf{Hybrid}_{4\mathcal{B}}(1^\lambda) = 1 | i' = i^*, \mathsf{Hybrid}_{3\mathcal{A}}(1^\lambda) = 1\right] \\ \cdot \Pr\left[i' = i^* | \mathsf{Hybrid}_{3\mathcal{A}}(1^\lambda) = 1\right] \cdot \Pr\left[\mathsf{Hybrid}_{3\mathcal{A}}(1^\lambda) = 1\right]\end{aligned}$$

Using Eq. (9) and Eq. (14), we have

$$\Pr\left[\mathsf{Hybrid}_{4\mathcal{B}}(1^\lambda) = 1\right] = 1 \cdot \frac{1}{|\mathsf{Hon}|} \cdot \Pr\left[\mathsf{Hybrid}_{3\mathcal{A}}(1^\lambda) = 1\right]$$

Since

$$\Pr\left[\mathsf{Hybrid}_{3\mathcal{A}}(1^\lambda) = 1\right] > \mathsf{negl}(\lambda)$$

Then

$$\Pr\left[\mathsf{Hybrid}_{4\mathcal{B}}(1^\lambda) = 1\right] = \frac{1}{|\mathsf{Hon}|} \cdot \Pr\left[\mathsf{Hybrid}_{3\mathcal{A}}(1^\lambda) = 1\right] > \mathsf{negl}(\lambda)$$

Lemma B.11. For any non-uniform PPT adversary $\mathcal{A}$ for Hybrid_4, $\Pr\left[\mathsf{Hybrid}_{4\mathcal{A}}(1^\lambda) \to 1\right] = \mathsf{negl}(\lambda)$.

Proof. Hybrid_4 is – except with different notation – the same as the TS-UF-1 experiment described in [42] figure 1, where we use the following parameter choices: $t = k = l = 1$, $n = 2$, and we use Shamir sharing for the Share function. Note that for $(n, t) = (2, 1)$, this sharing scheme simply sets both shares equal to the secret. Then both honest signers in the TS-UF-1 game receive the same signing key. Additionally, the TS-UF-1 adversary outputs $(n, t, \mathsf{Cor}) = (2, 1, \emptyset)$ after receiving pp and the crs. We omit this step from Hybrid_4 for notational convenience.

The TSPS scheme of [42] achieves TS-UF-1 security under the $\mathcal{D}_k$-MDDH assumption ([42], theorem 2). Therefore, any PPT adversary $\mathcal{A}$ will win Hybrid_4 with only negligible probability.

Finally, Lemmas B.7 to B.11 together imply that for all non-uniform PPT adversaries $\mathcal{A}$ for Hybrid_0 (Forge-Exp),

$$\Pr\left[\mathsf{Forge\text{-}Exp}_{\mathcal{A}}(1^\lambda) \to 1\right] = \mathsf{negl}(\lambda)$$

Therefore, the STSPS scheme is secure.

C Anonymous Credentials Proofs of Security

In this section, we define and prove several notions of security for the anonymous credential scheme from Section 4.

Unlinkability says that the verifier should not know which attributes underly the credential, beyond the subset of revealed attributes. In the unlinkability security game, Fig. 5, the adversary corrupts all the issuers and chooses two sets of attributes $\mathsf{attrs}_0, \mathsf{attrs}_1$ that agree on some indices idx. The adversary can issue credentials on these attributes on behalf of any issuers. Finally, the challenger generates a showing based on attrs_0 or attrs_1, and the adversary should not be able to guess which attributes were used in the showing, except with negligible advantage.

1. **Setup:** Let $\mathcal{A}$ be the adversary.
   (a) The adversary is given 1^λ and then picks $N \in \mathbb{N}$ and l.
   (b) Then the challenger runs $\mathsf{Init}(1^\lambda, 1^N, 1^l) \to \mathsf{pp}$ and sends pp to the adversary.
   (c) For each $i \in [N]$, the adversary outputs $(\mathsf{vk}_i, \mathsf{hint}_i)$.
   (d) The challenger computes $\mathsf{Preprocess}(\mathbf{crs}_{\mathsf{TSPS}}, (\mathsf{hint}_i, \mathsf{vk}_i)_{i\in[N]}) \to (\mathsf{vk}, \mathsf{ak})$ and sends them to the adversary.
   (e) The challenger keeps two lists C_0, C_1, of the partial credentials received and two lists S_0, S_1, of the issuers of those partial credentials.
2. **Attribute Choice**: $\mathcal{A}$ chooses two arrays of attributes attrs_0 and attrs_1 such that $|\mathsf{attrs}_0| = |\mathsf{attrs}_1| = l$, and an array of indices $\mathsf{idx} \subset [l]$ such that $\forall i \in \mathsf{idx}, \mathsf{attrs}_{0,i} = \mathsf{attrs}_{1,i}$.
3. **Signing**: The adversary can query for a signing session on (b, i), as many times as they want. On each query, the challenger conducts a signing session between the honest user U and the adversary who plays issuer I_i as $\mathsf{Issue}(U(\mathsf{vk}_i, \mathsf{attrs}_b, [l]), I_i(\cdots, \mathsf{attrs}'_b, [l])) \to \{\mathsf{cred}, \perp\}$
   If $\mathcal{A}$ returns a valid cred, and $i \notin S_b$, then append i to S_b and append cred to C_b.[a]
4. **Validation:** If $|C_0| < t$ or $|C_1| < t$, then the game halts and outputs a uniformly random bit.
5. **Query:** The adversary can make queries of the form (b^*, idx^*). The challenger generates a showing from the partial credentials in C_{b^*}. The challenger computes $\mathsf{AggrCred}(\mathsf{pp}, \mathsf{attrs}_{b^*}, S_{b'}, t, \mathsf{ak}, C_{b^*}) \to \mathsf{cred}_{b^*}$ and $\mathsf{Show}(\mathsf{cred}_{b^*}, \mathsf{VK}, \mathsf{attrs}_{b^*}, \mathsf{idx}, t) \to (\mathsf{attrs}^*, \mathsf{idx}^*, \pi^*)$ and sends $(\mathsf{attrs}^*, \mathsf{idx}^*, \pi^*)$ to $\mathcal{A}$. The adversary can make a polynomial number of such queries.
6. **Challenge:** The challenger samples $b' \xleftarrow{\$} \{0, 1\}$ and generates a showing from the partial credentials in $C_{b'}$.
   The challenger computes $\mathsf{AggrCred}(\mathsf{pp}, \mathsf{attrs}_{b'}, S_{b'}, t, \mathsf{ak}, C_{b'}) \to \mathsf{cred}_{b'}$ and $\mathsf{Show}(\mathsf{cred}_{b'}, \mathsf{VK}, \mathsf{attrs}_{b'}, \mathsf{idx}, t) \to (\mathsf{attrs}', \mathsf{idx}, \pi)$ and sends $(\mathsf{attrs}', \mathsf{idx}, \pi)$ to $\mathcal{A}$.
7. **Response:** $\mathcal{A}$ responds with b^*. The game outputs 1 if $b^* = b'$ and outputs 0 otherwise.

[a] For simplicity we assume all attributes are revealed to the issuer during these signing sessions. Unlinkability should hold regardless.

Figure 5: $\mathsf{Unlinkability\text{-}Exp}_{\mathcal{A}}(1^\lambda)$

Definition C.1 (Unlinkability)**.** A credential scheme satisfies *unlinkability* if for all non-uniform PPT adversaries $\mathcal{A}$ in the GGM, we have

$$\Pr\Big[\mathsf{Unlinkability\text{-}Exp}_{\mathcal{A}}(1^\lambda) = 1\Big] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

where $\mathsf{Unlinkability\text{-}Exp}_{\mathcal{A}}$ is defined in Fig. 5.

Theorem C.1. The anonymous credentials scheme constructed in Construction 4.1 satisfies unlinkability (Definition C.1).

Proof. Assume toward contradiction that the scheme does not satisfy unlinkability. Then we can construct an adversary that breaks the witness indistinguishability of the GS proof system.

Given any adversary $\mathcal{A}$ for $\mathsf{Unlinkability\text{-}Exp}_{\mathcal{A}}(1^\lambda)$, there is an adversary $\mathcal{B}$ for $\mathsf{Witness\text{-}Indistinguishability\text{-}GS}_{\mathcal{B}}(1^\lambda)$ that wins with the same probability. $\mathcal{B}$ runs $\mathcal{A}$ internally and simulates $\mathsf{Unlinkability\text{-}Exp}_{\mathcal{A}}$. $\mathcal{B}$ outputs the statement that Show passes, and the witnesses $(\mathsf{cred}_0, \mathsf{attrs}_0)$ and $(\mathsf{cred}_1, \mathsf{attrs}_1)$. Then, the challenger picks a bit b' and $\mathcal{B}$ receives the GS proof $\pi_{b'}$ from the challenger. $\mathcal{B}$ passes this on to $\mathcal{A}$, who outputs b^*. Then $\mathcal{B}$ outputs b^*. From the perspective of $\mathcal{A}$, this is exactly the unlinkability game. If $b' = b^*$, $\mathcal{A}$ wins the unlinkability game and $\mathcal{B}$ wins the witness indistinguishability game.

Since the GS proof system satisfies witness indistinguishability, we have reached a contradiction. Therefore, the construction in Construction 4.1 satisfies unlinkability.

1. **Setup:** Let $\mathcal{A}$ be the adversary. The adversary is given 1^λ and then picks $N \in \mathbb{N}$ and l. Then the challenger runs $\mathsf{Init}(1^\lambda, 1^N, 1^l) \to \mathsf{pp}$ and sends pp to the adversary. Then the adversary picks $\mathsf{Cor} \subset [N]$, the set of corrupted parties. Let $\mathsf{Hon} = [N] \backslash \mathsf{Cor}$.
2. **Key Generation:**
   (a) For each $k \in \mathsf{Cor}$, the adversary outputs $(\mathsf{vk}_k, \mathsf{hint}_k)$.
   (b) For each $k \in \mathsf{Hon}$, the challenger computes $\mathsf{IssuerRegister}(\mathsf{pp}) \to (\mathsf{vk}_k, \mathsf{hint}_k, \mathsf{sk}_k)$.
   (c) The challenger computes $\mathsf{Preprocess}(\mathsf{pp}, [N], (\mathsf{hint}_k, \mathsf{vk}_k)_{k\in[N]}) \to (\mathsf{VK}, \mathsf{ak})$ and sends them to the adversary.
3. **Credential Queries:** The adversary may make queries of form (attrs, i) where attrs is a list of l attributes and $i \in \mathsf{Hon}$ is the issuer. The challenger records the query in Q_{Cred}, and runs the issuance protocol with i as the honest issuer and $\mathcal{A}$ as the user.
   $$\mathsf{Issue}(U(\cdots), I(\mathsf{sk}_i, \mathsf{attrs}' = \mathsf{attrs}, \mathsf{idx})) \to \{\mathsf{cred}, \perp\}$$
   The adversary can make any number of such credential queries.
4. **Forgery:** The adversary outputs a forgery $(\mathsf{attrs}^*, \mathsf{idx}^*, \pi, t)$. Let M be the number of honest issuers that issued a credential whose attributes agree with attrs^* on indices idx^*.
   The $\mathsf{Credential\text{-}Forge\text{-}Exp}_{\mathcal{A}}(1^\lambda)$ game outputs 1 if $\mathsf{Verify}(\mathsf{attrs}^*, \mathsf{idx}^*, \pi, t, \mathsf{VK}) = 1$ and $|\mathsf{Cor}| + M < t$. Else output 0.

Figure 6: $\mathsf{Credential\text{-}Forge\text{-}Exp}_{\mathcal{A}}(1^\lambda)$

Unforgeability says that the adversary should not be able to generate valid a showing on some attrs^* unless at least t issuers issued credentials that are consistent with attrs^* on the revealed indices.

In the unforgeability security game (Fig. 6), the adversary can corrupt some of the issuers and make issuing queries to the honest issuers on any attributes of their choice. In the end, the adversary outputs a forged credential showing for some attributes attrs^* revealed on indices idx^*. Let M be the number of honest issuers who issued a partial credential on some attrs that agree with attrs^* on indices idx^*. The adversary wins if the showing passes verification, but $M + |\mathsf{Cor}| < t$. Finally, note that during issuance, the user reveals all attributes to the issuer.

Definition C.2 (Unforgeability)**.** A credential scheme is *unforgeable* if for all non-uniform PPT adversaries $\mathcal{A}$ in the GGM, we have

$$\Pr\Big[\mathsf{Credential\text{-}Forge\text{-}Exp}_{\mathcal{A}}(1^\lambda) = 1\Big] \leq \mathsf{negl}(\lambda)$$

where $\mathsf{Credential\text{-}Forge\text{-}Exp}_{\mathcal{A}}$ is defined in Fig. 6.

Theorem C.2. The anonymous credentials scheme constructed in Construction 4.1 is unforgeable (Definition C.2).

Proof. Given any adversary $\mathcal{A}$ that wins $\mathsf{Credential\text{-}Forge\text{-}Exp}$ with non-negligible probability, we can either construct an adversary $\mathcal{B}$ that wins $\mathsf{Forge\text{-}Exp}$ – the unforgeability game for our signature scheme given in Fig. 2 – with non-negligible probability, or we can construct an adversary $\mathcal{C}$ that breaks the binding of the KZG commitment scheme.

$\mathcal{B}$ and $\mathcal{C}$ run $\mathcal{A}$ internally. If $\mathcal{A}$ wins $\mathsf{Credential\text{-}Forge\text{-}Exp}$, then it provides a valid GS proof that it knows a witness for the verification relation. By the knowledge soundness of Groth-Sahai, $\mathcal{B}$ and $\mathcal{C}$ can extract the witness, which includes a commitment $\mathsf{com} = \mathsf{KZG.Commit}(\mathsf{attrs})$. After extracting com, $\mathcal{B}$ and $\mathcal{C}$ can extract attrs by the extraction property of the KZG commitment.

Then there can be two cases:

1. Fewer than $t - |\mathsf{Cor}|$ honest issuers issued a signature on com. Then, $\mathcal{B}$ can output the witness as a forged signature to the $\mathsf{Forge\text{-}Exp}$ challenger. Since it passes verification, $\mathcal{B}$ wins $\mathsf{Forge\text{-}Exp}$.

2. At least $t - |\mathsf{Cor}|$ honest issuers issued a signature on com.

   Let Q be the list of Issue queries on com. Then the adversary must have revealed some attribute array attrs_0 during these queries, and provided an opening proof π_0.

   (a) If the attrs_0 array isn't identical for all queries, then $\exists \mathsf{attrs}_1$ such that $\mathsf{attrs}_0 \neq \mathsf{attrs}_1$, and the adversary must have provided an opening proof π_1 to open attrs_1.

   $\mathcal{C}$ can use $\mathsf{com}, \mathsf{attrs}_0, \pi_0, \mathsf{attrs}_1, \pi_1$, as its output. Since $\mathsf{attrs}_0 \neq \mathsf{attrs}_1$ and both proofs verified and produced a valid credential, then $\mathcal{C}$ wins the binding game for KZG.

   (b) If all queries in Q revealed the same array attrs_0.

   Recall that $\mathsf{com} = \mathsf{KZG.Commit}(\mathsf{attrs})$. After extracting com, $\mathcal{C}$ can extract attrs by the extraction property of the KZG commitment.

   Assume for contradiction that $\forall j \in \mathsf{idx}^*, \mathsf{attrs}_0[\mathsf{idx}^*[j]] = \mathsf{attrs}[\mathsf{idx}^*[j]]$. This implies that $\mathsf{attrs}_0 \in Q_{attr}$. We know that at least $t - |\mathsf{Cor}|$ queries were made on attrs_0. Then, $M \geq t - |\mathsf{Cor}|$, which implies that $\mathcal{A}$ would lose the

   $\mathsf{Unforgeability\text{-}Exp}_{\mathcal{A}}(1^\lambda)$ game. Since this is a contradiction, we know that $\exists j \in [\mathsf{idx}^*]$, s.t $\mathsf{attrs}_0[\mathsf{idx}^*[j]] \neq \mathsf{attrs}[\mathsf{idx}^*[j]]$.

   $\mathcal{C}$ can extract the KZG opening proof π_{KZG} from GS. Then $\mathcal{C}$ can output $\mathsf{com}, \mathsf{attrs}, \pi_{KZG}, \mathsf{attrs}_0, \pi_0$. Note that π_{KZG} and π_0 both reveal j for com, and they open to different values. Thus, we win the binding game for KZG.

1. **Setup:** Let $\mathcal{A}$ be the adversary. The adversary is given 1^λ and then picks $N \in \mathbb{N}$ and l. Then the challenger runs $\mathsf{Init}(1^\lambda, 1^N, 1^l) \to \mathsf{pp}$ and sends pp to the adversary. Then the adversary picks $\mathsf{Cor} \subset [N]$, the set of corrupted parties. Let $\mathsf{Hon} = [N] \backslash \mathsf{Cor}$.

2. **Key Generation:**

   (a) For each $k \in \mathsf{Hon}$, the challenger computes $\mathsf{IssuerRegister}(\mathsf{pp}) \to (\mathsf{vk}_k, \mathsf{hint}_k, \mathsf{sk}_k)$ and sends $(\mathsf{vk}_k, \mathsf{hint}_k)$ to the adversary.

   (b) For each $k \in \mathsf{Cor}$, the adversary outputs $(\mathsf{vk}_k, \mathsf{hint}_k)$.

   (c) The challenger computes $\mathsf{Preprocess}(\mathsf{pp}, [N], (\mathsf{hint}_k, \mathsf{vk}_k)_{k\in[N]}) \to (\mathsf{VK}, \mathsf{ak})$ and sends them to the adversary.

3. **Issuer Set Choice**: The adversary chooses t, and two sets of issuers I_0 and I_1, where $|I_0| = |I_1| = t$, and $I_0 \neq I_1$. It also outputs two sets of l attributes, attrs_0 and attrs_1, and a set of indices $\mathsf{idx} \subseteq [l]$ such that $\left(\mathsf{attrs}_{0j}\right)_{j\in\mathsf{idx}} = \left(\mathsf{attrs}_{1j}\right)_{j\in\mathsf{idx}}$.

4. **Credential Queries**: For each $b \in \{0,1\}$, and each $k \in I_b$, the challenger makes a credential request on attrs_b to issuer k. If $k \in \mathsf{Hon}$, the challenger computes $\mathsf{Issue}(U(\mathsf{vk}_k, \mathsf{attrs}_b, [l]), I(\mathsf{sk}_k, \mathsf{attrs}_b, [l])) \to \mathsf{cred}_{k,b}$.

   Else, it runs the issuing protocol with $\mathcal{A}$ as the issuer and outputs $\mathsf{Issue}(U(\mathsf{vk}_k, \mathsf{attrs}_b, [l]), I(\cdots, \mathsf{attrs}_b, [l])) \to \{\mathsf{cred}_{k,b}, \perp\}$ If the user does not obtain a valid credential, the game immediately ends and outputs a uniformly random bit.

5. **Challenge:** The challenger samples $b' \xleftarrow{\$} \{0,1\}$. Then it computes $\mathsf{AggrCred}(\mathsf{pp}, \mathsf{attrs}_{b'}, I_{b'}, t, \mathsf{ak}, \{\mathsf{cred}_{k,b'}\}_{k\in I_{b'}}) \to \mathsf{cred}$ and $\mathsf{Show}(\mathsf{cred}, \mathsf{VK}, \mathsf{attrs}_{b'}, \mathsf{idx}, t) \to (\mathsf{attrs}', \mathsf{idx}, \pi)$ and sends $(\mathsf{attrs}', \mathsf{idx}, \pi)$ to $\mathcal{A}$.

6. **Response:** $\mathcal{A}$ responds with b^*. The game outputs 1 if $b^* = b'$ and outputs 0 otherwise.

Figure 7: $\mathsf{Issuer\text{-}Hiding\text{-}Exp}_{\mathcal{A}}(1^\lambda)$

Issuer-hiding says that a credential showing should not reveal which issuers issued the partial credentials for the showing, except what is revealed by the attributes. In the issuer-hiding security game (Fig. 7), the adversary chooses two sets of issuers I_0, I_1 of the same size and two sets of attributes $\mathsf{attrs}_0, \mathsf{attrs}_1$ that agree on the revealed indices idx. Then for each $b \in \{0,1\}$, the challenger requests partial credentials on attrs_b from each issuer in I_b. Then the challenger samples $b' \xleftarrow{\$} \{0,1\}$ and generates a showing on the credentials obtained for $\mathsf{attrs}_{b'}$ that reveals only the attributes at indices idx. The adversary should not be able to guess b' except with negligible advantage.

Definition C.3 (Issuer Hiding)**.** A credential scheme is *issuer-hiding* if for all non-uniform PPT adversaries $\mathcal{A}$ in the GGM, we have:

$$\Pr\left[\mathsf{Issuer\text{-}Hiding\text{-}Exp}_{\mathcal{A}}(1^\lambda) = 1\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

where $\mathsf{Issuer\text{-}Hiding\text{-}Exp}_{\mathcal{A}}$ is defined in Fig. 7.

Theorem C.3. The anonymous credentials scheme constructed in Construction 4.1 is issuer-hiding (Definition C.3).

Proof. Assume toward contradiction that the scheme does not satisfy issuer-hiding. Then there is an adversary $\mathcal{A}$ for $\mathsf{Issuer\text{-}Hiding\text{-}Exp}$ that wins with probability $\frac{1}{2} + \mathsf{negl}(\lambda)$. Then we can construct an adversary $\mathcal{B}$ that breaks the witness indistinguishability of the GS proof system.

$\mathcal{B}$ runs $\mathcal{A}$ internally and plays the role of the challenger in Issuer-Hiding-Exp. For each $b \in \{0,1\}$, $\mathcal{B}$ computes the aggregate credential: $\mathsf{AggrCred}(\mathsf{pp}, \mathsf{attrs}_b, I_b, t, \mathsf{ak}, \{\mathsf{cred}_{k,b}\}_{k \in I_b}) \to \mathsf{cred}_b$. Then $\mathcal{B}$ outputs the statement Show, and the witnesses cred_0 and cred_1.

Then, the challenger picks a bit b' and $\mathcal{B}$ receives the GS proof $\pi_{b'}$ from the challenger. $\mathcal{B}$ passes this on to $\mathcal{A}$, who outputs b^*. Then $\mathcal{B}$ outputs b^*. From the perspective of $\mathcal{A}$, this is exactly Issuer-Hiding-Exp. If $b' = b^*$, then $\mathcal{A}$ wins Issuer-Hiding-Exp and $\mathcal{B}$ wins the witness indistinguishability game.

Since the GS proof system satisfies witness indistinguishability, we have reached a contradiction. Therefore, the construction in Construction 4.1 satisfies issuer-hiding.