





Interactive Threshold Mercurial Signatures and Applications

Masayuki Abe^{1,2} , Masaya Nanri² , Octavio Perez Kempner¹ , and Mehdi Tibouchi^{1,2} 

¹ NTT Social Informatics Laboratories
{msyk.abe,octavio.perezkempner,mehdi.tibouchi}@ntt.com

² Kyoto University
nanri.masaya.26n@st.kyoto-u.ac.jp

Abstract. Mercurial signatures are an extension of equivalence class signatures that allow malleability for the public keys, messages, and signatures within the respective classes. Unfortunately, the most efficient construction to date suffers from a weak public key class-hiding property, where the original signer with the signing key can link the public keys in the same class. This is a severe limitation in their applications, where the signer is often considered untrustworthy of privacy.

This paper presents two-party and multi-party *interactive threshold mercurial signatures* that overcome the above limitation by eliminating the single entity who knows the signing key. For the general case, we propose two constructions. The first follows the same interactive structure as the two-party case, avoiding complex distributed computations such as randomness generation, inversion, and multiplication, and even eliminates the need for private communication between parties. The second is based on a blueprint for general multi-party computation using verifiable secret sharing, but adopting optimizations.

We show applications in anonymous credential systems that individually fit the two-party and multi-party constructions. In particular, in the two-party case, our approach provides stronger privacy by completely removing the trust in the authorities. We also discuss more applications, from blind signatures to multi-signatures and threshold ring signatures.

Finally, to showcase the practicality of our approach, we implement our interactive constructions and compare them against related alternatives.

Keywords: Mercurial Signatures, Equivalence Class Signatures, Threshold Signatures, Class-Hiding, Anonymous Credentials

1 Introduction

Equivalence Class Signatures (EQS) [HS14, FHS19] are malleable signatures [CKLM14] defined over a vector of group elements. They are structure-preserving [AFG⁺10, AGHO11] and, thus, equipped with a bilinear pairing so that public keys and signatures also consist of group elements. This allows them to be verified evaluating pairing-product equations without requiring any specific encoding. They have been extensively used as a building block for many cryptographic primitives, including anonymous credentials (*e.g.*, [HS14, DHS15, HS21, FHS19, CLPK22]), blind signatures [FHS15, FHKS16], group signatures [DS18, BHKS18, BHSB19] and sanitizable signatures [BLL⁺19] to name a few. Related primitives include signatures with flexible public keys [BHKS18] and Mercurial Signatures (MS) [CL19, CL21, CLPK22, MBG⁺23], which is the main focus of this paper.

EQS allow one to randomize a signature, adapting it to a new message and a new public key in the same equivalence class. Security requires adapted signatures to look like freshly computed ones (signature adaption) and some notion of unlinkability when adapting messages and public keys (also referred to as class-hiding). In many applications, the adversary does not know the discrete logarithms of the message vector, and thus, *message class-hiding* is implied by the decisional Diffie-Hellman assumption. Recently, Bauer and Fuchsbauer [BF20] proposed an EQS construction based on the idea of signatures on randomizable ciphertexts [BFPV11], achieving a stronger notion of message class-hiding covering the case in which the adversary knows the discrete logarithms of the message vector. Consequently, for message class-hiding, all possible scenarios are well-studied.

Regarding public key class-hiding, however, no satisfactory solution has been put forth so far. All known constructions only provide public key class-hiding as long as the adversary does not know the signing key. In other words, the original signer must be trusted. This is most evident for anonymous credentials where MS have been used to provide issuer-hiding features [CLPK22, MBG⁺23, CDLP22]

and to build delegatable schemes [CL19, CL21] where the issuer has to be trusted for issuer-hiding. That is, given a valid key pair (sk, pk) and a randomized public key pk' of pk , the issuer with sk can determine whether pk' is related to pk . Thus, issuers can identify if a credential has been issued to a user belonging to their organization, even if they do not know specifically to whom. While this can be tolerated in some scenarios, it can suffice to fully de-anonymize users in others. The situation is even worse for delegatable credentials because every user in the credential chain must be trusted. Otherwise, an adversary can identify chains containing a corrupted user by recognizing randomized keys.

1.1 Our Contributions

We propose *Threshold Mercurial Signature* (TMS) schemes where signing keys are distributed among signers, and a quorum cooperates to produce mercurial signatures. With secure distributed key generation, no signer below the threshold knows the key. This ensures no sub-threshold parties can access the secret key, justifying the weak public key class-hiding property and broadening the privacy-preserving applications of MS. Our contributions are summarized as follows.

1. **Two-party Mercurial Signature Scheme (Sec. 4):** We develop a distributed two-party signing protocol for the MS scheme described in [FHS19, CL19]. Two signers with additively shared signing keys interact with each other to generate a mercurial signature on a given message. The protocol consists of three sequenced moves and is secure against static corruption. This minimal setting is not only essential for illustrating our ideas for eliminating expensive distributed computations but also serves a crucial role in issuer-hiding anonymous credentials, eliminating the need for trustworthy issuers (see Sec. 1.2 and 4.4 for more discussion and details). This solves an open problem of anonymous credentials based on EQS [CLPK22].
2. **Multi-party Mercurial Signature Scheme (Sec. 5):** We generalize the two-party protocol to the t -out-of- n threshold setting. It is not a straightforward task due to the asymmetric nature of our two-party protocol. In this protocol, the three moves of interaction in the two-party case are simulated with the signers lined up in order, taking input from the previous signer and sending the result of local computation to the next one. A malicious signer in the sequence complicates security analysis, but the essential idea remains unchanged. As an application, we demonstrate how this t -out-of- n scheme is useful for delegatable credentials (see Sec. 5.3 for details). More applications are discussed in Appendix B, including blind signatures, multi-signatures, and threshold ring signatures.
3. **Experimental Evaluation (Sec. 6):** The actual efficiency depends on the instantiation of underlying zero-knowledge proofs of knowledge and optimization of the group operations. Besides, the computational complexity scales linearly with the number of parties. For this reason, we implement our protocols and report benchmarks considering different numbers of parties and application settings. The overhead is relatively minor compared to the implementation of the original MS, allowing us to produce signatures in less than 0.5s for practical scenarios involving ten parties.

1.2 Technical Overview

Distributed Signing: In the MS from [CL19], defined over pairing groups generated by G and \hat{G} , a signature (Z, Y, \hat{Y}) on message M is computed as $Z = M^{xy}$, $Y = G^{1/y}$, $\hat{Y} = \hat{G}^{1/y}$ with signing key x and ephemeral randomness y . Let $[x]$ denote additive (or polynomial) shares of x , and consider the signers having $[x]$ collaborate to compute a signature. A naive approach would be to generate shared randomness $[y]$, compute shared product $[xy]$ and shared inverse $[1/y]$, and reconstruct xy and $1/y$ on the exponent of M , G and \hat{G} . This would require three invocations of distributed key generation (DKG) protocols involving commitments or verifiable secret sharing to avoid rushing adversaries that attempt to bias the resulting signature. Since this can be a cumbersome task for both two-party and multi-party cases, we first consider constructions that do not require such machinery. Instead, we observe that the bias caused by a rushing adversary can be ignored since mercurial signatures are malleable. In brief, the recipient can remove the bias by adapting the signature. In light of this observation, our first proposal incorporates the following techniques:

- We generate ephemeral randomness y in a multiplicative manner. Denote the multiplicative sharing by $\langle y \rangle$. This makes shared inversion $\langle 1/y \rangle$ a local computation. Computing M^{xy} could be done first by computing M^x using $[x]$ and then compute $(M^x)^y$ using $\langle y \rangle$ in sequence.
- However, the above method leaks intermediate value M^x that prevents the security proof from going through. We develop efficient blind computation of M^{xy} where M^x is blinded by random factor Y^r and unblinded with G^r . Namely, the signers first compute $Y^r M^x$ and then $(Y^r M^x)^y$, which can be done efficiently by the sequence of local computations. Since $(Y^r M^x)^y = G^r M^{xy}$, unblinding it with G^r results in M^{xy} as desired.
- In the threshold case, where more than two parties are involved in the signing process, we found that the randomness r mentioned above is insufficient to simulate more than two honest parties simultaneously. To address this issue, we introduce additional randomness into the signing protocol without altering its fundamental structure. This is achieved by incorporating random additive shares of zero into the intermediate computations, which cancel out when $Y^r M^x$ is computed correctly. We develop a technique to generate these shares solely through public communication between the parties.

We also consider several optimizations to the naive approach based on multi-party computation and propose a second construction based on Abe’s multiplication protocol [Abe99], which nicely fits our needs as explained in Sec. 5.2.

Enhancing Issuer-Hiding in Anonymous Credentials: The authorities’ role in a credential system with MS is to issue a signature on the user’s attributes. As discussed earlier, however, authorities are trusted for privacy in the sense that they do not abuse their signing key to trace the signatures. Plug-in replacement of MS with our TMS immediately raises the bar for violating users’ privacy. Nevertheless, threshold authorities are assumed to not collude to retain privacy.

We eliminate such an unverifiable trust using our TMS. In our issuing protocol, the authority and user Alice engage in the two-party TMS. The resulting signature verifies with the joint public key from the authority and Alice. When Alice anonymously shows the credential, she proves in zero knowledge that the randomized joint key properly includes a valid, authoritative public key, and she knows the randomized secret key for the remainder. This way, Alice can protect her privacy by herself without trusting the authority.

1.3 Related Work

Mercurial Signatures. There are two constructions of MS in the literature: one by Crites and Lysyanskaya [CL19] and another by Connolly *et al.* [CLPK22]. The MS from [CLPK22] was recently shown to be flawed in [BFR24, BF24], and it is broken. In Sec. 2.2, we recall the construction from [CL19]. As mentioned, it presents a major drawback, as any signer can track randomizations of previously issued signatures. This is because a public key pk is a vector of elements, and any randomization is just a multiplication in the exponent by the same randomization factor ρ . Hence, given knowledge of a secret key sk and any pk' , it suffices to multiply pk' in the exponent by the inverse of sk . Consequently, if all elements are the same, it must be the case that pk' is a randomization of pk for some ρ . Our work presents a threshold version for [CL19] that, instead of getting a multiplicative share in the exponent of each element in the public key, we get an additive share. As a result, we can provide a stronger class-hiding notion, as further discussed in Sec. 4.4.

Pointcheval-Sanders signatures. Very recently, Sanders and Traoré [ST24] proposed a modified version of Pointcheval-Sanders (PS) signatures [PS16, PS18] to build an efficient issuer-hiding mechanism for anonymous credentials with strong security guarantees. Their approach consists of letting credential verifiers define an access policy for a set of issuers. More precisely, users take the verifier’s access policy to adapt their signature to verify if and only if the policy is satisfied (*i.e.*, the user’s signature/credential was signed by one of the issuers in the set). For security, verifiers must compute a zero-knowledge proof attesting to the correct computation of their access policy for the issuers’ set. In other words, this approach can be seen as letting each verifier define a custom common reference string (CRS) as their access policy, and the zero-knowledge proof attests to the correct computation of said CRS. Our approach to anonymous credentials resembles [ST24], and we borrow their NIZK proof. However, in our case, verifiers only specify the issuer’s set as their access policy, and our

solution does not require any proof of knowledge for the hidden attributes during the showing. Furthermore, we provide backward compatibility with previous attribute-based credentials constructions from EQS that provide revocation and auditability features [DHS15, CDLP22], potentially covering a more comprehensive range of functionalities.

Threshold Signatures. The ongoing NIST standardization effort related to threshold signatures [BP23] motivated many recent works tackling different settings, *e.g.*, [BCK⁺22, TZ22, CKM⁺23b, CKM23a, CKP⁺23]. Considering pairing-based constructions, threshold versions of the BLS signature [BLS01, BLS04] such as [Bol03] have gained significant attention over the past years, with security proven in the adaptive setting [BL22, BCK⁺22, DR24]. Threshold versions of BLS can be verified as a regular signature and are key-randomizable [DS19]. However, they are not structure-preserving and cannot be used as an alternative for EQS/MS. This is the first work to address the construction of threshold schemes for EQS. Closely related work to ours by Crites *et al.* [CKP⁺23] presented (non-interactive) Threshold Structure-Preserving Signatures. Their motivation was to have a drop-in replacement for standard SPS in the threshold setting. While the non-interactive setting is attractive and allows the authors to propose constructions compatible with the UC framework, this comes at the cost of using an indexed message space. In particular, a relatively new assumption called *Indexed Diffie-Hellman Message Space* is required to prove security. Very recent work by Mitrokotsa *et al.* [MMS⁺24] overcomes the previous limitation of [CKP⁺23] by removing the need of an indexed space. However, we stress that none of these works are EQS (let alone MS). Moreover, the constructions provided are not even randomizable. The indexed message space used in [CKP⁺23] defines an equivalence class, but this does not carry over to the TSPS construction (for a message m , \hat{G}^m always stays as is, and thus, m is fixed). Looking at [MMS⁺24], it is a tag-based construction whose tag is not randomizable (see σ_1 in [MMS⁺24]).

We take a different approach considering an interactive signing process. As we show, our approach offers several advantages for different applications where non-interactive TSPS fall short. Thus, our contribution broadens the scope of threshold SPS to include EQS, opening new research directions (see, for instance, the case of threshold ring signatures [BSS02] from TMS discussed in Appendix B.3).

Multi-signatures. Multi-signatures are a special case of threshold signatures where the threshold $t = n$. Recent work mostly focuses on pairing-free and non-interactive constructions (*e.g.*, [DEF⁺19, NRS21, AB21, BCK⁺22]), compatible with existing deployments in the blockchain sphere. Our approach is more general and focused on privacy-preserving applications that could benefit from malleable signatures with added functionalities and stronger security properties.

1.4 Organization

We give the preliminaries in Sec. 2. Syntax and security properties of TMS are presented in Sec. 3. Our two-party signing protocol is discussed in Sec. 4. We dedicate Sec. 5 to the threshold case. We report our experimental evaluation in Sec. 6 before concluding in Sec. 7. A detailed presentation of zero-knowledge proofs used in this work is given in Appendix A.

2 Preliminaries

Notation. The set of integers $1, 2, \dots, n$ is denoted $[n]$. We call \mathbb{Z}_p the ring of integers modulus p if $p \in \mathbb{N}$. For a set \mathcal{S} and $r \in \mathcal{S}$, $r \leftarrow_{\$} \mathcal{S}$ denotes that r has been sampled uniformly randomly from \mathcal{S} . The security parameter κ is usually passed in unary form. We use λ for Lagrange coefficients, and we denote the adversary's state by st . Let BGGen be a PPT algorithm that on input 1^κ , returns public parameters $\text{pp} = (p, \mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_\top, G, \hat{G}, e)$ describing an asymmetric bilinear group where $\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_\top$ are cyclic groups of prime order p with $\lceil \log_2 p \rceil = \kappa$, G and \hat{G} are generators of \mathbb{G} and $\hat{\mathbb{G}}$, and $e : \mathbb{G} \times \hat{\mathbb{G}} \rightarrow \mathbb{G}_\top$ is an efficiently computable (non-degenerate) bilinear map. pp is considered Type-III if no efficiently computable isomorphism between \mathbb{G} and $\hat{\mathbb{G}}$ is known. We then assume that the following DDH assumption in \mathbb{G} holds for BGGen , as well as $\hat{\mathbb{G}}$.

DDH Assumption. Let BGGen be a bilinear group generator that outputs public parameters pp . The *decisional Diffie-Hellman assumption* holds relative to \mathbb{G} for BGGen , if for all p.p.t adversaries \mathcal{A} the following probability is negligible,

$$\Pr \left[\text{pp} \leftarrow \text{BGGen}(1^\kappa); r, s, t \leftarrow \mathbb{Z}_p; b \leftarrow \{0, 1\} : b^* = b \right] - \frac{1}{2}$$

2.1 Zero-Knowledge Proofs of Knowledge

We require secure Zero-Knowledge Proofs of Knowledge (ZKPoK) that are complete, zero-knowledge, and knowledge sound. Many instantiations are available in different models and with different assumptions, directly affecting our protocols' security. In this paper, for presentation and performance, we consider a non-interactive form of zero-knowledge proofs that allows online witness extraction. It is available in the random oracle model for a stand-alone execution where our implementation resorts. Alternatively, the ZKPoK's can be instantiated via interactive five-round PoK's in the standard model [GK96]. We leave the evaluation of other variants, such as using Fischlin's transform [Fis05, CL24b], for future research.

2.2 Mercurial Signatures

We recap syntax and security notions of MS as presented in [CL19]. We recall that MS are EQS that support key randomization. Let \mathcal{R} be an equivalence relation where $[x]_{\mathcal{R}} = \{y | \mathcal{R}(x, y)\}$ denotes the equivalence class of which x is a representative. As in [CL19], we will loosely consider parametrized relations and say they are well-defined as long as the corresponding parameters are.

Definition 1 (Mercurial signature). A MS scheme for parametrized equivalence relations $\mathcal{R}_M, \mathcal{R}_{\text{pk}}, \mathcal{R}_{\text{sk}}$ is a tuple of the following polynomial-time algorithms, which are deterministic algorithms unless otherwise stated:

$\text{PGen}(1^\kappa) \rightarrow \text{pp}$: On input the security parameter 1^κ , this PPT algorithm outputs the public parameters pp . This includes parameters for the parametrized equivalence relations $\mathcal{R}_M, \mathcal{R}_{\text{pk}}$, and \mathcal{R}_{sk} so they are well-defined. It also includes parameters for the algorithms sample_ρ and sample_μ , which sample key and message converters, respectively.

$\text{KGen}(\text{pp}, \ell) \rightarrow (\text{pk}, \text{sk})$: On input the public parameters pp and a length parameter ℓ , this PPT algorithm outputs a key pair (pk, sk) . The message space \mathcal{M} is well-defined from pp and ℓ . This algorithm also defines a correspondence between public and secret keys: we write $(\text{pk}, \text{sk}) \in \text{KGen}(\text{pp}, \ell)$ if there exists a set of random choices that KGen could make to output (pk, sk) .

$\text{Sign}(\text{pp}, \text{sk}, M) \rightarrow \sigma$: On input the signing key sk and a message $M \in \mathcal{M}$, this PPT algorithm outputs a signature σ .

$\text{Verify}(\text{pp}, M, \sigma, \text{pk}) \rightarrow 0/1$: On input the public key pk , a message $M \in \mathcal{M}$, and a purported signature σ , output 0 or 1.

$\text{ConvertSK}(\text{sk}, \rho) \rightarrow \text{sk}'$: On input sk and a key converter $\rho \in \text{sample}_\rho$, output a new secret key $\text{sk}' \in [\text{sk}]_{\mathcal{R}_{\text{sk}}}$.

$\text{ConvertPK}(\text{pk}, \rho) \rightarrow \text{pk}'$: On input pk and a key converter $\rho \in \text{sample}_\rho$, output a new public key $\text{pk}' \in [\text{pk}]_{\mathcal{R}_{\text{pk}}}$.

$\text{ConvertSig}(\text{pk}, M, \sigma, \rho) \rightarrow \sigma'$: On input pk , a message $M \in \mathcal{M}$, a signature σ , and key converter $\rho \in \text{sample}_\rho$, this PPT algorithm returns a new signature σ' .

$\text{ChgRep}(\text{pk}, M, \sigma, \mu) \rightarrow (M', \sigma')$: On input pk , a message $M \in \mathcal{M}$, a signature σ , and a message converter $\mu \in \text{sample}_\mu$, this PPT algorithm computes a new message $M' \in [M]_{\mathcal{R}_M}$ and a new signature σ' and outputs (M', σ') .

Definition 2 (Correctness). A MS scheme for parametrized equivalence relations $\mathcal{R}_M, \mathcal{R}_{\text{pk}}, \mathcal{R}_{\text{sk}}$ is correct if it satisfies the following conditions for all κ , for all $\text{pp} \in \text{PGen}(1^\kappa)$, for all $\ell > 1$, for all $(\text{pk}, \text{sk}) \in \text{KGen}(\text{pp}, \ell)$:

- *Verification*: $\forall M \in \mathcal{M}, \forall \sigma \in \text{Sign}(\text{sk}, M), \text{Verify}(\text{pk}, M, \sigma) = 1$.
- *Key conversion*: $\forall \rho \in \text{sample}_\rho, (\text{ConvertPK}(\text{pk}, \rho), \text{ConvertSK}(\text{sk}, \rho)) \in \text{KGen}(\text{pp}, \ell)$. Moreover, $\text{ConvertSK}(\text{sk}, \rho) \in [\text{sk}]_{\mathcal{R}_{\text{sk}}}$ and $\text{ConvertPK}(\text{pk}, \rho) \in [\text{pk}]_{\mathcal{R}_{\text{pk}}}$.

- *Signature conversion*: $\forall M \in \mathcal{M}, \forall \sigma$ such that $\text{Verify}(\text{pk}, M, \sigma) = 1, \forall \rho \in \text{sample}_\rho, \forall \sigma' \in \text{ConvertSig}(\text{pk}, M, \sigma, \rho), \text{Verify}(\text{ConvertPK}(\text{pk}, \rho), M, \sigma') = 1$.
- *Change of message representative*: $\forall M \in \mathcal{M}, \forall \sigma$ such that $\text{Verify}(\text{pk}, M, \sigma) = 1, \forall \mu \in \text{sample}_\mu, \text{Verify}(\text{pk}, M', \sigma') = 1$, where $(M', \sigma') = \text{ChgRep}(\text{pk}, M, \sigma, \mu)$. Moreover, $M' \in [M]_{\mathcal{R}_M}$.

Definition 3 (Unforgeability). A MS scheme for parametrized equivalence relations $\mathcal{R}_M, \mathcal{R}_{\text{pk}}, \mathcal{R}_{\text{sk}}$ is unforgeable if for all polynomial-length parameters $\ell(\kappa)$ and any PPT adversary \mathcal{A} having access to a signing oracle, the following probability is negligible,

$$\Pr \left[\begin{array}{l} \text{pp} \leftarrow \text{PGen}(1^\kappa) \\ (\text{sk}, \text{pk}) \leftarrow \text{KGen}(\text{pp}, \ell(\kappa)) \\ (\text{pk}^*, M^*, \sigma^*) \leftarrow \mathcal{A}^{\text{Sign}(\text{sk}, \cdot)}(\text{pk}) \end{array} : \begin{array}{l} \forall M \in Q, [M^*]_{\mathcal{R}_M} \neq [M]_{\mathcal{R}_M} \\ \wedge [\text{pk}^*]_{\mathcal{R}_{\text{pk}}} = [\text{pk}]_{\mathcal{R}_{\text{pk}}} \\ \wedge \text{Verify}(M^*, \sigma^*, \text{pk}^*) = 1 \end{array} \right],$$

where Q is the set of queries that \mathcal{A} has issued to the signing oracle.

Definition 4 (Class-Hiding). A MS scheme is class-hiding if it satisfies the following two properties:

- *Message class-hiding*: if the advantage of any PPT adversary \mathcal{A} defined by $\text{Adv}_{\text{MS}, \mathcal{A}}^{\text{MSG-CH}}(\kappa) := 2 \cdot \Pr[\text{Exp}_{\text{MS}, \mathcal{A}}^{\text{MSG-CH}}(\kappa) \Rightarrow \text{true}] - 1 = \epsilon(\kappa)$.
- *Public key class-hiding*: if the advantage of any PPT adversary \mathcal{A} defined by $\text{Adv}_{\text{MS}, \mathcal{A}}^{\text{PK-CH}}(\kappa) := 2 \cdot \Pr[\text{Exp}_{\text{MS}, \mathcal{A}}^{\text{PK-CH}}(\kappa) \Rightarrow \text{true}] - 1 = \epsilon(\kappa)$.

The experiments $\text{Exp}_{\text{MS}, \mathcal{A}}^{\text{MSG-CH}}(\kappa)$ and $\text{Exp}_{\text{MS}, \mathcal{A}}^{\text{PK-CH}}(\kappa)$ are defined as follows:

Experiment $\text{Exp}_{\text{MS}, \mathcal{A}}^{\text{MSG-CH}}(\kappa)$

$\text{pp} \leftarrow \text{PGen}(1^\kappa); b \leftarrow \{0, 1\}; M_1 \leftarrow \mathcal{M}; M_2^0 \leftarrow \mathcal{M}; M_2^1 \leftarrow [M_1]_{\mathcal{R}_M}$
 $b' \leftarrow \mathcal{A}(\text{pp}, M_1, m_2^b); \text{return } b = b'$

Experiment $\text{Exp}_{\text{MS}, \mathcal{A}}^{\text{PK-CH}}(\kappa)$

$\text{pp} \leftarrow \text{PGen}(1^\kappa); b \leftarrow \{0, 1\}; \rho \leftarrow \text{sample}_\rho(\text{pp}); (\text{sk}_1, \text{pk}_1) \leftarrow \text{KGen}(\text{pp}, \ell(\kappa))$
 $(\text{sk}_2^0, \text{pk}_2^0) \leftarrow \text{KGen}(\text{pp}, \ell(\kappa)); \text{pk}_2^1 \leftarrow \text{ConvertPK}(\text{pk}_1, \rho); \text{sk}_2^1 \leftarrow \text{ConvertSK}(\text{sk}_1, \rho)$
 $b' \leftarrow \mathcal{A}^{\text{Sign}(\text{pp}, \text{sk}_1, \cdot), \text{Sign}(\text{pp}, \text{sk}_2^b, \cdot)}(\text{pk}_1, \text{pk}_2^b); \text{return } b = b'$

Definition 5 (Origin-hiding). A MS scheme is origin-hiding if for all $\kappa, \text{pp} \in \text{PGen}(1^\kappa), \text{pk}^*, m$, and σ , the following two properties hold:

1. if $\text{Verify}(\text{pk}, M, \sigma) = 1$ and $\mu \leftarrow \text{sample}_\mu$, then $\text{ChgRep}(\text{pk}^*, m, \sigma, \mu)$ outputs uniformly random $M' \in [M]_{\mathcal{R}_M}$ and $\sigma' \in \{\hat{\sigma} \mid \text{Verify}(\text{pk}^*, M', \sigma') = 1\}$.
2. if $\text{Verify}(\text{pk}, M, \sigma) = 1$ and $\rho \leftarrow \text{sample}_\rho$, then $\text{ConvertSig}(\text{pk}^*, M, \sigma, \rho)$ outputs a uniformly random $\sigma' \in \{\hat{\sigma} \mid \text{Verify}(\text{ConvertPK}(\text{pk}^*, \rho), M, \hat{\sigma}) = 1\}$ and $\text{ConvertPK}(\text{pk}^*, \rho)$ outputs a uniformly random element of $[\text{pk}^*]_{\mathcal{R}_{\text{pk}}}$.

In the following, we present the MS by Crites and Lysyanskaya [CL19], which is an extension of the EQS from [FHS19]. It's the state-of-the-art signature in terms of efficiency and has its security proven in the generic group model for Type-III pairings. The message space is $(\mathbb{G}^*)^\ell$ where ℓ is the length of the message vector. We recall that all elements of a vector $(M)_{i \in [\ell]} \in (\mathbb{G}^*)^\ell$ share different mutual ratios that depend on their discrete logarithms. Hence, it is possible to partition $(\mathbb{G}^*)^\ell$ into equivalence classes given by: $\mathcal{R} = \{(M, M') \in (\mathbb{G}^*)^\ell \times (\mathbb{G}^*)^\ell \mid \exists s \in \mathbb{Z}_p^* : M' = M^s\} \subseteq (\mathbb{G}^*)^\ell$. Moreover, an analogous relation can be defined for the public keys, inducing equivalence classes on the key space as well.

$\text{PGen}(1^\kappa) \rightarrow \text{pp}$: **return** $\text{BGen}(1^\kappa)$.

$\text{KGen}(\text{pp}, \ell) \rightarrow (\text{pk}, \text{sk})$: $\forall 1 \leq i \leq \ell : x_i \leftarrow \mathbb{Z}_p^*; \text{sk} \leftarrow (x_i)_{i \in [\ell]}; \text{pk} \leftarrow (\hat{G}^{x_i})_{i \in [\ell]}$ **return** (pk, sk) .

$\text{Sign}(\text{pp}, \text{sk}, M) \rightarrow \sigma$: $y \leftarrow \mathbb{Z}_p^*; Z \leftarrow (\prod_{i=1}^\ell M_i^{x_i})^y; Y \leftarrow G^{\frac{1}{y}}; \hat{Y} \leftarrow \hat{G}^{\frac{1}{y}}$

return (Z, Y, \hat{Y}) .

$\text{Verify}(\text{pp}, M, \sigma, \text{pk} = (\hat{X})_{i \in [\ell]}) \rightarrow 0/1$:

return $\prod_{i=1}^\ell e(M_i, \hat{X}_i) = e(Z, \hat{Y}) \wedge e(Y, \hat{G}) = e(G, \hat{Y})$.

$\text{ConvertSK}(\text{sk}, \rho) \rightarrow \text{sk}' : \text{sk}' \leftarrow \rho \cdot \text{sk}; \text{return } \text{sk}'.$
 $\text{ConvertPK}(\text{pk}, \rho) \rightarrow \text{pk}' : \text{pk}' \leftarrow \text{pk}^\rho; \text{return } \text{pk}'.$
 $\text{ConvertSig}(\text{pk}, M, \sigma, \rho) \rightarrow \sigma' : \psi \leftarrow_{\$} \mathbb{Z}_p^*; \text{return } (Z^{\psi\rho}, Y^{\frac{1}{\psi}}, \hat{Y}^{\frac{1}{\psi}}).$
 $\text{ChgRep}(\text{pk}, M, \sigma, \mu) \rightarrow (M', \sigma') : \psi \leftarrow_{\$} \mathbb{Z}_p^*; M' \leftarrow M^\mu; \sigma' \leftarrow (Z^{\psi\mu}, Y^{\frac{1}{\psi}}, \hat{Y}^{\frac{1}{\psi}}) \text{ return } (M', \sigma').$

Theorem 6 ([CL19]). *The above MS scheme is unforgeable, public key class-hiding and origin-hiding in the generic group model for Type-III bilinear groups. Moreover, it is message class-hiding if the DDH assumption holds in \mathbb{G} .*

2.3 Verifiable Secret Sharing

Our construction from Sec. 5.2 uses the verifiable secret sharing scheme by Pedersen [Ped92]. In this paper we follow the notation in [Abe99]. Let G and H be two elements of \mathbb{G} s.t. the discrete logarithm of H with base G is unknown. To share a secret y in \mathbb{Z}_p , a dealer first chooses two t -degree random polynomials $F_y(X)$ and $D_y(X)$ from $\mathbb{Z}_p[X]$ s.t. $F_y(0) = y$. Let R_y denote the random free term of $D_y(X)$. The dealer sends a pair $(y^j, R_y^j) := (F_y(j), D_y(j))$ to party P_j via a private channel. Subsequently, it broadcasts $EY^k := G^{a_k} H^{b_k}$ (a Pedersen commitment) for $k = 0, \dots, t$ where a_k and b_k are the k -degree coefficients of $F_y(X)$ and $D_y(X)$ respectively. Given EY^k , correctness of a share (y^j, R_y^j) can be verified by checking $G^{y^j} + H^{R_y^j} = \prod_{k=0}^t EY^{k^t}$. Hereinafter, we denote the execution of this verifiable secret sharing protocol by $\text{VSS}(y, R_y)[G, H] \xrightarrow{F_y, D_y} (y^j, R_y^j)[EY^0, EY^1, \dots, EY^t]$.

3 Threshold Mercurial Signatures

We follow the notation from [CKP⁺23] to present the syntax and security properties.

Definition 7 (Threshold mercurial signature). *A TMS scheme is a MS scheme where KGen and Sign , are replaced with:*

$\text{TKGen}(\text{pp}, \ell, t, n) \rightarrow (\vec{\text{sk}}, \vec{\text{pk}}, \text{pk})$: *On input the public parameters pp , a length parameter ℓ , and two integers $t, n \in \text{poly}(1^\kappa)$ such that $1 \leq t \leq n$, this PPT algorithm outputs two vectors of size n of signing and public keys along with the global (threshold) public key pk . Both, the signing keys $\vec{\text{sk}} = (\text{sk}_1, \dots, \text{sk}_n)$ and the public keys $\vec{\text{pk}} = (\text{pk}_1, \dots, \text{pk}_n)$ are distributed among parties such that party P_i gets $(\text{sk}_i, \vec{\text{pk}}, \text{pk})$. The message space \mathcal{M} is well-defined from pp and ℓ .*
 $\text{TSign}(\text{pp}, \{\text{sk}_j\}_{j \in \mathcal{T}}, M) \rightarrow \sigma$: *On input $\{\text{sk}_j\}_{j \in \mathcal{T}}$ for some $\mathcal{T} \subseteq [n], |\mathcal{T}| \geq t$ and a message $M \in \mathcal{M}$, this PPT algorithm is run interactively by a set of parties in \mathcal{T} . At the end, they either abort or output a signature σ .*

We also consider threshold key converter versions for shared keys (ConvertTPK and ConvertTSK) that are analogous to ConvertPK and ConvertSK (now acting on the global keys). For convenience, we include an algorithm SimTKGen that given a key pair (pk, sk) , on input pk , n and a subset of corrupted parties $\mathcal{C} \subsetneq [n]$ of size $t - 1$, outputs \mathcal{C} shares for sk and n shares of pk according to Definition 8.

Security Properties. A public key of our TMS consists of independent random group elements, and distributed TKGen can be instantiated with a DKG protocol. Since many DKG protocols are available with various security properties, e.g., [Fel87, Ped91, AF04, CL24a] (we also refer to a recent survey on the history and state of the art on DKG [Kat23]), we consider construction of distributed TKGen an independent topic and consider TKGen being done by a single trusted party throughout the paper. Nevertheless, we state the security of TKGen required in this work as follows.

Definition 8 (Security of key generation). *TKGen is secure if it outputs pk with the same distribution as KGen does, and there exists a simulator, SimTKGen that, for any sufficiently large κ , any $\text{pp} \in \text{PGen}(1^\kappa)$, $\ell \in \mathbb{N}$, $(\text{pk}, \text{sk}) \in \text{KGen}(\text{pp}, \ell)$, $t, n \in \mathbb{N}$, $\mathcal{C} \subsetneq [n]$ of size $t - 1$, $\text{SimTKGen}(\text{pk}, n, \mathcal{C})$ outputs $\{\text{sk}_j\}_{j \in \mathcal{C}}$ and $\{\text{pk}_j\}_{j \in [n]}$. The joint distribution of $(\text{pk}, \{\text{pk}_j\}_{j \in [n]}, \{\text{sk}_j\}_{j \in \mathcal{C}})$ is indistinguishable from that of $\text{TKGen}(\text{pp}, \ell, t, n)$.*

Correctness of TMS follows the usual notion for MS. Below we present the corresponding definition for completeness.

Experiment $\mathbf{Exp}_{\text{TMS}, \ell, t, n}^{\text{UNF}}(1^\kappa, \mathcal{A})$	Steps (SK, M)
$\text{st} \leftarrow \emptyset; \Sigma \leftarrow \emptyset; \text{pp} \leftarrow \text{PGen}(1^\kappa); (\mathcal{C}, \text{st}) \leftarrow \mathcal{A}(\text{st}, \text{pp})$ if $\mathcal{C} \not\subseteq [n] \vee \mathcal{C} > t - 1$ return \perp $\mathcal{H} \leftarrow [n] \setminus \mathcal{C}; (\vec{\text{sk}}, \vec{\text{pk}}, \text{pk}) \leftarrow \text{TKGen}(\text{pp}, \ell, t, n)$ $(M^*, \sigma^*, \rho^*) \leftarrow \mathcal{A}^{\text{OTSign}(\text{sk}, \cdot)}(\text{st}, \{\text{sk}_i\}_{i \in \mathcal{C}}, \vec{\text{pk}}, \text{pk})$ return $\forall M \in \Sigma : [M]_{\mathcal{R}_M} \neq [M^*]_{\mathcal{R}_M}$ $\wedge \text{Verify}(M^*, \sigma^*, \text{ConvertPK}(\text{pk}, \rho^*)) = 1$	$\text{steps} \leftarrow \emptyset$ foreach $j \in [3t - 2]$ do if $\phi(j) \in \mathcal{C}$ then $(\text{st}, \text{steps}) \leftarrow \mathcal{A}(\text{st}, \text{steps})$ else $\text{steps} \leftarrow \text{TSign}^j(\text{SK}, M; \text{steps})$ $\text{st} \leftarrow \mathcal{A}(\text{st}, \text{steps})$ return steps
Oracle $\text{OTSign}(\text{sk}, M, \mathcal{T})$	
if $ \mathcal{T} \neq t \vee \mathcal{T} \cap \mathcal{H} = \emptyset$ return \perp $\sigma \leftarrow \text{Steps}(\{\vec{\text{sk}}_j\}_{j \in \mathcal{T}}, M); \Sigma \leftarrow \Sigma \cup \{M\};$ return σ	

Fig. 1. Unforgeability experiment. \mathcal{C} and \mathcal{H} are the sets of corrupt and honest signers.

Definition 9 (Correctness). A (n, t) -TMS is correct if it satisfies the following conditions for all $\kappa, \text{pp} \in \text{PGen}(1^\kappa), \ell > 1, (\vec{\text{sk}}, \vec{\text{pk}}, \text{pk}) \in \text{TKGen}(\text{pp}, \ell, t, n)$ and $\mathcal{T} \subseteq [n]$ of size $\geq t$:

- *Verification.* $\forall M \in \mathcal{M}, \forall \sigma \in \text{TSign}(\text{pp}, \{(j, \text{sk}_j)\}_{j \in \mathcal{T}}, M),$
 $\text{Verify}(\text{pp}, M, \sigma, \text{pk}) = 1.$
- *Key conversion.* $\forall \rho \in \text{sample}_\rho, (\text{ConvertTSK}(\text{sk}, \rho), \text{ConvertTPK}(\text{pk}, \rho),$
 $\text{ConvertPK}(\text{pk}, \rho)) \in \text{TKGen}(\text{pp}, \ell, t, n).$
- *Signature conversion.* $\forall M \in \mathcal{M}, \forall \sigma$ such that $\text{Verify}(\text{pp}, \text{pk}, M, \sigma) = 1, \forall \rho \in \text{sample}_\rho, \forall \sigma' \in$
 $\text{ConvertSig}(\text{pk}, M, \sigma, \rho), \text{Verify}(\text{ConvertPK}(\text{pk}, \rho), M, \sigma') = 1.$
- *Change of message representative.* $\forall M \in \mathcal{M}, \forall \sigma$ such that $\text{Verify}(\text{pp}, M, \sigma, \text{pk}) = 1, \forall \mu \in \text{sample}_\mu,$
 $\text{Verify}(\text{pp}, M', \sigma', \text{pk}) = 1,$ where $(M', \sigma') = \text{ChgRep}(\text{pk}, M, \sigma, \mu).$ Moreover, $M' \in [M]_{\mathcal{R}_M}.$

For unforgeability and unlinkability (class-hiding) we follow the definitions from [CL19, CKP⁺23], adapting them to the threshold setting (the adversary can corrupt up to $t - 1$ parties). To that end, we consider a signing oracle OTSign that internally executes TSign as a step function in a sequentially interactive protocol where computations are executed by a party while obtaining input from the previous and passing the output to the next one. Consequently, TSign can be seen as a sequence of step functions TSign^j for $j \in [k]$ where k is the total number of steps of the protocol, *i.e.*, $k = 3(t - 2) + 2 \times 2 = 3t - 2$. We define a function $\phi : [k] \rightarrow [n]$ that maps the index of a step to the id of the party executing the step. On receiving a query (M, \mathcal{T}) , oracle OTSign executes TSign^j for $j = 1$ to k in sequence. If party $\phi(j)$ is corrupted, OTSign consults adversary \mathcal{A} and obtains the output of TSign^j . Otherwise, OTSign executes TSign^j and sends the output to \mathcal{A} . OTSign finally outputs σ that TSign^k outputs. For ease of exposition, we define an auxiliary internal procedure ($\text{Steps}()$) in OTSign to capture it.

Definition 10 (Unforgeability). A TMS scheme is unforgeable if the advantage of any PPT adversary \mathcal{A} defined by $\mathbf{Adv}_{\text{TMS}, \ell, t, n}^{\text{UNF}}(1^\kappa, \mathcal{A}) := \Pr[\mathbf{Exp}_{\text{TMS}, \ell, t, n}^{\text{UNF}}(1^\kappa, \mathcal{A}) \Rightarrow \text{true}] \leq \epsilon(\kappa),$ where $\mathbf{Exp}_{\text{TMS}, \ell, t, n}^{\text{UNF}}(1^\kappa, \mathcal{A})$ is shown in Fig. 1.

Unlike the original class-hiding definition for mercurial signatures (Definition 4), we aim to capture a stronger definition in which the adversary is given access to the challenge public keys and shares of the corresponding secret keys associated to corrupted parties for one of them. Our approach is to consider a scenario under *key leakage* where the adversary gets to know a subset of the secret key shares, similar to the class-hiding definition from [BHKS18]. Our notion is in-between the class-hiding notion from [CL19] that only considers honestly generated keys with no key leakage and the one from [BHKS18] (which is strictly weaker than $(\cdot, 1, 3)$ -UNL from [CGH⁺23]). We refer to it as *public key unlinkability* to make a distinction. For $n = t = 1$, our definition seamlessly gives the notion of public key unlinkability for MS, implied by public key class-hiding and fulfilled by the instantiation of MS in [CL19]. We will use these facts to prove public key unlinkability.

Definition 11 (Public Key Unlinkability). A TMS scheme is public key unlinkable if the advantage of any PPT adversary \mathcal{A} defined by $\mathbf{Adv}_{\text{TMS}, \ell, t, n}^{\text{PK-UNL}}(1^\kappa, \mathcal{A}) := 2 \cdot \Pr[\mathbf{Exp}_{\text{TMS}, \ell, t, n}^{\text{PK-UNL}}(1^\kappa, \mathcal{A}) \Rightarrow \text{true}] - 1 \leq \epsilon(\kappa),$ where $\mathbf{Exp}_{\text{TMS}, \ell, t, n}^{\text{PK-UNL}}(1^\kappa, \mathcal{A})$ is shown in Fig. 2.

Experiment $\text{Exp}_{\text{TMS}, \ell, t, n}^{\text{PK-UNL}}(1^\kappa, \mathcal{A})$

$\text{st} \leftarrow \emptyset; b \leftarrow_{\$} \{0, 1\}; \rho \leftarrow_{\$} \text{sample}_\rho; \text{pp} \leftarrow_{\$} \text{PGen}(1^\kappa); (\mathcal{C}, \text{st}) \leftarrow \mathcal{A}(\text{st}, \text{pp});$ **if** $\mathcal{C} \notin [n] \vee |\mathcal{C}| > t - 1$ **return** \perp
 $(\vec{\text{sk}}_1, \vec{\text{pk}}_1, \text{pk}_1) \leftarrow_{\$} \text{TKGen}(\text{pp}, \ell, t, n); (\vec{\text{sk}}_2^0, \vec{\text{pk}}_2^0, \text{pk}_2^0) \leftarrow_{\$} \text{TKGen}(\text{pp}, \ell, t, n)$
 $\text{pk}_2^1 \leftarrow \text{ConvertPK}(\text{pk}_1, \rho); \vec{\text{sk}}_2^1 \leftarrow \text{ConvertSK}(\vec{\text{sk}}_1, \rho)$
 $b' \leftarrow_{\$} \mathcal{A}^{\text{OTSign}(\{\vec{\text{sk}}^{(j)}\}_{j \in \mathcal{T}, \cdot, \cdot})}(\text{st}, \{\vec{\text{sk}}_1^{(j)}\}_{j \in \mathcal{C}}, \text{pk}_1, \text{pk}_2^b);$ **return** $b = b'$

Oracle $\text{OTSign}(M, \text{pk}, \mathcal{T})$

if $|\mathcal{T}| \neq t$ **return** $\perp; \Sigma \leftarrow \Sigma \cup \{M\}$
if $\text{pk} = \text{pk}_2^b$ **return** $\text{Sign}(\vec{\text{sk}}_2^b, M)$ **else if** $\text{pk} = \text{pk}_1$ **return** $\text{Steps}(\{\vec{\text{sk}}^{(j)}\}_{j \in \mathcal{T}}, M)$
 $\text{Steps}(\text{SK}, M)$

$\text{steps} \leftarrow \emptyset$
foreach $j \in [3t - 2]$ **do**
 if $\phi(j) \in \mathcal{C}$ **then** $(\text{st}, \text{steps}) \leftarrow \mathcal{A}(\text{st}, \text{steps})$ **else** $\text{steps} \leftarrow \text{TSign}^j(\text{SK}, M; \text{steps}); \text{st} \leftarrow \mathcal{A}(\text{st}, \text{steps})$
return steps

Fig. 2. Public key unlinkability experiment.

Remark 12. We stress that the above definition serves two purposes. First, it provides backward compatibility with the original MS formalization, assuming the usual single-party signing process where the adversary cannot corrupt a party and thus is not given any secret key shares. Secondly, as discussed in the context of anonymous credentials (Sec. 4.4), when a signature is distributively computed, no adversary can distinguish if an adapted signature that verifies under an adapted public key is related or not to the original public key, even if it knows a subset of shares for the corresponding secret key. Before, since the adversary knew the full secret key, it was trivial to distinguish an adapted public key from one from a different equivalence class.

4 Two-Party Case

We present the two-party case as a (2-2)-TMS scheme. This decision will become clearer when we discuss the applications of this setting.

Our approach to building a (2-2)-TMS is to modify the scheme from [CL19] so that the signing protocol runs interactively between two parties. Intuitively, a signature that verifies under a jointly computed public key is obtained at the end. The resulting signature has the same structure as the one from [CL19] and it can work as a drop-in replacement. In particular, we assume one honest party for public key unlinkability (if they collude they can recognize the public key).

4.1 Construction 1

We assume that $\text{PGen}(1^\kappa)$ and $\text{TKGen}(\text{pp}, \ell, 2, 2)$ are run honestly. Every signer j is given $\text{pp} = (p, \mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T, G, \hat{G}, e)$, $\vec{\text{pk}} := \{\hat{G}^{x_j^i}\}_{j \in [0, 1], i \in [\ell]}$, pk , and $\vec{\text{sk}}_j := \{x_j^i\}_{i \in [\ell]}$. The (global) signing key x_i for $i \in [\ell]$ is implicitly set to $x_i := x_0^i + x_1^i \in \mathbb{Z}_p$.

In Fig. 3, we present our main protocol for instantiating TSign . The protocol's goal is to compute (Z, Y, \hat{Y}) for a given message M . It consists of two parts; one to compute Y and \hat{Y} , and another to compute Z . Below we give an intuition and subsequently discuss the technical details required to prove security.

Computing $Y = G^{\frac{1}{y}}$ and $\hat{Y} = \hat{G}^{\frac{1}{y}}$ for $y = y_0 y_1$ is done straightforwardly in sequence. Computing $Z = (\prod M_i^{x_0^i + x_1^i})^y$ for $i \in [\ell]$ could be done first by computing $Z_1 = \prod M_i^{x_1^i}$ at signer P_1 , then $Z_0 = Z_1 \prod M_i^{x_0^i}$ at signer P_0 , and finally $(Z_0)^{y_0 y_1}$ with y_0 and y_1 in sequence. However, Z_1 is computed deterministically, requiring full knowledge about P_1 's signing key, while we have to simulate P_1 without knowing the signing keys for the case where P_0 is corrupted.

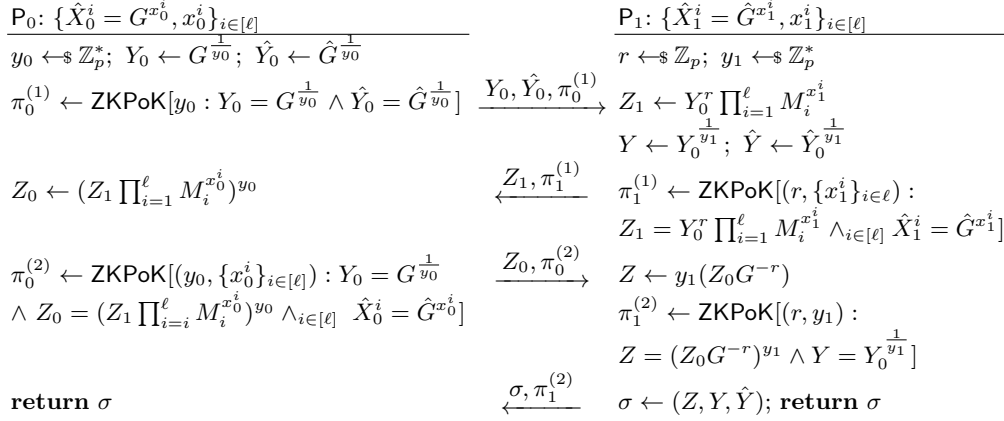


Fig. 3. TSign(pp, $\{x_j^i\}_{j \in \{0,1\}, i \in [\ell]}, M$)

Our approach to getting around the above problem is to blind Z_1 by using $Y_0 = G^{\frac{1}{y_0}}$, obtained in the first part of the protocol as the basis of a blinding factor. Computing $Z_1 = Y_0^r \prod M_i^{x_1^i}$ with random r perfectly blinds it. Once P_0 computes $Z_0 = (Z_1 \prod M_i^{x_0^i})^{y_0}$, factor Y_0 in Z_1 is cancelled out since $(Y_0^r)^{y_0} = (G^{\frac{r}{y_0}})^{y_0} = G^r$. Thus, P_1 , who holds r , can easily unblind Z_0 by multiplying G^{-r} . This blinding of Z_1 causes another problem in the opposite case where P_1 is corrupted; It makes it hard for the simulator to control the resulting signature. We address it by extracting the randomization factor r from the zero-knowledge proof of well-formedness of blinded Z_1 . Since the unblinding is deterministic with respect to r , the simulator knowing r can embed an intended signature to Z_0 .

As previously mentioned, we require zero-knowledge proofs to prove the right computation of the values sent by each party. In particular, we require knowledge soundness of $\pi_1^{(1)}$ and zero-knowledge of $\pi_0^{(1)}$ and $\pi_0^{(2)}$ to simulate P_0 . Analogously, to simulate P_1 . Below we discuss how each ZKPoK can be implemented.

- $\pi_0^{(1)} := \text{ZKPoK}[y_0 : Y_0 = G^{1/y_0} \wedge \hat{Y}_0 = \hat{G}^{1/y_0}]$: This can be done with the standard Chaum-Pedersen protocol [CP93] with witness $1/y_0$ instead of y_0 .
- $\pi_1^{(1)} := \text{ZKPoK}[(r, \{x_1^i\}_{i \in [\ell]}) : Z_1 = Y_0^r \prod_{i=1}^{\ell} M_i^{x_1^i} \wedge_{i \in [\ell]} \hat{X}_1^i = \hat{G}^{x_1^i}]$: as above.
- $\pi_0^{(2)} := \text{ZKPoK}[(y_0, \{x_0^i\}_{i \in [\ell]}) : Z_0 = (Z_1 \prod_{i=1}^{\ell} M_i^{x_0^i})^{y_0} \wedge G = Y_0^{y_0} \wedge_{i \in [\ell]} \hat{X}_0^i = \hat{G}^{x_0^i}]$: The first clause involves a witness product in the exponent. The proof must not expose intermediate value $Z_1 \prod_{i=1}^{\ell} M_i^{x_0^i}$ to the verifier. We thus translate the statement to the following equivalent one:

$$\pi_0^{(2)} := \text{ZKPoK} \left[\left((\{x_0^i\}_{i \in [\ell]}, 1/y_0) : Z_1 = Z_0^{1/y_0} \prod_{i=1}^{\ell} M_i^{-x_0^i} \wedge Y_0 = G^{1/y_0} \wedge_{i \in [\ell]} \hat{X}_0^i = \hat{G}^{x_0^i} \right) \right]$$

- $\pi_1^{(2)} := \text{ZKPoK}[(r, y_1) : Z = (Z_0 G^{-r})^{y_1} \wedge Y = Y_0^{1/y_1}]$: This statement involves a witness product as well, and is translated into $\pi_1^{(2)} := \text{ZKPoK}[(1/y_1, r) : Z_0 = Z^{1/y_1} G^r \wedge Y = Y_0^{1/y_1}]$. It is worth noting that r is not guaranteed to be the same as the one used in $\pi_1^{(1)}$ since the final output is accepted if it verifies as a signature. Nevertheless, $\pi_1^{(2)}$ is needed for the proper link between the resulting signature and the inputs from honest P_0 .

Each proof is verified by the respective recipient. The same for σ . If any verification fails, the party aborts and TSign outputs \perp .

4.2 Efficiency

Except for ZKPoK's, computation and communication complexity at each party are the same as those for the original MS. Party P_1 has three extra exponentiations in \mathbb{G} for blinding and unblinding.

Table 1 presents the computational and communication costs of each ZKPoK when instantiating them using sigma protocols. Computation costs count the number of exponentiations in the respective

Proof	Computation		Proof Size
	Prover	Verifier	
$\pi_0^{(1)}$	$1 \mathbb{G} + 1 \hat{\mathbb{G}} $	$2 \mathbb{G} + 2 \hat{\mathbb{G}} $	$1 \mathbb{G} + 1 \hat{\mathbb{G}} + 2 \mathbb{Z}_p $
$\pi_1^{(1)}$	$(\ell + 1) \mathbb{G} + \ell \hat{\mathbb{G}} $	$(\ell + 2) \mathbb{G} + 2\ell \hat{\mathbb{G}} $	$1 \mathbb{G} + \ell \hat{\mathbb{G}} + (\ell + 1) \mathbb{Z}_p $
$\pi_0^{(2)}$	$(\ell + 2) \mathbb{G} + \ell \hat{\mathbb{G}} $	$(\ell + 4) \mathbb{G} + 2\ell \hat{\mathbb{G}} $	$2 \mathbb{G} + \ell \hat{\mathbb{G}} + (\ell + 1) \mathbb{Z}_p $
$\pi_1^{(2)}$	$3 \mathbb{G} $	$5 \mathbb{G} $	$2 \mathbb{G} + 3 \mathbb{Z}_p $

Table 1. Costs of ZKPoK protocols. Computation counts number of exponentiations in the relevant groups without optimization for multi-base exponentiations.

groups, and the proof size is in terms of scalar values and group elements. We defer the full presentation of each protocol to Appendix A. We note that ℓ is usually instantiated for short vectors. Considering the applications, $\ell = 2$ for blind signatures, $\ell = 3$ for the basic attribute-based credential scheme from [FHS19], $\ell = 5$ considering revocation [DHS15], and $\ell = 7$ for adding auditability [CDLP22].

4.3 Security

Theorem 13 (Correctness). *Construction 1 is correct.*

Proof. We show that our (2, 2)-TMS scheme produces a signature (Z, Y, \hat{Y}) that distributes the same as $\text{Sign}(\text{pp}, \text{sk}, M)$ for $\text{sk} := (x_i)_{i \in [\ell]} = (x_0^i + x_1^i)_{i \in [\ell]}$ if both P_0 and P_1 are honest. Observe that $Y = G^{\frac{1}{y_0 y_1}}$, $\hat{Y} = \hat{G}^{\frac{1}{y_0 y_1}}$, and

$$\begin{aligned}
Z &= (Z_0 G^{-r})^{y_1} = \{(Z_1 \cdot \prod_{i=1}^{\ell} M_i^{x_0^i})^{y_0} G^{-r}\}^{y_1} \\
&= \{(Y_0^r \prod_{i=1}^{\ell} M_i^{x_1^i} \cdot \prod_{i=1}^{\ell} M_i^{x_0^i})^{y_0} G^{-r}\}^{y_1} \\
&= \left\{ \left(G^{\frac{1}{y_0} r} \prod_{i=1}^{\ell} M_i^{(x_0^i + x_1^i)} \right)^{y_0} G^{-r} \right\}^{y_1} \\
&= \left\{ \left(G^{\frac{1}{y_0} r} \right)^{y_0} G^{-r} \left(\prod_{i=1}^{\ell} M_i^{(x_0^i + x_1^i)} \right)^{y_0} \right\}^{y_1} \\
&= \left(\prod_{i=1}^{\ell} M_i^{(x_0^i + x_1^i)} \right)^{y_0 y_1}
\end{aligned}$$

hold. Thus, for $x_i = x_0^i + x_1^i$ and $y = y_0 y_1$, the resulting signature is $(Z, Y, \hat{Y}) = \left(\left(\prod_{i=1}^{\ell} M_i^{x_i} \right)^y, G^{1/y}, \hat{G}^{1/y} \right)$. Since y_0 and y_1 are uniformly taken from \mathbb{Z}_p^* , $y = y_0 y_1$ distributes uniformly over \mathbb{Z}_p^* . Accordingly, the signature distributes identically as the original, *i.e.*, single-signer MS. Consequently, correctness of key conversion, signature conversion and change of message representative follow from correctness of the original MS. \square

Unforgeability is considered for static corruptions in the stand-alone execution model. We first explain our proof strategy and key technical points. In mercurial signatures [CL19], unforgeability (Definition 3) is proved by contradiction. If there were a PPT algorithm that could break unforgeability through accessing the signing oracle, one could construct a reduction breaking the unforgeability of the base SPS-EQ[FHS19].

For TMS, the security assurance of unforgeability slightly alters the one from [CL19]. Since we have an interactive signing protocol, we must prove that the adversary's advantage when interacting with TSign is no greater than its advantage in the original unforgeability game. Moreover, we give strong power to the adversary allowing it to run the (interactive) signing oracle on behalf of any corrupted party of its choice instead of just leaking the key share of its choice. Hence, care should be taken when instantiating the signing oracle from Definition 10 as it can be run between the adversary and the environment. We have three cases:

1. \mathcal{A} calls the oracle for two honest parties (honest signing). In this case, the environment runs the honest protocol, and it is easy to see that the \mathcal{A} 's advantage is the same as in the original game due to the zero-knowledge property of the ZKPoK's and the fact that the signatures computed by the interactive protocol are identically distributed as the signatures from [CL19].

$$\begin{array}{ccc}
\underline{P_0: \text{sk}_0, \text{pk}_0, \text{pk}_1, M \text{ (corr.)}} & & \underline{P_1: \text{pk}_0, \text{pk}_1, M \text{ (sim. with Sign(sk, \cdot))}} \\
& & (Z', Y', \hat{Y}') \leftarrow \text{Sign}(\text{sk}, M) \\
(Y_0, \hat{Y}_0, \pi_0^{(1)}) \leftarrow \mathcal{A}(\text{st}) & \xrightarrow{Y_0, \hat{Y}_0, \pi_0^{(1)}} & Z_1 \leftarrow \mathbb{G}; Y \leftarrow Y'; \hat{Y} \leftarrow \hat{Y}' \\
& & \xleftarrow{Z_1, \pi_1^{(1)}} \pi_1^{(1)} \leftarrow \text{ZKPoK.Sim}(Z_1, Y_0, M) \\
(Z_0, \pi_0^{(2)}) \leftarrow \mathcal{A}(\text{st}, Z_1, \pi_1^{(1)}) & \xrightarrow{Z_0, \pi_0^{(2)}} & Z \leftarrow Z' \\
& & \xleftarrow{Z_0, \pi_0^{(2)}} \pi_1^{(2)} \leftarrow \text{ZKPoK.Sim}(Z, Z_0, Y, Y_0) \\
\text{return } (\sigma, \pi_1^{(2)}) & \xleftarrow{\sigma, \pi_1^{(2)}} & \sigma \leftarrow (Z, Y, \hat{Y}); \text{return } (\sigma, \pi_1^{(2)})
\end{array}$$

Fig. 4. Simulator's algorithm considering corruption of P_0 .

$$\begin{array}{ccc}
\underline{P_0: \text{pk}_0, \text{pk}_1, M \text{ (sim. with Sign(sk, \cdot))}} & & \underline{P_1: \text{sk}_1, \text{pk}_0, \text{pk}_1, M \text{ (corr.)}} \\
& & (Z', Y', \hat{Y}') \leftarrow \text{Sign}(\text{sk}, M) \\
& & Y_0 \leftarrow Y'; \hat{Y}_0 \leftarrow \hat{Y}' \\
\pi_0^{(1)} \leftarrow \text{ZKPoK.Sim}(Y_0, \hat{Y}_0) & \xrightarrow{Y_0, \hat{Y}_0, \pi_0^{(1)}} & (Z_1, \pi_1^{(1)}) \leftarrow \mathcal{A}(\text{st}, Y_0, \hat{Y}_0, \pi_0^{(1)}) \\
& & \xleftarrow{Z_1, \pi_1^{(1)}} \\
r \leftarrow \text{ZKPoK.Ext}(\pi_1^{(1)}); Z_0 \leftarrow Z' G^r & & \\
\pi_0^{(2)} \leftarrow \text{ZKPoK.Sim}(Z_0, Z_1, M, Y_0) & \xrightarrow{Z_0, \pi_0^{(2)}} & (\sigma, \pi_1^{(2)}) \leftarrow \mathcal{A}(\text{st}, Z_0, \pi_0^{(2)}) \\
& & \xleftarrow{\sigma, \pi_1^{(2)}} \text{return } (\sigma, \pi_1^{(2)}) \\
\text{return } (\sigma, \pi_1^{(2)}) & &
\end{array}$$

Fig. 5. Simulator's algorithm considering corruption of P_1 .

2. \mathcal{A} controls P_0 . We simulate P_1 so that it ignores inputs from P_0 and outputs signatures following the same distribution as in the original game.
3. \mathcal{A} controls P_1 . We simulate P_0 based on P_1 's knowledge extraction.

In either case, we show that the joint view of the adversary and the corrupted party is essentially the same as that of the adversary in the original unforgeability game of MS due to the security of the zero-knowledge proofs involved.

Theorem 14 (Unforgeability). *Construction 1 is unforgeable against static corruption of at most one party if TKGen is secure, all ZKPoK's are secure, and the original MS is unforgeable.*

Proof. Given access to adversary \mathcal{A} playing the unforgeability game against TMS as in Figure 1, we construct a simulator that plays the role of the adversary in the unforgeability game against MS as in Definition 3. Let $(\text{sk}, \text{pk}) := (\{x^i\}_{i \in [\ell]}, \{\hat{X}^i\}_{i \in [\ell]})$ be a key pair of MS generated by KGen(pp, ℓ). Given pp as input, the simulator first invokes \mathcal{A} and outputs \mathcal{C} obtained from \mathcal{A} . Here, \mathcal{C} is either 0 or 1 meaning P_0 or P_1 is corrupted, respectively. Then, given pk as input, the simulator executes SimTKGen(pk, 2, \mathcal{C}) to obtain $\text{sk}_j := \{x_j^i\}_{i \in [\ell]}$ for $j \in \mathcal{C}$ and $\text{pk}_j := \{\hat{X}_j^i\}_{i \in [\ell]}$ for $j \in [n]$. Shared signing keys $\{x_j^i\}_{i \in [\ell]}$ for $j \notin \mathcal{C}$ are not given to the simulator but implicitly set so that $x_0^i + x_1^i = x^i$ holds. The simulator then invokes \mathcal{A} with $\{\text{sk}_j\}_{j \in \mathcal{C}}$, $\{\text{pk}_j\}_{j \in [n]}$, and pk as input.

Recall that \mathcal{A} is allowed to make signing queries to OTSign that internally executes TSign in the presence of a corrupted party. Thus, the simulator has to simulate the honest party in TSign. Whenever \mathcal{A} queries message M to OTSign, the simulator forwards M to signing oracle Sign of MS and obtains signature (Z', Y', \hat{Y}') . From here, the simulator works along with the possible corruption scenarios. The first case considers corruption of P_0 , as shown in Fig. 4. The case in which P_1 is corrupted is shown in Fig. 5.

We show that, for both cases of corruption, the honest party can be simulated indistinguishably from the real execution of the corresponding algorithm in TSign. For the first case (Fig. 4), the real computation of Z_1 and the simulated one in the first round are perfectly indistinguishable because the real one includes a uniformly random factor and the simulated one is chosen uniformly. It implicitly determines random factor $r := \log_{Y_0} Z_1 (\prod_{i=1}^{\ell} M_i^{x_1^i})^{-1}$ for x_1^i also implicitly determined by \hat{X}_1^i . Furthermore, the quality of simulated $\pi_1^{(1)}$ is due to its zero-knowledge property. Moving into the second round, we claim that P_0 cannot distinguish the difference between the original computation of σ and

the simulated one provided that both $\pi_0^{(1)}$ and $\pi_0^{(2)}$ are sound. Observe that the proper computation of (Z, Y, \hat{Y}) is deterministic from Z_0 , r and y_1 implicitly determined by $y_1 = \log_Y Y_0 = \log_{\hat{Y}} \hat{Y}_0$. Therefore, if $\pi_0^{(1)}$ and $\pi_0^{(2)}$ are sound and **Sign** is correct, the simulated (Z, Y, \hat{Y}) distributes perfectly in the same way as the original one does. The quality of simulated $\pi_1^{(2)}$ is due to its zero-knowledge property, hiding how Z was computed.

We now consider corruption of P_1 (Fig. 5). During the first round, Y_0 and \hat{Y}_0 are distributed identically as their original computation and $\pi_0^{(1)}$ is zero-knowledge. Looking at the second round, we claim that Z_0 is perfectly simulated if $\pi_1^{(1)}$ is knowledge sound. That is, the knowledge soundness of $\pi_1^{(1)}$ assures that Z_1 has been correctly computed as $Z_1 = Y_0^r \prod_{i=1}^{\ell} M_i^{x_i}$ with the extracted r . Thus, for $Z' = (\prod_{i=1}^{\ell} M_i^{x_i})^y$ where $x_i = (x_0^i + x_1^i)$ for $i \in [\ell]$ and $y = y_0 = \log_G Y'$, we have: $Z_0 = Z' G^r = (\prod_{i=1}^{\ell} M_i^{x_0^i + x_1^i})^{y_0} G^r = (Y_0^r \prod_{i=1}^{\ell} M_i^{x_1^i})^{y_0} (\prod_{i=1}^{\ell} M_i^{x_0^i})^{y_0} = (Z_1 \prod_{i=1}^{\ell} M_i^{x_0^i})^{y_0}$. Accordingly, the simulation of P_0 is perfect modulo the knowledge soundness of $\pi_1^{(1)}$ and zero-knowledge of $\pi_0^{(1)}$ and $\pi_0^{(2)}$.

The simulator outputs whatever \mathcal{A} outputs. As **TKGen** is assumed secure and the honest party within **OTSign** is correctly simulated, the view of $\mathcal{A}^{\text{OTSign}}$ is indistinguishable from that of the real unforgeability game in the presence of corrupt party \mathcal{C} . Thus, whenever \mathcal{A} is successful in forging **TMS**, so is the simulator in forging **MS**. Finally, we evaluate the advantage based on the error bounds for the sub-procedures. Let maximum error bounds for the zero-knowledge part be: \mathcal{E}_{Snd} and \mathcal{E}_{ZK} for the soundness and zero-knowledge of each proof, respectively, and $\mathcal{E}_{\text{KSnd}}$ for the knowledge soundness of $\pi_1^{(1)}$. Also let \mathcal{E}_{TKG} denote the error bound for **SimTKGen**. We obtain that the adversary's advantage when executing q queries to the signing oracle is given by $\text{Adv}_{\text{TMS}, \ell, 2, 2}^{\text{UNF}}(1^\kappa, \mathcal{A}) < \text{Adv}_{\text{MS}, \ell}^{\text{UNF}}(1^\kappa, \mathcal{A}) + q \cdot (2\mathcal{E}_{\text{Snd}} + 2\mathcal{E}_{\text{ZK}} + \mathcal{E}_{\text{KSnd}}) + \mathcal{E}_{\text{TKG}}$. \square

We prove unlinkability assuming at least one honest signer and consider a signing oracle in the presence of the corrupted party (\mathcal{A} can corrupt any party of its choice).

Theorem 15 (Public Key Unlinkability). *Construction 1 is public key unlinkable against static corruption of at most one party if **TKGen** is secure, all **ZKPoK**'s are secure, and **MS** is origin and public key class-hiding.*

Proof. The proof strategy considers a similar simulator from Theorem 14. Given access to adversary \mathcal{A} playing the unlinkability game against **TMS**, we construct a simulator that plays the role of the adversary in the unlinkability game against **MS** (public key class-hiding). Given pp as input, the simulator invokes \mathcal{A} and outputs \mathcal{C} obtained from \mathcal{A} . Then, given $(\text{pk}_1, \text{pk}_2^b)$ as input, the simulator runs **SimTKGen** for pk_1 with \mathcal{C} to obtain, for $\text{sk}_1^{(j)}$ for $j \in \mathcal{C}$ and $\text{pk}_1^{(j)}$ for $j \in [n]$. The simulator then invokes \mathcal{A} with $\{\text{sk}_1^{(j)}\}_{j \in \mathcal{C}}$, $\{\text{pk}_1^{(j)}\}_{j \in [n]}$, and pk_2^b as input. The validity of the simulation up to this point is due to the security of **TKGen**.

On receiving a query from \mathcal{A} to **OTSign** on pk_2^b and M , the simulator forwards it to its oracle and returns the response to \mathcal{A} . This part of the simulation is perfect due to the origin-hiding property of **MS**.

On receiving a query from \mathcal{A} to **OTSign** on pk_1 and M , the simulator forwards it to its oracle to obtain an **MS** signature for M . Subsequently, it uses the obtained signature to simulate an invocation of **TSign** (using **Steps**) on pk_1 with M as explained in the proof of Theorem 14. Validity of this part of the simulation is due to the security of **ZKPoK**'s as before. If **Steps**'s simulation results in \perp (due to misbehaviour of a corrupted party), it returns \perp . Finally, the simulator outputs b' that \mathcal{A} outputs. Since the view of \mathcal{A} is correctly simulated, the output is correct whenever \mathcal{A} wins the game against **TMS**. This concludes the proof. \square

4.4 Application to Anonymous Credentials

Attribute-based anonymous credentials (ABC) allow users to authenticate themselves with respect to a set of attributes while hiding their identity. A prominent framework in this setting based on EQS originated with the work by Fuchsbauer, Hanser and Slamanig [FHS19] (hereinafter FHS19). FHS19 provides constant-size credentials (*i.e.*, the credential size as well as the bandwidth required to show it are constant-size irrespectively of the attributes shown) supporting a selective-disclosure of attributes (users can decide which attributes to show each time).

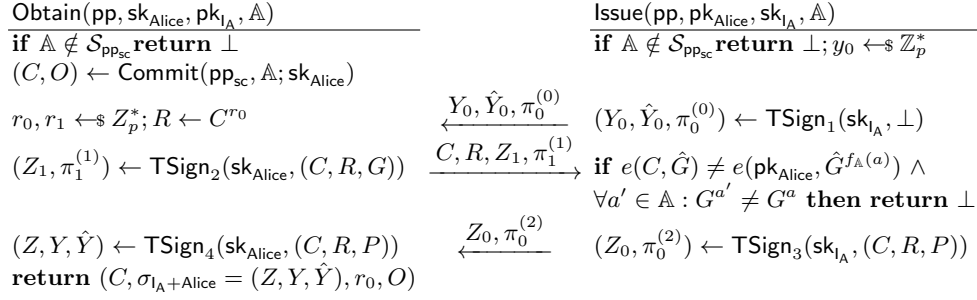


Fig. 6. FHS19's credential issuing protocol instantiated with our TMS.

Subsequent works extended FHS19 to consider revocation [DHS15], auditability [CDLP22], more expressiveness, and issuer-hiding features [BEK⁺21, CLPK22]. In particular, [CDLP22] proposes to use the MS from [CL19] to provide issuer-hiding features based on an OR-Proof for the correct randomization of the issuer's public key relative to a list of authorized issuers. Recall that using MS, users can adapt their signature to a randomized public key. A downside of this approach is that now the showing is linear in the number of issuers because of the OR-Proof. Furthermore, since the MS used only provides a weak issuer-hiding feature (*i.e.*, an issuer can recognize randomizations of its own public key), the ABC from [CDLP22] is limited to settings where partial trust can be tolerated. Put differently, a verifier colluding with an issuer can determine if the authenticated user belongs to the issuer's organization, severely reducing the user's anonymity set.

A closer look at FHS19's framework. The ABC from FHS19 enables anonymous and unlinkable credential showings. In terms of security, *unforgeability* of their credential scheme ensures that users can only authenticate with respect to attributes they possess. Besides, it also prevents any collusion among malicious users to collectively perform valid showings for attributes that none of them hold. *Anonymity* assures users that neither a verifier nor a malicious organization can collude to identify them. Additionally, it ensures that multiple showings of the same credential cannot be linked together (*i.e.*, credentials are multi-show). To realize an efficient ABC, FHS19 uses EQS on set-commitments where the latter primitive is basically an accumulator supporting subset membership proofs. Moreover, its public parameters are given by $(G^{a^i}, G^{a^i})_{i \in [t]}$, where t is the maximum cardinality of an attribute set (to be accumulated) and a is randomly picked and used as the accumulator's evaluation point.

Each user produces a set commitment C to her set of attributes $\mathbb{A} \subset \mathbb{Z}_p$, whose randomness is the user's secret key. $\mathcal{S}_{\text{ppsc}} = \{S \subset \mathbb{Z}_p : 0 < |S| \leq t\}$ defines the set of valid attributes sets and $f_S(X) := \prod_{s \in S} (X - s)$ its polynomial representation, allowing everyone to efficiently compute $G^{f_{\mathbb{A}}(a)}$ using G^{a^i} without knowledge of a itself. Following the original notation, a set commitment scheme includes a `Commit` algorithm that takes as input public parameters pp_{sc} , a set of attributes \mathbb{A} , and the user's secret key as randomness. It returns a commitment C to the user's attributes and opening information O . To obtain a credential, users interact with the issuer in the following way: they compute a commitment C to their attribute set and request a signature on (C, C^{r_0}, G) . The second and third elements in (C, C^{r_0}, G) are required to prove the scheme's unforgeability.

Improving issuer-hiding. Our idea is to replace the signing protocol used in FHS19 to issue credentials with our interactive signing protocol for TMS. To that end, looking at Fig. 3, the user will play the role of P_1 while the issuer will play the role of P_0 . Moreover, since only the user needs to obtain the signature, we can remove the last step from P_1 . In Fig. 6, we show how to instantiate the original issuing protocol from [FHS19] (Fig. 2) considering a user Alice who runs `TSign` with an issuer IA . For simplicity, we split `TSign` into `TSigni` for $i \in \{1, 2, 3, 4\}$ where each i abstracts the computation corresponding to the i -th round. As a result, we obtain a three-round protocol when using non-interactive ZKPoK instantiations.

Unlike the original protocol, in our case Alice obtains a signature $\sigma_{\text{IA}+\text{Alice}}$, which verifies under $\text{pk} = \text{pk}_{\text{IA}} \cdot \text{pk}_{\text{Alice}}$. Subsequently, Alice can produce a showing proof randomizing the signature-message pair with μ (using `ChgRep` as in FHS19), ρ (using `ConvertSig` to hide pk), and giving a NIZK proof for statement $\{(\rho, \text{sk}_{\text{Alice}}) : \text{pk}' = (\text{pk}_{\text{IA}} \cdot \text{pk}_{\text{Alice}})^{\rho}\}$. Such type of NIZK, whose idea we borrow from

[ST24], can be efficiently implemented in the ROM from Schnorr proofs. Intuitively, it attest that Alice generated her credential with the authority and thus the signature is valid. Note that Alice could produce the NIZK proof without having the TMS but that alone is useless. While this approach attest correctness, it is not yet issuer-hiding because it leaks the issuer's public key pk_{i_A} . The user can generate an OR-Proof for the same previous statement for every key in the issuers' set to make it (fully) issuer-hiding. Now, thanks to the public key unlinkability of TMS, issuers cannot link their public key with a randomized one (this was possible in all previous works from EQS[CDLP22]).

Comparison with [ST24]. The recent work by Sanders and Traoré [ST24] provides a strong issuer-hiding notion with different trade-offs compared to our approach. A showing proof for [ST24] will be shorter when the attribute set is small and the use of PS signatures provides richer functionalities in terms of expressiveness (*e.g.*, one can easily prove relations between attributes). However, their showing is linear in the number of attributes encoded in a credential (ours is linear in the number of issuers), and the policies that are defined by each verifier to authorize a set of issuers are also linear in the number of attributes and the number of issuers. Besides, we provide backwards compatibility with all of the prior work under the same framework, obtaining an ABC scheme with all its extensions (some of which are not considered in [ST24]).

5 Threshold Case

This section describes n -party protocols where keys are distributed in a (t, n) -threshold manner among n users P_1, \dots, P_n . Consequently, we focus on threshold signing protocols where a subset of users P_1, \dots, P_t engage to produce a signature. We assume the key generation is done by a single honest party and only describe the syntax and relation satisfied by the keys. Concretely, for given t and n such that $1 \leq t \leq n$, TGen generates local key pairs $\text{sk}_j := (x_j^1, \dots, x_j^\ell)$, $\text{pk}_j := (\hat{G}^{x_j^1}, \dots, \hat{G}^{x_j^\ell})$ for $j \in [n]$, and global public key. The global signing key is implicitly set to $\text{sk} := (x_1, \dots, x_\ell)$ where each x_i is shared into (x_1^i, \dots, x_n^i) by (t, n) -threshold scheme over \mathbb{Z}_p . For any set of indices, $\mathcal{T} \subseteq [n]$, of size t , it holds that $x_i = \sum_{j \in \mathcal{T}} \lambda_j x_j^i \pmod p$ where λ_j is a Lagrange coefficient defined as $\lambda_j := \prod_{t \in \mathcal{T} \setminus \{j\}} \frac{t}{t-j} \pmod p$.

5.1 Construction 2

Our first protocol follows the structure of the two-party case; parties work in sequence, from P_1 to P_t communicating only over the public broadcast channel. The first P_1 and the last P_t do the same as P_0 and P_1 did in the two-party case, respectively. However, each intermediate participant, P_2, \dots, P_{t-1} , must independently take on the roles of P_0 and P_1 , bringing forth new considerations to both the protocol and its security analysis. We present our construction in the preprocessing model where participating parties join the preprocessing procedure and then engage in the main signing protocol.

Zero-Sharing over Public Channel. In the preprocessing phase, participating parties set up shares of zero. We do this via Pedersen's commitments, and thus, extend the pp by adding $H \in \mathbb{G}$ ($H \neq G$) so that PGen outputs $\text{pp} = (\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T, H, G, \hat{G}, e)$. At the beginning, each party P_j picks $r_j \leftarrow \mathbb{Z}_p^*$, which is shared into r_{ji} so that P_j broadcasts r_{ji}, t_{ji} for all $i \neq j$ and a commitment $T_{ji} := G^{r_{ji}} H^{t_{ji}}$ of $i = j$. Commitments $T_{ij} := G^{r_{ij}} H^{t_{ij}}$ for $i \neq j$ are computed publicly. Thereafter, each P_j locally computes its share of zero as $w_j := r_j - \sum r_{ij}$ and commits to it computing $W_j := G^{w_j} H^{s_j}$. It also computes $t'_j := \sum t_{ji}$. The last steps consists in broadcasting a NIZK proof for the statement $\pi_j := [w_j, s_j, t'_j : W_j = G^{w_j} H^{s_j} \wedge \prod T_{ji} = G^{w_j} H^{t'_j}]$, whose verification confirms correctness of all shares.

Protocol description. Without loss of generality, let $\mathcal{T} = (1, \dots, t)$, *i.e.*, $(P_1, \dots, P_t) = (1, \dots, t)$ be the parties engaging in $\text{TSign}(\text{pp}, \{\text{sk}_j\}_{j \in \mathcal{T}}, M)$ as presented in Fig. 7. Fully general description is recoverable by replacing $\lambda_j \cdot x_j^i$ with $\lambda_{P_j} \cdot x_{P_j}^i$ in the following. We follow the template of the two-party case, which operates sequentially. The initial party P_1 communicates with the first intermediate party P_2 , and all intermediate parties behave the same until the last one, P_{t-1} , communicates with the final party P_t . The protocol proceeds backward until P_1 is reached. Subsequently, P_1 triggers the

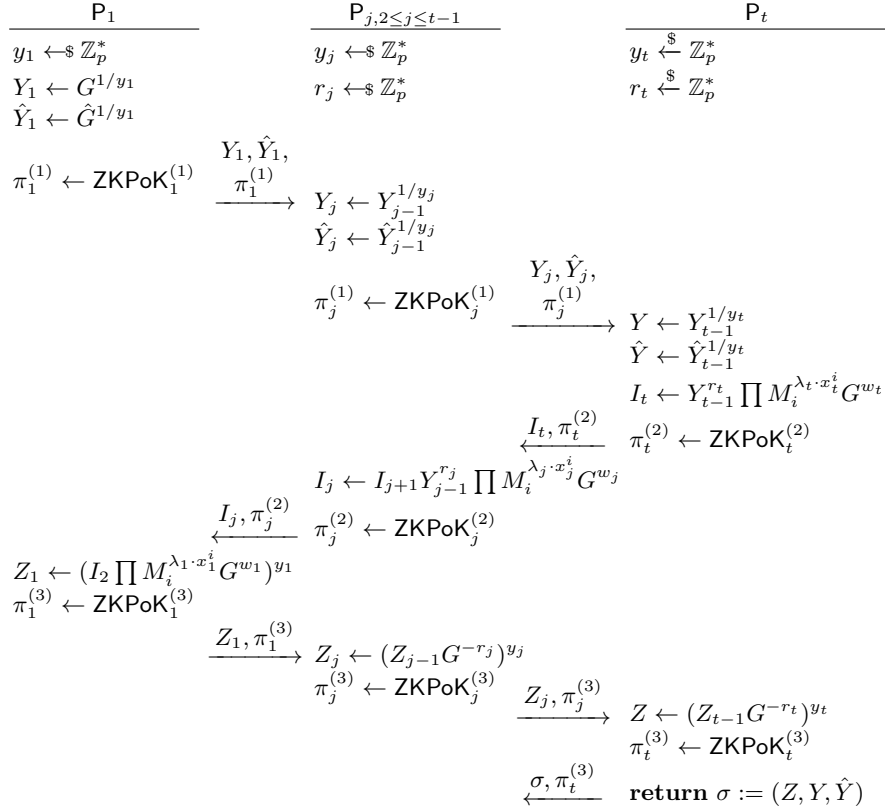


Fig. 7. t -party protocol for $\text{TSign}(\text{pp}, \{\text{sk}_j\}_{j \in J}, M)$. Products are taken for $i = 1$ to ℓ .

last round, which concludes when P_t broadcasts the signature. All proofs and the resulting signature are received and verified by everyone. If any party rejects, the output of the protocol is defined as \perp . Zero-knowledge proofs in Fig. 7 are defined as follows:

- $\pi_j^{(1)} := \text{ZKPoK}_j^{(1)}[y_j : Y_j = Y_{j-1}^{1/y_j} \wedge \hat{Y}_j = \hat{Y}_{j-1}^{1/y_j}]$ where $Y_0 = G$ and $\hat{Y}_0 = \hat{G}$ at $j = 1$.
- $\pi_j^{(2)} := \text{ZKPoK}_j^{(2)}[(r_j, w_j, s_j, \{x_j^i\}_{i \in [\ell]}) : I_j = I_{j+1} \cdot Y_{j-1}^{r_j} \prod_{i=1}^{\ell} M_i^{\lambda_j \cdot x_j^i} G^{w_j} \wedge W_j = G^{w_j} H^{s_j} \wedge_{i \in [\ell]} \hat{X}_j^i = \hat{G}^{x_j^i}]$ where $I_{t+1} = 1$ at $j = t$.
- $\pi_1^{(3)} := \text{ZKPoK}_1^{(3)}[(\{x_1^i\}_{i \in [\ell]}, y_1, w_1, s_1) : Z_1 = (I_2 \cdot \prod_{i=1}^{\ell} M_i^{\lambda_1 \cdot x_1^i} G^{w_1})^{y_1} \wedge Y_1^{y_1} = G \wedge W_1 = G^{w_1} H^{s_1} \wedge_{i \in [\ell]} X_1^i = \hat{G}^{x_1^i}]$
- $\pi_j^{(3)} := \text{ZKPoK}_j^{(3)}[y_j : Z_j = (Z_{j-1} \cdot G^{-r_j})^{y_j} \wedge Y_j^{y_j} = Y_{j-1}]$

An obvious difference from the two-party case is the presence of the intermediate parties, P_2, \dots, P_{t-1} . Computing $Y = G^{\frac{1}{y}}$ and $\hat{Y} = \hat{G}^{\frac{1}{y}}$ is done sequentially from P_1 to P_t , and y is defined by $y = \prod_{j=1}^t y_j$. If all parties are honest, the following holds for Z_1 (recall that at Z_1 all G^{w_j} are canceled out):

$$Z_1 = (I_2 \cdot \prod_{i=1}^{\ell} M_i^{\lambda_1 \cdot x_1^i})^{y_1} = (G^{\frac{r_2}{y_1} + \frac{r_3}{y_2 y_1} + \dots + \frac{r_t}{y_{t-1} \dots y_1}} \cdot \prod_{i=1}^{\ell} M_i^{\sum_{j=1}^t \lambda_j \cdot x_j^i})^{y_1} \quad (1)$$

The reason why Z_1 is computed in the second stage is the same as the two-party case: the blinding is useful for constructing the simulator in the presence of corrupted parties. It allows computing Z by sequentially unblinding Z_1 in the reverse order from P_2 to P_t . To see that the blinding factors are canceled as expected, observe that

$$\begin{aligned} Z &= (Z_{t-1} \cdot G^{-r_t})^{y_t} = (\dots (Z_1 \cdot G^{-r_2})^{y_2} \dots)^{y_{t-1}} \cdot G^{-r_t})^{y_t} \\ &= (\dots (G^{\frac{r_2}{y_1} + \frac{r_3}{y_2 y_1} + \dots + \frac{r_t}{y_{t-1} \dots y_1}} \cdot \prod_{i=1}^{\ell} M_i^{\sum_{j=1}^t \lambda_j \cdot x_j^i})^{y_1} G^{-r_2})^{y_2} \dots)^{y_{t-1}} \cdot G^{-r_t})^{y_t} \end{aligned}$$

holds. Concerning the exponent of G , we have:

$$\begin{aligned} & (\cdots (\frac{r_2}{y_1} + \frac{r_3}{y_2 y_1} + \cdots + \frac{r_t}{y_{t-1} \cdots y_1}) y_1 - r_2) \cdots) y_{t-1} - r_t) y_t \\ &= (\cdots (\frac{r_t}{y_{t-1}} + r_{t-1}) \frac{1}{y_{t-2}} \cdots) \frac{1}{y_3} + r_3) \frac{1}{y_2} + r_2) \frac{1}{y_1} \cdot y_1 - r_2) \cdots) y_{t-1} - r_t) y_t \\ &= (0 + \cdots + 0) y_t = 0. \end{aligned}$$

Therefore: $Z = (\prod_{i=1}^{\ell} M_i^{\sum_{j=1}^t \lambda_j \cdot x_j^i}) \prod_{j=1}^t y_j = (\prod_{i=1}^{\ell} M_i^{x_i})^y$.

Efficiency. We first note that the two-move pre-processing phase can be interleaved with the interactive signing protocol from Fig. 7 for a total of 5 moves. Besides, computation at this stage is cheap and most of the communication involves transmitting elements in \mathbb{Z}_p . The ZKPoK's are analogous to the two-party case and can be instantiated as shown in Appendix A. Communication and computation increase linearly with t . In many cases, however, the number of signers does not grow beyond one order of magnitude so t can stay relatively small. Furthermore, considering the applications from Appendix B that are multi-signatures and threshold ring signatures, ℓ will be small, meaning the ZKPoK's will be efficient and short.

Security. The key observation for correctness was given above. The strategies to prove unforgeability and unlinkability are similar to their two-party counterparts. An essential difference, however, is the presence of intermediate parties to simulate, considering a situation in which at most $t - 1$ signers can be corrupted.

Theorem 16 (Unforgeability). *Construction 2 is unforgeable against static corruption of at most $t - 1$ parties if TKGen is secure, all ZKPoK's are adaptive zero-knowledge, simulation extractable, and the original MS is unforgeable.*

Proof. As the overall proof structure is the same as the two-party case, we focus on describing the simulation of the signing oracle in the presence of corrupted parties. We begin by considering the simplest case where there is only one honest party among the participating parties, P_1, \dots, P_t . Depending on the position of the player, the simulation differs as follows:

- If the initial party, P_1 , is the honest one, it is simulated similarly as simulating P_0 in the two-party case considering the random factor in an accumulated form. Precisely, in the first step, it sets $(Y_1, \hat{Y}_1) := (Y', \hat{Y}')$ and simulates $\pi_1^{(1)}$. In the second step, it extracts r_j and y_j from all other parties and computes their accumulated random factor

$$r' := r_2 + \frac{r_3}{y_2} + \cdots + \frac{r_t}{y_{t-1} \cdots y_2} = \sum_{k=2}^t \left(r_k \prod_{\ell=2}^{k-1} y_{\ell}^{-1} \right)$$

where product $\prod_{\ell=2}^{k-1} y_{\ell}^{-1}$ is defined as 1 at $k = 2$. (Note that y_t is unnecessary.) It then outputs $Z_1 = G^{r'} Z'$ and simulates $\pi_1^{(3)}$. Provided that $\pi_j^{(1)}$ and $\pi_j^{(2)}$ for $j > 1$ allow knowledge extraction, Z_1 distributes the same as in the real protocol run. Indeed, as we argued for correctness (see eq. (1)), if all parties are honest, $Z_1 = (G^{\frac{r_2}{y_1} + \frac{r_3}{y_2 y_1} + \cdots + \frac{r_t}{y_{t-1} \cdots y_1}} \cdot \prod_{i=1}^{\ell} M_i^{\sum_{j=1}^t \lambda_j \cdot x_j^i}) y_1 = G^{r_2 + \frac{r_3}{y_2} + \cdots + \frac{r_t}{y_{t-1} \cdots y_2}} (\prod_{i=1}^{\ell} M_i^{\sum_{j=1}^t \lambda_j \cdot x_j^i}) y_1 = G^{r'} Z'$ where y_1 is implicitly determined by $1/\log_G Y_1$. Thus, the output from P_1 is perfectly simulated modulo the simulation extractability errors of $\pi_2^{(1)}, \dots, \pi_{t-1}^{(1)}$ and $\pi_t^{(2)}, \dots, \pi_2^{(2)}$, and zero-knowledge of $\pi_1^{(1)}$ and $\pi_1^{(3)}$.

- If the tail party, P_t , is the honest one, the simulation is the same as that of simulating P_1 in the two-party case except for the obvious notational adjustment. In the first step, it sets (Y, \hat{Y}) by (Y', \hat{Y}') , samples I_t uniformly from \mathbb{G} , and simulates $\pi_t^{(2)}$. In the second step, it sets $Z := Z'$ and simulates $\pi_t^{(3)}$. The simulation is perfect modulo the soundness errors of $\pi_1^{(1)}, \dots, \pi_{t-1}^{(1)}, \pi_{t-1}^{(2)}, \dots, \pi_2^{(2)}, \pi_1^{(3)}, \dots, \pi_{t-1}^{(3)}$, (that assure that (Y_{t-1}, \hat{Y}_{t-1}) and (Y, \hat{Y}) are in the same distribution), and zero-knowledge of $\pi_t^{(2)}$ and $\pi_t^{(3)}$.

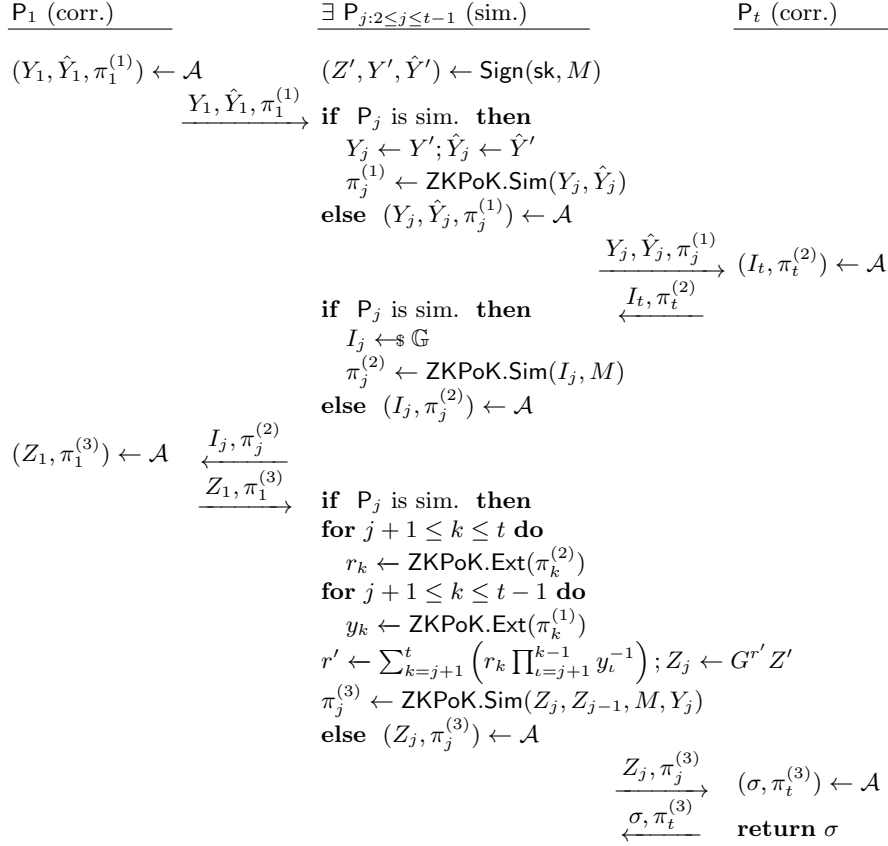


Fig. 8. Simulator for intermediate party P_j . \mathcal{A} manages its internal state. At $k = j + 1$, product $\prod_{\ell=j+1}^{k-1} y_\ell^{-1}$ is defined as 1.

- If an intermediate party, P_j , $j \in [2, t-1]$, is the honest one, the simulation is done as shown in Fig. 8. It is a mixture of the above simulation strategies. It works like the initial party for the right (ascending) parties in the first and third steps, and like the tail party for the left (descending) parties in the second step. For the same reason for above cases, the simulation in this case is perfect modulo the soundness errors of $\pi_1^{(1)}, \dots, \pi_{j-1}^{(1)}, \pi_{j-1}^{(2)}, \dots, \pi_1^{(2)}$, and $\pi_1^{(3)}, \dots, \pi_{j-1}^{(3)}$, and simulation extractability errors of $\pi_{j+1}^{(1)}, \dots, \pi_{t-1}^{(1)}$ and $\pi_t^{(2)}, \dots, \pi_{j+1}^{(2)}$, and zero-knowledge of $\pi_j^{(1)}, \pi_j^{(2)}$, and $\pi_j^{(3)}$.

The above procedure relies on the proof $\pi^{(1)}$ and $\pi^{(2)}$ for simulation and extraction, which requires simulation extractability. For the simulation extractability error ($\mathcal{E}_{\text{SimExt}}$), the soundness errors (\mathcal{E}_{Snd}), and zero-knowledge errors (\mathcal{E}_{ZK}) we obtain that the adversary's advantage when executing q queries to the signing oracle is given by $\text{Adv}_{\text{TMS}, \ell, t, n}^{\text{UNF}}(1^\kappa, \mathcal{A}) < \text{Adv}_{\text{MS}, \ell}^{\text{UNF}}(1^\kappa, \mathcal{A}) + q((t-1)(2\mathcal{E}_{\text{SimExt}} + 2\mathcal{E}_{\text{Snd}}) + 3\mathcal{E}_{\text{ZK}})$. This concludes the case where only one honest party is participating.

As the next step, we consider the case where only two parties, P_u and P_v for $u < v$, are honest and all others are corrupted. The simulation is as follows:

- Simulate the right-hand honest party, P_v , in the same way as the single honest party case as above.
- For the left-hand honest party, P_u , first follow the protocol to output Y_u, \hat{Y}_u , and $\pi_u^{(1)}$ in the first round. Then, in the second round, pick I_u uniformly from \mathbb{G} and simulate $\pi_u^{(2)}$. Finally, in the third round, pick Z_u uniformly from \mathbb{G} and simulate $\pi_u^{(3)}$.

We claim that the simulation is perfect modulo the simulations of the relevant zero-knowledge proofs, which we ignore in the following. We follow the game transition argument as follows:

- Let θ_0 be a joint view of corrupted parties in a protocol run.

- We first replace the right-hand honest party, P_v , with the above simulation. Let θ_1 be the joint view obtained during the protocol. Since the single honest party case is the perfect simulation, θ_0 and θ_1 distributes identically.
- Next we replace the left-hand honest party, P_u , with the simulation. Let the resulting view as θ_2 . We claim that θ_1 and θ_2 distribute identically and prove it with the following claim.

Claim 1. θ_2 distributes identically to θ_1 .

Proof. We actually compare θ_2 with θ_0 instead. We first explain some notations and conventions. For group elements such as Z_1 appear in the transcript, their small-case counterpart such as z_1 denotes a logarithm with respect to generator G (or \hat{G}). By \vec{y}_k , we denote $\prod_{j=1}^k y_j$. Without loss of generality, we consider the dimension of the message vector is one, and denote the message by M . Thus each party P_i has signing key x_i and public-key $\hat{X}_i = \hat{G}^{x_i}$. By x and y , we denote $\sum_{i=1}^t x_i$ and $\prod_{i=1}^t y_i$, respectively.

For honest parties P_u and P_v , a joint view of corrupt players consist of all variables appear in the description of the protocol except for $x_u, x_v, y_u, y_v, r_u, r_v, w_u$, and w_v , which we call *hidden variables*. To handle the corner case without notational gaps, we let $Y_0 = G$, and $Z_0 = I_1$, and let P_1 compute $I_1 = I_2 Y_0^{r_1} M^{x_1} G^{w_1}$ and $Z_1 = (Z_0 G^{-r_1})^{y_1}$ for random r_1 . This is only a cosmetic change where the original description is obtained by letter $r_1 = 0$.

Observe that x_u is determined uniquely by relation $X_u = G^{x_u}$, and x_v, y_u , and y_v are also determined uniquely by X_v, Y_u, Y_v , respectively. We call them fixed hidden values and remaining r_u, r_v, w_u , and w_v , as *free hidden values*. Hidden values respective to the rightmost honest party P_v establishes relations:

$$i_v = i_{v+1} + \vec{y}_{v-1}^{-1} r_v + m x_v + w_v \quad (2\text{nd round computation}) \quad (2)$$

$$z_v = z_{v-1} y_v - r_v y_v \quad (3\text{rd round computation}) \quad (3)$$

Similarly, for those respective to P_u establishes relations:

$$i_u = i_{u+1} + \vec{y}_{u-1}^{-1} r_u + m x_u + w_u \quad (2\text{nd round computation}) \quad (4)$$

$$z_u = z_{u-1} y_u - r_u y_u \quad (3\text{rd round computation}) \quad (5)$$

Pre-processed values satisfy:

$$w_u + w_v + \sum_{i \in [t] \setminus \{u, v\}} w_i = 0 \quad (6)$$

The view θ_0 is obtained from concrete values assigned to the hidden variables respecting all these relations. On the other hand, θ_2 is obtained by picking i_u and z_u uniformly at random. We show that for any choice of i_u and z_u there exist an assignment to the free hidden variables that is consistent regarding the above relations. As for a preparation, we present some fixed relations in second and third rounds that hold in both views.

$$i_{v+1} = \sum_{j=v+1}^t r_j \vec{y}_{j-1}^{-1} + m \sum_{j=v+1}^t x_j + \sum_{j=v+1}^t w_j \quad (I_{v+1} \leftarrow I_t) \quad (7)$$

$$i_{u+1} = i_v + \sum_{j=u+1}^{v-1} (r_j \vec{y}_{j-1}^{-1} + m x_j + w_j) \quad (I_{u+1} \leftarrow I_v) \quad (8)$$

$$z_{u-1} = i_u \vec{y}_{u-1} + m \vec{y}_{u-1} \sum_{j=1}^{u-1} x_j + \vec{y}_{u-1} \sum_{j=1}^{u-1} w_j \quad (I_u \rightarrow I_1 \rightarrow Z_{u-1}) \quad (9)$$

$$z_{v-1} = z_u \prod_{j=u+1}^{v-1} y_j - \sum_{j=u+1}^{v-1} r_j \prod_{k=j}^{v-1} y_k \quad (Z_u \rightarrow Z_{v-1}) \quad (10)$$

$$z_v = m x \vec{y}_v + \sum_{j=v+1}^t r_j \prod_{k=v+1}^{j-1} y_k^{-1} \quad (Z_v \rightarrow Z) \quad (11)$$

Now, consider that i_u and z_u are selected and fixed by the simulation. From (5) and (9), free hidden variable r_u is determined as

$$r_u = i_u \vec{y}_{u-1} + m \vec{y}_{u-1} \sum_{j=1}^{u-1} x_j + \vec{y}_{u-1} \sum_{j=1}^{u-1} w_j - z_u y_u^{-1}. \quad (12)$$

This r_u is consistent with other relations since it can be determined also from (2,3,4,6) and (8,10,11,7) as follows. From (3) and (10),

$$r_v = z_{v-1} - z_v y_v^{-1} \quad (13)$$

$$= z_u \prod_{j=u+1}^{v-1} y_j - \sum_{j=u+1}^{v-1} r_j \prod_{k=j}^{v-1} y_k - z_v y_v^{-1} \quad (14)$$

From (14) and (2), we obtain w_v as

$$w_v = i_v - i_{v+1} - m x_v - (z_k \vec{y}_u^{-1} - \sum_{j=u+1}^{v-1} r_j \prod_{k=1}^{j-1} y_j^{-1} - z_v \vec{y}_v^{-1}). \quad (15)$$

From (15), (6), (4), and (8), we compute r_u as

$$\begin{aligned} r_u &= i_u \vec{y}_{u-1} - (i_v + \sum_{u+1}^{v-1} (r_j \vec{y}_{j-1}^{-1}) + m x_j + w_j) \vec{y}_{u-1} - m x \vec{y}_{u-1} \\ &\quad + \vec{y}_{u-1} (i_v - i_{v+1} - m x_v - z_u \vec{y}_u^{-1} \\ &\quad + \sum_{j=u+1}^{v-1} r_j \prod_{k=1}^{j-1} y_k^{-1} + z_v \vec{y}_v^{-1} + \sum_{j \in [t] \setminus \{u,v\}} w_j) \end{aligned} \quad (16)$$

Substitute i_{v+1} and z_v in (16) with those of (7) and (11) we have:

$$\begin{aligned} r_u &= i_u \vec{y}_{u-1} - i_v \vec{y}_{u-1} - \vec{y}_{u-1} \sum_{j=u+1}^{v-1} r_j \vec{y}_{j-1}^{-1} - \vec{y}_{u-1} m \sum_{j=u+1}^{v-1} x_j \\ &\quad - \vec{y}_{u-1} \sum_{j=u+1}^{v-1} w_j - m x_u \vec{y}_{u-1} + i_v \vec{y}_{u-1} - \vec{y}_{u-1} \sum_{j=v+1}^t r_j \vec{y}_{j-1}^{-1} \\ &\quad - \vec{y}_{u-1} m \sum_{j=v+1}^t x_j - \vec{y}_{u-1} \sum_{j=v+1}^t w_j - m x_v \vec{y}_{u-1} - z_u \vec{y}_u^{-1} \vec{y}_{u-1} \\ &\quad + \vec{y}_{u-1} \sum_{j=u+1}^{v-1} r_j \vec{y}_{j-1}^{-1} + \vec{y}_{u-1} \sum_{j \in [t] \setminus \{u,v\}} w_j \\ &\quad + m x \vec{y}_{u-1} + (\sum_{j=v+1}^t r_j \prod_{k=v+1}^{j-1} y_k^{-1}) \vec{y}_v^{-1} \vec{y}_{u-1} \end{aligned} \quad (17)$$

Wrapping terms in (17), we see that all terms including r_j vanish and we have

$$r_u = i_u \vec{y}_{u-1} + m \vec{y}_{u-1} \sum_{j=1}^{u-1} x_j + \vec{y}_{u-1} \sum_{j=1}^{u-1} w_j - z_u y_u^{-1} \quad (18)$$

which is the same as (12). This complete the proof of the claim. \square

From the above, we conclude the two-honest-party case. Since the simulation is perfect, it straightforwardly extends to the most general case where an arbitrary number of honest parties are present in the signing process:

- Simulate the rightmost honest party as done for the single honest party case.

- For each of other left-located honest parties, follow the simulation procedure of the left-hand honest party in the two honest party case.

To show that the simulation remains perfect, we follow the same procedure where we first replace the rightmost party with the simulation, and then replace other honest parties one by one in the descending order, *i.e.*, from the right to left. In every transition, the output from the simulated party distributes identically to the original one. This concludes the proof of Theorem 16. \square

The following theorem can be proven similarly to the two-party case.

Theorem 17 (Public Key Unlinkability). *Construction 2 is public key unlinkable against static corruption of at most $t - 1$ parties if TGen is secure, all ZKPoK's are secure, and the original MS is origin and public key class-hiding.*

Proof. We proceed as in Theorem 15 except that we make use of the simulation strategies addressed in the above proof of unforgeability (Theorem 16). We construct a simulator that plays the role of the adversary in the unlinkability game against MS given access to an adversary \mathcal{A} who plays the unlinkability game against TMS. As before, given pp as input, the simulator invokes \mathcal{A} and outputs \mathcal{C} obtained from \mathcal{A} . Then, given $(\text{pk}_1, \text{pk}_2^b)$ as input, the simulator runs SimTKGen for pk_1 with \mathcal{C} to obtain, for $\text{sk}_1^{(j)}$ for $j \in \mathcal{C}$ and $\text{pk}_1^{(j)}$ for $j \in \mathcal{T}$. The simulator then invokes \mathcal{A} with $\{\text{sk}_1^{(j)}\}_{j \in \mathcal{C}}$, $\{\text{pk}_1^{(j)}\}_{j \in \mathcal{T}}$, and pk_2^b as input. Again, validity of the simulation up to this point is due to the security of TGen.

On receiving a query from \mathcal{A} to OTSign on pk_2^b and M , the simulator forwards it to its oracle and returns the response to \mathcal{A} . This part of the simulation is perfect due to the origin-hiding of MS.

On receiving a query from \mathcal{A} to OTSign on pk_1 and M , the simulator forwards it to its oracle to obtain an MS signature for M . Subsequently, it uses the obtained signature to simulate an invocation of TSign (using Steps) on pk_1 with M as explained in the proof of Theorem 16. Validity of this part of the simulation is due to the security of ZKPoK's as before. If Steps 's simulation results in \perp (due to misbehavior of a corrupted party), it returns \perp .

The simulator outputs whatever \mathcal{A} outputs. Since the view of \mathcal{A} is correctly simulated, the output is correct whenever \mathcal{A} wins the game against TMS. \square

5.2 Construction 3

Our second protocol builds on more general results and thus follows a different strategy compared to the previous constructions. Specifically, we observe that the (non-interactive) distributed multiplication protocol proposed by Abe [Abe99] aligns well with the signing requirements. This protocol enables parties to compute the product of x and y (the key operation in the MS signing algorithm) using the Pedersen VSS scheme presented in Sec. 2.3. Abe's protocol tolerates up to $n/2$ corrupt players, resulting in a four-move interactive signing process that eliminates the need for any zero-knowledge proofs as we describe next.

As before, we let $\mathcal{T} = (1, \dots, t)$, *i.e.*, $(P_1, \dots, P_t) = (1, \dots, t)$. Recall that our goal is to compute $Z = \prod_{i=1}^{\ell} M_i^{x_i y}$, $Y = G^{\frac{1}{y}}$ and $\hat{Y} = \hat{G}^{\frac{1}{y}}$. This can be done by having each party deliver verifiable shares $w_i^{jk} := x_i^{jk} y^{jk}$, allowing to reconstruct the share of product $x_i^k y^k$ without revealing any hidden shares. It remains to compute the final $\frac{1}{y}$ but this becomes cumbersome given only shares of y . Our approach is the use of an auxiliary value C (whose shares C^j are also distributed via VSS) that can be used to compute Cy easily following Abe's method. In brief, any party that gets $Cy \in \mathbb{Z}_p$ can compute $d^j := (Cy)^{-1} C^j$ from the values they hold. This translates into a share for $\frac{1}{y}$ that can restore $\frac{1}{y}$ by Lagrange interpolation. Finally, upon obtaining $\{x_i y\}_{i \in [\ell]}$ and $\frac{1}{y}$, parties can compute their partial signatures and gather them to output the full signature as shown below (analogous for \hat{Y}):

$$\begin{aligned} Z &= \prod_{j \in \mathcal{T}} Z^j = \prod_{i=1}^{\ell} M_i^{\sum_{j \in \mathcal{T}} \lambda_j w_i^j} = \prod_{i=1}^{\ell} M_i^{x_i y} = \left(\prod_{i=1}^{\ell} M_i^{x_i} \right)^y \\ Y &= \prod_{j \in \mathcal{T}} Y^j = G^{\sum_{j \in \mathcal{T}} \lambda_j d^j} = G^{\sum_{j \in \mathcal{T}} (Cy)^{-1} \lambda_j C^j} = G^{\frac{C}{Cy}} = G^{\frac{1}{y}} \end{aligned} \tag{19}$$

MPC Round	Computation	Communication
Round I - P_j	-	$6t \mathbb{G} + 5(\ell + 1)$
Round II - P_k	$6(\ell + 1)(t + 3) \mathbb{G} $	$\ell + 1$
Round III - P_j	$(\ell + 1) \mathbb{G} + \hat{\mathbb{G}} $	$2 \mathbb{G} + \hat{\mathbb{G}} $
Round IV - P_k	$2 \mathbb{G} + \hat{\mathbb{G}} $	-

Table 2. Costs of construction 2. Computation counts number of exponentiations in the relevant groups without optimization for multi-base exponentiations.

Successively, recall that TKGen generates local keys $\text{sk}_j := (x_j^1, \dots, x_j^\ell)$ and $\text{pk}_j := (\hat{G}^{x_j^1}, \dots, \hat{G}^{x_j^\ell})$ for $j \in \mathcal{T}$, and global public key pk . As before, the global signing key is implicitly set to $\text{sk} := (x_1, \dots, x_\ell)$ where each x_i is shared into (x_1^i, \dots, x_n^i) using (t, n) -threshold scheme over \mathbb{Z}_p . This can be done with Pedersen’s VSS to obtain $(\{x_j^i\}_{j \in \mathcal{T}}^{i \in [\ell]}, \{\hat{G}^{x_j^i}\}_{j \in \mathcal{T}}^{i \in [\ell]}, \text{pk}) \leftarrow \text{TKGen}(\text{pp}, \ell, t, n)$ such that for each x_i a unique random value $R_{x_i} \xleftarrow{\$} \mathbb{Z}_p$ is computed from polynomials $F_{x_i}(X)$ and $D_{x_i}(X)$ alongside the corresponding commitments as shown below:

$$\text{VSS}(x_i, R_{x_i})[G, H] \xrightarrow{F_{x_i}, D_{x_i}} (\{x_j^i\}, \{R_{x_i}^j\})_{j \in \mathcal{T}}^{i \in [\ell]} [EX_i^0, EX_i^1, \dots, EX_i^n] \quad (20)$$

As a result, every party P_j obtains $(\{x_j^i\}_{i \in [\ell]}, \{\hat{G}^{x_j^i}\}_{i \in [\ell]}, \{R_{x_i}^j, EX_i^j\}_{i \in [\ell]}, \text{pk})$, where each EX_i^j is needed to verify the validity of the shares x_j^i and $R_{x_i}^j$. The next step is to do the same for the value Y :

$$\text{VSS}(y, R_y)[G, H] \xrightarrow{F_y, D_y} (y^j, R_y^j) [EY^0, EY^1, \dots, EY^n]$$

For ease of exposition, we also assume a dealer who computes the VSS for y and C . Once that all parties have shares for x and y , Abe’s protocol proceeds with each P_j picking a t -degree random polynomial to share $x_j y_j$. The share C^{j_i} is privately sent to party i .

The resulting non-interactive protocol $\text{TSign}(\{\text{sk}_j\}_{j \in \mathcal{T}}, M)$ for $M = (M_1, \dots, M_\ell)$ as run by parties P_j ($1 \leq j \leq t$) is given in Fig. 9. We denote by VY^j the product $\prod_{m=0}^t EY^m y^m$ and $\langle \dots \rangle$ is used for variables that every party can compute locally. We observe that parties Party j, k can be selected in any way. When P_j prepares the shares for P_k , w_i^{jk} denotes $x_i^{jk} y^{jk}$ and v^{jk} denotes $C^{jk} y^{jk}$. In addition, \mathcal{K} is the subgroup of $[\ell]$ including enough number to restore Cy and \mathcal{T} is the subgroup of n including more than $t - 1$ numbers to compute the full signature. Security of this protocol directly follows from that of the underlying VSS and Abe’s multiplication protocol. We analyse its efficiency on Table 5.2.

5.3 Application to Delegatable Credentials

The first work on delegatable anonymous credentials from MS by Crites and Lysyanskaya [CL19] was based on the following idea: a root authority (or CA) produces a signature on Alice’s public key, who can subsequently do the same to delegate her credential to Bob. More in detail, the CA produces a signature $\sigma_{\text{CA} \rightarrow \text{Alice}}$ for a message $M = \text{pk}_{\text{Alice}}$, which Alice can use to authenticate herself by proving knowledge of sk_{Alice} . To delegate her credential, Alice signs Bob’s public key pk_{Bob} , producing $\sigma_{\text{Alice} \rightarrow \text{Bob}}$. This approach works for any regular signature scheme. However, as observed in [CL19], when the signatures are MS, they can be adapted to provide stronger privacy guarantees. In brief, if Alice is known to Bob under a pseudonymous public key $\text{pk}'_{\text{Alice}}$, she can consistently adapt the signatures in the delegation chain to produce $\sigma_{\text{CA} \rightarrow \text{Alice}'}$ and $\sigma_{\text{Alice}' \rightarrow \text{Bob}}$. this way, Bob can prove knowledge of his secret key for a valid chain (*i.e.*, a chain that goes all the way back to a valid signature from the CA).

Subsequent work by Crites and Lysyanskaya [CL21] and Putman and Martin [PM24] improved the original construction to support attributes. In all cases, it is essential that the CA’s signature is a MS so that it can be adapted to a new user pseudonym (even if the root key is never randomized). Because of this, all previous work assume a single trusted issuer as the CA. Our TMS construction can

Dealer :

$R_y, R_c \xleftarrow{\$} \mathbb{Z}_p$ and generate F_y, D_y, F_c and D_c .
 $VSS(y, R_y)[G, H] \xrightarrow{F_y, D_y} (y^j, R_y^j)[EY^0, EY^1, \dots, EY^n]$
 $VSS(C, R_C)[G, H] \xrightarrow{F_C, D_C} (C^j, R_C^j)[EC^0, EC^1, \dots, EC^m]$
return to P_j ($1 \leq j \leq t$) : $(y_j, R_y^j, \{EY^j\}_{j \in \mathcal{T}}, C^j, R_C^j, \{EC^j\}_{j \in \mathcal{T}})$

Party j : $(M, sk_j, R_j, y^j, R_y^j, C^j, R_C^j)$

for $i \in [\ell]$:

$R_{xy}^j \xleftarrow{\$} \mathbb{Z}_p$ and generate $F_{xy}, D_{xy}, F_{xy}^\dagger, D_{xy}^\dagger, F_{xy}^{\dagger\dagger}$ and $D_{xy}^{\dagger\dagger}$.
 $VSS(x_i^j, R_{x_i}^j)[G, H] \xrightarrow{F_{xy}, D_{xy}} (x_i^{jk}, R_{x_i}^{jk})[\langle VX_i^{jk} \rangle, EX_i^{j1}, \dots, EX_i^{jt}]$
 $VSS(x_i^j, R_{xy}^j)[VY^j, H] \xrightarrow{F_{xy}^\dagger, D_{xy}^\dagger} ((x_i^{jk}), R_{xy}^{jk})[EXY_i^{j0}, EXY_i^{j1}, \dots, EXY_i^{jt}]$
 $VSS(x_i^j \cdot y^j, R_y^j \cdot x_i^j + R_{xy}^j)[G, H] \xrightarrow{F_{xy}^{\dagger\dagger}, D_{xy}^{\dagger\dagger}} (w_i^{jk}, R_{w_i}^{jk})[\langle EXY_i^{jk} \rangle, EW_i^{j1}, \dots, EW_i^{jt}]$
 $R_{Cy}^j \xleftarrow{\$} \mathbb{Z}_p$; Generate $F_{Cy}, D_{Cy}, F_{Cy}^\dagger, D_{Cy}^\dagger, F_{Cy}^{\dagger\dagger}$ and $D_{Cy}^{\dagger\dagger}$.
 $VSS(C^j, R_C^j)[G, H] \xrightarrow{F_{Cy}, D_{Cy}} (C^{jk}, R_C^{jk})[\langle VC^j \rangle, EC^{j1}, \dots, EC^{jt}]$
 $VSS(C^j, R_{Cy}^j)[VC^j, H] \xrightarrow{F_{Cy}^\dagger, D_{Cy}^\dagger} ((C^{jk}), R_{Cy}^{jk})[ECY^{j0}, ECY^{j1}, \dots, ECY^{jt}]$
 $VSS(C^j \cdot y^j, R_y^j \cdot C^j + R_{Cy}^j)[G, H] \xrightarrow{F_{Cy}^{\dagger\dagger}, D_{Cy}^{\dagger\dagger}} (v^{jk}, R_v^{jk})[\langle ECY^{j0} \rangle, EV^{j1}, \dots, EV^{jt}]$
return to P_k ($1 \leq k \leq t$): $(\{x_i^{jk}\}_{i \in \ell}, \{R_{x_i}^{jk}\}_{i \in \ell}, \{R_{xy}^{jk}\}_{i \in \ell}, \{w_i^{jk}\}_{i \in \ell}, \{R_{w_i}^{jk}\}_{i \in \ell}, C^{jk}, R_C^{jk}, R_{Cy}^{jk}, v^{jk}, R_v^{jk})$
broadcast: $\{EX_i^{jk}\}_{k \in [t]}, \{EXY_i^{jk}\}_{k \in [t]}, \{EW_i^{jk}\}_{k \in [t]}, \{EC_i^{jk}\}_{k \in [t]}, \{ECY_i^{jk}\}_{k \in [t]}, \{EV_i^{jk}\}_{k \in [t]}$

Party k : (\dots)

*Verifies $\{\{x_i^{jk}\}, \{R_{x_i}^{jk}\}, \{R_{xy}^{jk}\}, \{w_i^{jk}\}, \{R_{w_i}^{jk}\}\}_{i \in \ell}$

for $i \in \ell$

if $G^{x_i^{jk}} H^{R_{x_i}^{jk}} = VX_i^j \prod_{m=1}^t (EX_i^{jm})^{k^m} \wedge VY_j^{x_i^{jk}} H^{R_{xy}^{jk}} = \prod_{m=0}^t (EXY_i^{jm})^{k^m} \wedge$
 $G^{w_i^{jk}} H^{R_{w_i}^{jk}} = EXY_i^{j0} \prod_{m=1}^t (EW_i^{jm})^{k^m}$ **then** $w_i^k := \sum_{j \in \mathcal{T}} \lambda_j w_i^{jk}$ **else return** \perp
if $G^{C^{jk}} H^{R_C^{jk}} = VC^j \prod_{m=1}^t (EC^{jm})^{k^m} \wedge VY_j^{C^{jk}} H^{R_{Cy}^{jk}} = \prod_{m=0}^t (ECY_i^{jm})^{k^m} \wedge$
 $G^{v^{jk}} H^{R_v^{jk}} = ECY^{j0} \prod_{m=1}^t (EV^{jm})^{k^m}$ **then** $v^k := \sum_{j \in \mathcal{T}} \lambda_j v^{jk}$ **else return** \perp
return to P_j : $(\{w_i^k\}_{i \in [\ell]}, v^k)$

Party j : $(\{w_i^k\}_{i \in [\ell]}, \{v^k\}_{k \in \mathcal{K}})$

$Cy := \sum_{k \in \mathcal{K}} \lambda_k v^k; d^j := (Cy)^{-1} C^j (= [y^{-1}]_j)$

Let $Z^j := \prod_{i=1}^t M_i^{\lambda_j w_i^j}; Y^j := G^{\lambda_j d^j}; \hat{Y}^j := \hat{G}^{\lambda_j d^j}$ **return** (Z^j, Y^j, \hat{Y}^j) to P_k

Party k : $(\{Z^j\}_{j \in \mathcal{T}}, \{Y^j\}_{j \in \mathcal{T}}, \{\hat{Y}^j\}_{j \in \mathcal{T}})$

$Z := \prod_{j \in \mathcal{T}} Z^j; Y := \prod_{j \in \mathcal{T}} Y^j; \hat{Y} := \prod_{j \in \mathcal{T}} \hat{Y}^j$; **return** $\sigma := (Z, Y, \hat{Y})$

Fig. 9. Non-interactive TSign

be used as a drop-in replacement for the CA's signature to distribute trust. With this in mind, any threshold of authorized issuers could produce a signature on the user's public key, enabling more use cases for this primitive as in the case where users get credentials from certain government authorities discussed in [CL19].

6 Experimental Evaluation

We prototyped our (interactive) constructions in Rust based on the mercurial signature implementation from [CDLP22]. However, we replaced the BLS12-381 crate by Filecoin's BLS12-381 crate (blasters [Lab21], a Rust wrapper around the blst library [Lab20]). Our implementation and related documentation is available in [Nan24]. As previously mentioned, we only considered cases where $\ell \in \{2, 5, 10\}$, which cover all known applications. We also implemented our schemes switching the message and public key groups. However, since the ZKPoK's require parties to prove knowledge of their secret key when computing the multi-exponentiations to the message part, no significant change

Scheme	# of Parties	Signing (ms)		
		$\ell = 2$	$\ell = 5$	$\ell = 10$
MS [FHS19]	1	0.3	0.4	0.5
TMS (Sec. 4)	2	3.9	6.2	10.1
TMS (Sec. 5.1)	5	13.3	19.3	29.6
TMS (Sec. 5.1)	10	28.0	40.8	60.5

Table 3. Timing of signature generation for messages of size ℓ in milliseconds.

in performance is gained. Nonetheless, if one relaxes the security requirement for semi-honest parties, switching groups would improve performance at the cost of a slightly bigger signature size (elements in \mathbb{G}_1 and \mathbb{G}_2 are of size 48 and 96 bytes, respectively).

Table 3 summarizes the execution times for the signing algorithm of our TMS and the original MS. Verification times are the same for all variants (1.8ms for $\ell = 2$, 3ms for $\ell = 5$ and 5ms for $\ell = 10$). We used the nightly compiler, the Criterion library, and all the benchmarks were run on a MacBook Pro M3 with 32 GB of RAM with no extra optimizations. In all cases, the standard deviation was below 1ms. For TMS we considered the two-party and threshold cases with five and ten parties. As expected, the computational complexity of our interactive signing process scales linearly with the number of parties. This is also the case for communication. More concretely, the initial and final parties broadcast two ZKPoK’s and receive $3n - 4$. Similarly, intermediate parties broadcast three messages and receive $3n - 5$. Nevertheless, considering that all the applications discussed require a few parties, and the additional features offered by TMS, we find the overhead compared to standard MS well justified.

The most closely related work to ours is the threshold SPS from [CKP⁺23], but, to the best of our knowledge, it has not been implemented. This leave us with few options to compare the performance of TMS. One option could be to consider the multi-signature MuSig2 from [NRS21], but their work is incomparable to ours as it works in a pairing-free group and focuses on other functionalities.

7 Conclusion

In this work, we develop interactive threshold mercurial signatures (TMS). To showcase the power of this primitive, we presented constructions for both the two-party and multi-party cases, discussing their instantiation under different scenarios. Our experimental evaluation suggests that our constructions are practical when instantiated in the ROM. Most importantly, our interactive approach allows us to generate signatures with an affine linear transformation in the public key structure, translating into stronger privacy properties for many applications. Something that previous works in the setting were unable to achieve.

We also explored the instantiation of TMS from standard building blocks, finding it practical. Compared to the existing threshold structure-preserving signatures for the generalized case, our constructions are very competitive in terms of efficiency while covering other use cases.

All in all, interactive TMS offer greater flexibility than standard MS with relatively little overhead. Therefore, revising existing applications of MS (and more in general EQS) through the optics of TMS can be a very promising direction for future work. Another is the study of alternatives that could offer straight-line knowledge extraction to obtain concurrently secure schemes.

Acknowledgements. The authors thank the anonymous reviewers of Asiacrypt 2024 for their insightful comments and very helpful suggestions.

References

- AB21. Handan Kiliç Alper and Jeffrey Burdges. Two-round trip schnorr multi-signatures via delinearized witnesses. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part I*, volume 12825 of *LNCS*, pages 157–188, Virtual Event, August 16–20, 2021. Springer, Heidelberg, Germany.

- Abe99. Masayuki Abe. Robust distributed multiplication without interaction. In Michael J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 130–147, Santa Barbara, CA, USA, August 15–19, 1999. Springer, Heidelberg, Germany.
- ABF23. Gennaro Avitabile, Vincenzo Botta, and Dario Fiore. Extendable threshold ring signatures with enhanced anonymity. In Alexandra Boldyreva and Vladimir Kolesnikov, editors, *PKC 2023, Part I*, volume 13940 of *LNCS*, pages 281–311, Atlanta, GA, USA, May 7–10, 2023. Springer, Heidelberg, Germany.
- AF04. Masayuki Abe and Serge Fehr. Adaptively secure feldman VSS and applications to universally-composable threshold cryptography. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 317–334, Santa Barbara, CA, USA, August 15–19, 2004. Springer, Heidelberg, Germany.
- AFG⁺10. 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, Santa Barbara, CA, USA, August 15–19, 2010. Springer, Heidelberg, Germany.
- AGHO11. Masayuki Abe, Jens Groth, Kristiyan Haralambiev, and Miyako Ohkubo. Optimal structure-preserving signatures in asymmetric bilinear groups. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 649–666, Santa Barbara, CA, USA, August 14–18, 2011. Springer, Heidelberg, Germany.
- AHAN⁺22. Diego F. Aranha, Mathias Hall-Andersen, Anca Nitulescu, Elena Pagnin, and Sophia Yakubov. Count me in! Extendability for threshold ring signatures. In Goichiro Hanaoka, Junji Shikata, and Yohei Watanabe, editors, *PKC 2022, Part II*, volume 13178 of *LNCS*, pages 379–406, Virtual Event, March 8–11, 2022. Springer, Heidelberg, Germany.
- BCK⁺22. Mihir Bellare, Elizabeth C. Crites, Chelsea Komlo, Mary Maller, Stefano Tessaro, and Chenzhi Zhu. Better than advertised security for non-interactive threshold signatures. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part IV*, volume 13510 of *LNCS*, pages 517–550, Santa Barbara, CA, USA, August 15–18, 2022. Springer, Heidelberg, Germany.
- BEK⁺21. 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, Vienna, Austria, December 13–15, 2021. Springer, Heidelberg, Germany.
- BF20. Balthazar Bauer and Georg Fuchsbauer. Efficient signatures on randomizable ciphertexts. In Clemente Galdi and Vladimir Kolesnikov, editors, *SCN 20*, volume 12238 of *LNCS*, pages 359–381, Amalfi, Italy, September 14–16, 2020. Springer, Heidelberg, Germany.
- BF24. Balthazar Bauer and Georg Fuchsbauer. On security proofs of existing equivalence class signature schemes. Cryptology ePrint Archive, Paper 2024/183, 2024.
- BFPV11. Olivier Blazy, Georg Fuchsbauer, David Pointcheval, and Damien Vergnaud. Signatures on randomizable ciphertexts. In Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio Nicolosi, editors, *PKC 2011*, volume 6571 of *LNCS*, pages 403–422, Taormina, Italy, March 6–9, 2011. Springer, Heidelberg, Germany.
- BFR24. Balthazar Bauer, Georg Fuchsbauer, and Fabian Regen. On proving equivalence class signatures secure from non-interactive assumptions. In Qiang Tang and Vanessa Teague, editors, *Public-Key Cryptography – PKC 2024*, pages 3–36, Cham, 2024. Springer Nature Switzerland.
- BGLS03. Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 416–432, Warsaw, Poland, May 4–8, 2003. Springer, Heidelberg, Germany.
- BHKS18. Michael Backes, Lucjan Hanzlik, Kamil Kluczniak, and Jonas Schneider. Signatures with flexible public key: Introducing equivalence classes for public keys. In Thomas Peyrin and Steven Galbraith, editors, *ASIACRYPT 2018, Part II*, volume 11273 of *LNCS*, pages 405–434, Brisbane, Queensland, Australia, December 2–6, 2018. Springer, Heidelberg, Germany.
- BHSB19. Michael Backes, Lucjan Hanzlik, and Jonas Schneider-Bensch. Membership privacy for fully dynamic group signatures. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 2181–2198, London, UK, November 11–15, 2019. ACM Press.
- BL22. Renas Bacho and Julian Loss. On the adaptive security of the threshold BLS signature scheme. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *ACM CCS 2022*, pages 193–207, Los Angeles, CA, USA, November 7–11, 2022. ACM Press.
- BLL⁺19. Xavier Bultel, Pascal Lafourcade, Russell W. F. Lai, Giulio Malavolta, Dominique Schröder, and Sri Aravinda Krishnan Thyagarajan. Efficient invisible and unlinkable sanitizable signatures. In Dongdai Lin and Kazue Sako, editors, *PKC 2019, Part I*, volume 11442 of *LNCS*, pages 159–189, Beijing, China, April 14–17, 2019. Springer, Heidelberg, Germany.
- BLS01. 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, Gold Coast, Australia, December 9–13, 2001. Springer, Heidelberg, Germany.

- BLS04. Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. *Journal of Cryptology*, 17(4):297–319, September 2004.
- Bol03. Alexandra Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme. In Yvo Desmedt, editor, *PKC 2003*, volume 2567 of *LNCS*, pages 31–46, Miami, FL, USA, January 6–8, 2003. Springer, Heidelberg, Germany.
- BP23. Luis Brandão and Rene Peralta. Nist first call for multi-party threshold schemes. Online, 2023.
- BSS02. Emmanuel Bresson, Jacques Stern, and Michael Szydlo. Threshold ring signatures and applications to ad-hoc groups. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 465–480, Santa Barbara, CA, USA, August 18–22, 2002. Springer, Heidelberg, Germany.
- CDLP22. Aisling Connolly, Jérôme Deschamps, Pascal Lafourcade, and Octavio Perez-Kempner. Protego: Efficient, revocable and auditable anonymous credentials with applications to hyperledger fabric. In Takanori Isobe and Santanu Sarkar, editors, *Progress in Cryptology - INDOCRYPT 2022 - 23rd International Conference on Cryptology in India, Kolkata, India, December 11-14, 2022, Proceedings*, volume 13774 of *Lecture Notes in Computer Science*, pages 249–271. Springer, 2022.
- CGH⁺23. Sofía Celi, Scott Griffy, Lucjan Hanzlik, Octavio Perez Kempner, and Daniel Slamanig. Sok: Signatures with randomizable keys. *Cryptology ePrint Archive*, Paper 2023/1524, 2023.
- CHY05. Sherman S. M. Chow, Lucas C. K. Hui, and S. M. Yiu. Identity based threshold ring signature. In Choon-sik Park and Seongtaek Chee, editors, *Information Security and Cryptology - ICISC 2004*, pages 218–232, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.
- CKLM14. Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Sarah Meiklejohn. Malleable signatures: New definitions and delegatable anonymous credentials. In Anupam Datta and Cedric Fournet, editors, *CSF 2014 Computer Security Foundations Symposium*, pages 199–213, Vienna, Austria, July 19–22, 2014. IEEE Computer Society Press.
- CKM23a. Elizabeth C. Crites, Chelsea Komlo, and Mary Maller. Fully adaptive schnorr threshold signatures. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part I*, volume 14081 of *LNCS*, pages 678–709, Santa Barbara, CA, USA, August 20–24, 2023. Springer, Heidelberg, Germany.
- CKM⁺23b. Elizabeth C. Crites, Chelsea Komlo, Mary Maller, Stefano Tessaro, and Chenzhi Zhu. Snowblind: A threshold blind signature in pairing-free groups. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part I*, volume 14081 of *LNCS*, pages 710–742, Santa Barbara, CA, USA, August 20–24, 2023. Springer, Heidelberg, Germany.
- CKP⁺23. Elizabeth Crites, Markulf Kohlweiss, Bart Preneel, Mahdi Sedaghat, and Daniel Slamanig. Threshold structure-preserving signatures. In Jian Guo and Ron Steinfeld, editors, *Advances in Cryptology - ASIACRYPT 2023*, pages 348–382, Singapore, 2023. Springer Nature Singapore.
- CL19. Elizabeth C. Crites and Anna Lysyanskaya. Delegatable anonymous credentials from mercurial signatures. In Mitsuru Matsui, editor, *CT-RSA 2019*, volume 11405 of *LNCS*, pages 535–555, San Francisco, CA, USA, March 4–8, 2019. Springer, Heidelberg, Germany.
- CL21. Elizabeth C. Crites and Anna Lysyanskaya. Mercurial signatures for variable-length messages. *PoPETs*, 2021(4):441–463, October 2021.
- CL24a. Yi-Hsiu Chen and Yehuda Lindell. Feldman’s verifiable secret sharing for a dishonest majority. *Cryptology ePrint Archive*, Paper 2024/031, 2024.
- CL24b. Yi-Hsiu Chen and Yehuda Lindell. Optimizing and implementing fishlin’s transform for uc-secure zero-knowledge. *Cryptology ePrint Archive*, Paper 2024/526, 2024.
- CLPK22. 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, Virtual Event, March 8–11, 2022. Springer, Heidelberg, Germany.
- CP93. David Chaum and Torben P. Pedersen. Wallet databases with observers. In Ernest F. Brickell, editor, *CRYPTO’92*, volume 740 of *LNCS*, pages 89–105, Santa Barbara, CA, USA, August 16–20, 1993. Springer, Heidelberg, Germany.
- DEF⁺19. Manu Drijvers, Kasra Edalatnejad, Bryan Ford, Eike Kiltz, Julian Loss, Gregory Neven, and Igor Stepanovs. On the security of two-round multi-signatures. In *2019 IEEE Symposium on Security and Privacy*, pages 1084–1101, San Francisco, CA, USA, May 19–23, 2019. IEEE Computer Society Press.
- DHS15. David Derler, Christian Hanser, and Daniel Slamanig. A new approach to efficient revocable attribute-based anonymous credentials. In Jens Groth, editor, *15th IMA International Conference on Cryptography and Coding*, volume 9496 of *LNCS*, pages 57–74, Oxford, UK, December 15–17, 2015. Springer, Heidelberg, Germany.
- DR24. Sourav Das and Ling Ren. Adaptively secure bls threshold signatures from ddh and co-cdh. In Leonid Reyzin and Douglas Stebila, editors, *Advances in Cryptology - CRYPTO 2024*, pages 251–284, Cham, 2024. Springer Nature Switzerland.
- DS18. David Derler and Daniel Slamanig. Highly-efficient fully-anonymous dynamic group signatures. In Jong Kim, Gail-Joon Ahn, Seungjoo Kim, Yongdae Kim, Javier López, and Taesoo Kim, editors, *ASIACCS 18*, pages 551–565, Incheon, Republic of Korea, April 2–6, 2018. ACM Press.

- DS19. David Derler and Daniel Slamanig. Key-homomorphic signatures: definitions and applications to multiparty signatures and non-interactive zero-knowledge. *Designs, Codes and Cryptography*, 87(6):1373–1413, 2019.
- Fel87. Paul Feldman. A practical scheme for non-interactive verifiable secret sharing. In *28th FOCS*, pages 427–437, Los Angeles, CA, USA, October 12–14, 1987. IEEE Computer Society Press.
- FHKS16. Georg Fuchsbauer, Christian Hanser, Chethan Kamath, and Daniel Slamanig. Practical round-optimal blind signatures in the standard model from weaker assumptions. In Vassilis Zikas and Roberto De Prisco, editors, *SCN 16*, volume 9841 of *LNCS*, pages 391–408, Amalfi, Italy, August 31 – September 2, 2016. Springer, Heidelberg, Germany.
- FHS15. Georg Fuchsbauer, Christian Hanser, and Daniel Slamanig. Practical round-optimal blind signatures in the standard model. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 233–253, Santa Barbara, CA, USA, August 16–20, 2015. Springer, Heidelberg, Germany.
- FHS19. 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.
- Fis05. Marc Fischlin. Communication-efficient non-interactive proofs of knowledge with online extractors. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 152–168, Santa Barbara, CA, USA, August 14–18, 2005. Springer, Heidelberg, Germany.
- FS87. Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO’86*, volume 263 of *LNCS*, pages 186–194, Santa Barbara, CA, USA, August 1987. Springer, Heidelberg, Germany.
- GK96. Oded Goldreich and Ariel Kahan. How to construct constant-round zero-knowledge proof systems for NP. *Journal of Cryptology*, 9(3):167–190, June 1996.
- Han23. Lucjan Hanzlik. Non-interactive blind signatures for random messages. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part V*, volume 14008 of *LNCS*, pages 722–752, Lyon, France, April 23–27, 2023. Springer, Heidelberg, Germany.
- HKSS22. Abida Haque, Stephan Krenn, Daniel Slamanig, and Christoph Striecks. Logarithmic-size (linkable) threshold ring signatures in the plain model. In Goichiro Hanaoka, Junji Shikata, and Yohei Watanabe, editors, *PKC 2022, Part II*, volume 13178 of *LNCS*, pages 437–467, Virtual Event, March 8–11, 2022. Springer, Heidelberg, Germany.
- HRS15. Christian Hanser, Max Rabkin, and Dominique Schröder. Verifiably encrypted signatures: Security revisited and a new construction. In Günther Pernul, Peter Y. A. Ryan, and Edgar R. Weippl, editors, *ESORICS 2015, Part I*, volume 9326 of *LNCS*, pages 146–164, Vienna, Austria, September 21–25, 2015. Springer, Heidelberg, Germany.
- HS14. Christian Hanser and Daniel Slamanig. Structure-preserving signatures on equivalence classes and their application to anonymous credentials. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part I*, volume 8873 of *LNCS*, pages 491–511, Kaoshiung, Taiwan, R.O.C., December 7–11, 2014. Springer, Heidelberg, Germany.
- HS20. Abida Haque and Alessandra Scafuro. Threshold ring signatures: New definitions and post-quantum security. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *PKC 2020, Part II*, volume 12111 of *LNCS*, pages 423–452, Edinburgh, UK, May 4–7, 2020. Springer, Heidelberg, Germany.
- HS21. 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, Virtual Event, Republic of Korea, November 15–19, 2021. ACM Press.
- Kat23. Jonathan Katz. Round optimal fully secure distributed key generation. Cryptology ePrint Archive, Paper 2023/1094, 2023.
- Lab20. Protocol Labs. Blast: Multilingual bls12-381 signature library. Online, 2020.
- Lab21. Protocol Labs. High performance implementation of bls12 381. Online, 2021.
- LWW04. Joseph K. Liu, Victor K. Wei, and Duncan S. Wong. A separable threshold ring signature scheme. In Jong-In Lim and Dong-Hoon Lee, editors, *Information Security and Cryptology - ICISC 2003*, pages 12–26, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.
- MBG⁺23. 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, *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security, CCS 2023, Copenhagen, Denmark, November 26-30, 2023*, pages 30–44. ACM, 2023.
- MHOY21. Alexander Munch-Hansen, Claudio Orlandi, and Sophia Yakoubov. Stronger notions and a more efficient construction of threshold ring signatures. In Patrick Longa and Carla Ràfols, editors, *LATINCRYPT 2021*, volume 12912 of *LNCS*, pages 363–381. Springer, Heidelberg, Germany, October 6–8, 2021.

- MMS⁺24. 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, *Public-Key Cryptography – PKC 2024*, pages 163–195, Cham, 2024. Springer Nature Switzerland.
- Nan24. Masaya Nanri. Implementation of interactive threshold mercurial signatures. Online, 2024.
- NRS21. Jonas Nick, Tim Ruffing, and Yannick Seurin. MuSig2: Simple two-round Schnorr multi-signatures. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part I*, volume 12825 of *LNCS*, pages 189–221, Virtual Event, August 16–20, 2021. Springer, Heidelberg, Germany.
- Ped91. Torben P. Pedersen. A threshold cryptosystem without a trusted party (extended abstract) (rump session). In Donald W. Davies, editor, *EUROCRYPT’91*, volume 547 of *LNCS*, pages 522–526, Brighton, UK, April 8–11, 1991. Springer, Heidelberg, Germany.
- Ped92. Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Joan Feigenbaum, editor, *CRYPTO’91*, volume 576 of *LNCS*, pages 129–140, Santa Barbara, CA, USA, August 11–15, 1992. Springer, Heidelberg, Germany.
- PM24. Colin Putman and Keith M. Martin. Selective delegation of attributes in mercurial signature credentials. In Elizabeth A. Quaglia, editor, *Cryptography and Coding*, pages 181–196, Cham, 2024. Springer Nature Switzerland.
- PS16. David Pointcheval and Olivier Sanders. Short randomizable signatures. In Kazue Sako, editor, *CT-RSA 2016*, volume 9610 of *LNCS*, pages 111–126, San Francisco, CA, USA, February 29 – March 4, 2016. Springer, Heidelberg, Germany.
- PS18. David Pointcheval and Olivier Sanders. Reassessing security of randomizable signatures. In Nigel P. Smart, editor, *CT-RSA 2018*, volume 10808 of *LNCS*, pages 319–338, San Francisco, CA, USA, April 16–20, 2018. Springer, Heidelberg, Germany.
- Sch91. Claus-Peter Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, January 1991.
- ST24. Olivier Sanders and Jacques Traoré. Compact issuer-hiding authentication, application to anonymous credential. *Proc. Priv. Enhancing Technol.*, 2024(3):645–658, 2024.
- TZ22. Stefano Tessaro and Chenzhi Zhu. Short pairing-free blind signatures with exponential security. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part II*, volume 13276 of *LNCS*, pages 782–811, Trondheim, Norway, May 30 – June 3, 2022. Springer, Heidelberg, Germany.

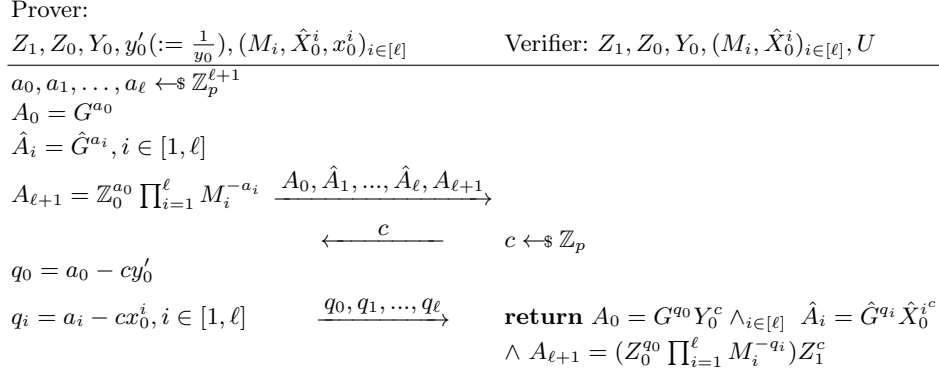


Fig. 13. ZKPoK protocol for $\pi_0^{(2)}$.

threshold key structure as well its relation with non-interactive blind signatures for random messages [Han23]. For (threshold) blind signatures the complexity of our interactive signing protocol scales linearly with the number of parties making it less attractive than non-interactive constructions such as the one from [CKP⁺23]. However, efficiency of [CKP⁺23] comes at the cost of introducing new security assumptions and requires the ROM. Hence, our approach could be of interest in cases where solutions in the standard model are preferred.

B.2 Verifiably Encrypted Signatures

Hanser *et al.* [HRS15] gave a black-box construction of verifiably encrypted signatures [BGLS03] and public-key encryption from EQS. Our work is compatible with theirs and could add a layer of privacy. Considering the application of contract signing protocols [BGLS03], users could prove that they obtained a legitimate signature from some valid contractor, without revealing whom. Looking at public-key encryption, besides the single party case outlined in [HRS15], parties could generate ciphertexts cooperatively.

B.3 Threshold Ring Signatures

A relatively long line of work studied the case of ring signatures in the threshold setting (*e.g.*, [BSS02, CHY05, LWW04, MHOY21, HS20, HKSS22]) and it continues to be an active area of research ([AHAN⁺22, ABF23]). In this regard, our public key unlinkability notion ensures that given a converted signature that verifies under a randomized public key, no set of $t-1$ parties can link it with the original global verification key while keeping the anonymity of the threshold ring setting. Moreover, the special case of multi-signatures ($t = n$) could also enable interesting use cases if instead of running a DKG protocol, each user picks her public key independently. In such setting, the global verification key is just the aggregation of each public key and one can obtain anonymous multi-signatures under our n -unlinkability notion. This latter case also generalizes the idea for anonymous credentials discussed in Section 4.4, allowing users to prove that they got a signature involving certain set of users.