Interactive Threshold Mercurial Signatures and Applications

Masaya Nanri
nanri.masaya.26n@st.kyoto-u.ac.jp
Kyoto University
Kyoto, Japan

Octavio Perez Kempner
octavio.perezkempner@ntt.com
NTT Social Informatics Laboratories
Tokyo, Japan

Mehdi Tibouchi
mehdi.tibouchi@ntt.com
NTT Social Informatics Laboratories and Kyoto University
Tokyo, Japan

Masayuki Abe
msyk.abe@ntt.com
NTT Social Informatics Laboratories and Kyoto University
Tokyo, Japan

ABSTRACT

Equivalence class signatures allow a controlled form of malleability based on equivalence classes defined over the message space. As a result, signatures can be publicly randomized and adapted to a new message representative in the same equivalence class. Notably, security requires that an adapted signature-message pair looks indistinguishable from a random signature-message pair in the space of valid signatures for the new message representative. Together with the decisional Diffie-Hellman assumption, this yields an unlinkability notion (class-hiding), making them a very attractive building block for privacy-preserving primitives.

Mercurial signatures are an extension of equivalence class signatures that allow malleability for the key space. Unfortunately, the most efficient construction to date suffers a severe limitation that limits their application: only a weak form of public key class-hiding is supported. In other words, given knowledge of the original signing key and randomization of the corresponding public key, it is possible to identify whether they are related.

In this work, we put forth the notion of *interactive threshold mercurial signatures* and show how they help to overcome the abovementioned limitation. Moreover, we present constructions in the two-party and multi-party settings, assuming at least one honest signer. We also discuss related applications, including blind signatures, multi-signatures, and threshold ring signatures. To showcase the practicality of our approach, we implement the proposed constructions, comparing them against related alternatives.

KEYWORDS

Equivalence class signatures, mercurial signatures, multi-signatures, threshold signatures, unlinkability.

1 INTRODUCTION

Equivalence Class signatures (EQS) [HS14, FHS19] are malleable signatures [CKLM14] defined over a vector of group elements. Moreover, they are structure-preserving [AFG+10, AGHO11] and, thus, inherit their benefits. 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]. The latter can be seen as an extension of EQS. Not only do they allow one to randomize a signature and adapt it to a new message but also to a new public key

Informally speaking, all EQS require an adapted signature to look like a freshly computed one (signature adaption) and some class-hiding notion when adapting messages and keys (also referred to as unlinkability). For most 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. Furthermore, a recent work on EQS by Bauer and Fuchsbauer [BF20] proposed a construction achieving a stronger notion of message class-hiding, covering the case in which the adversary knows the discrete logarithms of the message vector. Hence, for message class-hiding, all possible scenarios are well-studied. Unfortunately, the situation for public key class-hiding is different as the adversary usually knows the corresponding discrete logarithms (*i.e.,* secret key). Moreover, none of the constructions that have been proposed so far achieves an analogous stronger class-hiding notion for the public key. To couple with this, existing works introduce trust assumptions that weaken their anonymity guarantees. 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]. In the issuer-hiding setting, the issuer has to be trusted since given a valid key pair (sk,pk) and a randomized public key pk' of pk, the owner of sk can determine whether pk' is related to pk. Put differently, 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 with partial trust, it can suffice to fully de-anonymize users in others. For delegatable credentials, the situation is even worse as all users in a credential chain have to be trusted. Otherwise, an adversary that corrupts an user somewhere in the delegation chain can tell apart corrupted and honest chains by recognizing randomizations of its own key.

In this work, we propose the notion of *interactive Threshold Mercurial Signature* (TMS). Unlike Threshold Structure-Preserving Signatures (TSPS) [CKP+23] that are not randomizable nor consider any equivalence class (see Sec. 1.2 for a detailed discussion), TMS schemes are full MS schemes (*i.e.,* EQS signatures that support all kinds of randomizability) whose signatures are computed in a distributed manner with a quorum of signers. However, they can still be verified in the same way as the original mercurial signatures. Therefore, they possess the adaptable malleability akin to mercurial signatures, enabling them to transition seamlessly into the threshold setting. To begin with, we introduce the two-party case, the

most straightforward and comprehensible way to grab our idea. Subsequently, we present how to generalize our ideas to n parties. We also discuss related applications for each setting.

From a technical point of view, we offer a modular design. In particular, the security and efficiency of our constructions depend on how Zero-Knowledge Proofs of Knowledge are instantiated, and different alternatives are discussed. Considering our interactive signing approach, our technique combines polynomial secret sharing for long-term secrets and multiplicative sharing for ephemeral random factors using a sequential blinding and unblinding computation. While we are able to port MS into the threshold setting, 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. Our 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.

1.1 Our Contributions

We formalize TMS and propose a construction with a two-party distributed signing protocol that allows parties to obtain a signature that verifies under a secret-shared public key. This way, no single party can retrieve the corresponding secret key. A direct application is the construction of anonymous credentials with stronger issuer-hiding features, solving an open problem in the EQS domain (see Sec. 6.1 for details). We also discuss the general threshold case and propose a second construction. Alternative approaches and their efficiency are also covered for both cases. Before concluding, we present an implementation prototype and related benchmarks to show the feasibility of our strategy. Finally, we discuss future directions, including challenges and alternative security models.

The first MS scheme in the literature (and the only one that remains secure) was proposed by Crites and Lysyanskaya in [CL19]. It is exactly the same as the EQS proposed by Fuchsbauer, Hanser, and Slamanig [FHS19]. The key observation from [CL19] was that, given a valid signature, the same randomization procedure that was used to randomize a message-signature pair in [FHS19], could be used to randomize a public key-signature pair. In fact, both can be done simultaneously. Crites and Lysyanskaya formalized all the required security properties, proving that the scheme from [FHS19] was also secure with respect to randomization of the public key. Our TMS constructions are distributed signing protocols for the same scheme and, thus, work as a drop-in replacement for the EQS from [FHS19] and the MS from [CL19]. Notably, by distributing the signing protocol one can obtain signatures that verify under additive shares instead of a single multiplicative share in the exponent. This allows us to obtain stronger security notions, covering a wider range of applications for this type of signatures. Furthermore, the threshold nature of our signing protocol can be used to replace a root authority in delegatable schemes such as [CL21] by a quorum of authorities. Consequently, our work broadens the scope of privacy-preserving applications covered by EQS/MS.

1.2 Related Work

1.2.1 Mercurial Signatures. There are two constructions of MS in the literature: one by Crites and Lysyanskaya [CL19] and another by Connolly *et al.* [CLPK22]. Unfortunately, the MS from [CLPK22] was recently shown to be flawed in [BF24] and it's broken. In Sec. 2.3 we recall the construction from [CL19]. As previously 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 of it 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]. However, we do so in a way 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 are able to provide a stronger class hiding notion as further discussed in Sec. 6.

1.2.2 Pointcheval-Sanders signatures. Very recently, Sanders and Traoré [ST23] 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 so that it verifies 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 [ST23], 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 wider range of functionalities.

1.2.3 Threshold Signatures. The ongoing NIST standardization effort related to threshold signatures [NIS23] 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, DR23]. Threshold versions of BLS require a distributed key generation protocol (see *e.g.,* [GJKR07]) but can 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 the security of their construction. 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 given 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 and consider an interactive signing process. As we show, our interactive process 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. Interestingly, our multi-party TMS can also be seen as a *threshold ring signature* [BSS02].

1.2.4 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.

1.3 Organization

We give the preliminaries in Section 2. The syntax and security properties of TMS are presented in Section 3. Our interactive signing protocol is first discussed for two parties in Section 4. In Section 5, we present the general threshold case. Applications are discussed in Section 6. We report our experimental evaluation in Section 7 before concluding in Section 8. A detailed presentation of zero-knowledge proofs used in this work can be found in Appendix A.

2 PRELIMINARIES

Notation. The set of integers $1, 2, ..., n$ 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 $\mathsf{pp} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, P_1, P_2, e)$ describing an asymmetric bilinear group where $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are cyclic groups of prime order p with $\lceil \log_2 p \rceil = \kappa$, P_1 and P_2 are generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is an efficiently computable (non-degenerate) bilinear map. pp is considered Type-III if no efficiently computable isomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$ is known.

DDH Assumption. Let BGGen be a bilinear-group generator that outputs $\mathsf{pp}=(p,\mathbb{G}_1,\mathbb{G}_2,\mathbb{G}_T,P_1,P_2,e)$. The *decisional Diffie-Hellman assumption* holds relative to $\mathbb{G}_i$ for BGGen, if for all p.p.t adversaries $\mathcal{A}$ the following probability is negligible,

$$\Pr\left[\begin{array}{l} \mathsf{pp} \leftarrow\!\$\ \mathsf{BGGen}(1^\kappa); r, s, t \leftarrow\!\$\ \mathbb{Z}_p; b \leftarrow\!\$\ \{0,1\} \\ b^* \leftarrow\!\$\ \mathcal{A}(\mathsf{pp}, P_i^r, P_i^s, P_i^{(1-b)t+brs}) \end{array} : \ b^* = b \right] - \frac{1}{2}$$

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. We refer the reader to [Gol01] for related background.

2.2 Mercurial Signatures

As previously mentioned, MS are EQS that also 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 2.1 (**Mercurial signature** [CL19]).* A MS scheme for parametrized equivalence relations $\mathcal{R}_m, \mathcal{R}_{\mathsf{pk}}, \mathcal{R}_{\mathsf{sk}}$ is a tuple of the following polynomial-time algorithms, which are deterministic algorithms unless otherwise stated:

$\mathsf{PGen}(1^\kappa) \to \mathsf{pp}$: On input the security parameter 1^κ, this probabilistic algorithm outputs the public parameters pp. This includes parameters for the parameterized equivalence relations $\mathcal{R}_m$, $\mathcal{R}_{\mathsf{pk}}$, and $\mathcal{R}_{\mathsf{sk}}$ so they are well-defined. It also includes parameters for the algorithms sample_ρ and sample_μ, which sample key and message converters, respectively.
$\mathsf{KGen}(\mathsf{pp}, \ell) \to (\mathsf{pk}, \mathsf{sk})$: On input the public parameters pp and a length parameter ℓ, this probabilistic 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 $(\mathsf{pk}, \mathsf{sk}) \in \mathsf{KGen}(\mathsf{pp}, \ell)$ if there exists a set of random choices that KGen could make to output (pk, sk).
$\mathsf{Sign}(\mathsf{pp}, \mathsf{sk}, m) \to \sigma$: On input the signing key sk and a message $m \in \mathcal{M}$, this probabilistic algorithm outputs a signature σ.
$\mathsf{Verify}(\mathsf{pp}, m, \sigma, \mathsf{pk}) \to 0/1$: On input the public key pk, a message $m \in \mathcal{M}$, and a purported signature σ, output 0 or 1.
$\mathsf{ConvertSK}(\mathsf{sk}, \rho) \to \mathsf{sk}'$: On input sk and a key converter $\rho \in \mathsf{sample}_\rho$, output a new secret key $\mathsf{sk}' \in [\mathsf{sk}]_{\mathcal{R}_{\mathsf{sk}}}$.
$\mathsf{ConvertPK}(\mathsf{pk}, \rho) \to \mathsf{pk}'$: On input pk and a key converter $\rho \in \mathsf{sample}_\rho$, output a new public key $\mathsf{pk}' \in [\mathsf{pk}]_{\mathcal{R}_{\mathsf{pk}}}$.
$\mathsf{ConvertSig}(\mathsf{pk}, m, \sigma, \rho) \to \sigma'$: On input pk, a message $m \in \mathcal{M}$, a signature σ, and key converter $\rho \in \mathsf{sample}_\rho$, this probabilistic algorithm returns a new signature σ'.
$\mathsf{ChgRep}(\mathsf{pk}, m, \sigma, \mu) \to (m', \sigma')$: On input pk, a message $m \in \mathcal{M}$, a signature σ, and a message converter $\mu \in \mathsf{sample}_\mu$, this probabilistic algorithm computes a new message $m' \in [m]_{\mathcal{R}_m}$ and a new signature σ' and outputs (m', σ').

*Definition 2.2 (**Correctness** [CL19]).* A MS scheme for parameterized equivalence relations $\mathcal{R}_m, \mathcal{R}_{\mathsf{pk}}, \mathcal{R}_{\mathsf{sk}}$ is correct if it satisfies

the following conditions for all κ, for all $\mathsf{pp} \in \mathsf{PGen}(1^\kappa)$, for all $\ell > 1$, for all $(\mathsf{pk}, \mathsf{sk}) \in \mathsf{KGen}(\mathsf{pp}, \ell)$:

Verification. $\forall\, m \in \mathcal{M}, \forall\, \sigma \in \mathsf{Sign}(sk, m), \mathsf{Verify}(\mathsf{pk}, m, \sigma) = 1$.

Key conversion. $\forall\, \rho \in \mathsf{sample}_\rho$, $(\mathsf{ConvertPK}(\mathsf{pk}, \rho), \mathsf{ConvertSK}(\mathsf{sk}, \rho)) \in \mathsf{KGen}(\mathsf{pp}, \ell)$. Moreover, $\mathsf{ConvertSK}(\mathsf{sk}, \rho) \in [\mathsf{sk}]_{\mathcal{R}_{\mathsf{sk}}}$ and $\mathsf{ConvertPK}(\mathsf{pk}, \rho) \in [\mathsf{pk}]_{\mathcal{R}_{\mathsf{pk}}}$.

Signature conversion. $\forall\, m \in \mathcal{M}, \forall\, \sigma$ such that $\mathsf{Verify}(\mathsf{pk}, m, \sigma) = 1, \forall\, \rho \in \mathsf{sample}_\rho, \forall\, \sigma' \in \mathsf{ConvertSig}(\mathsf{pk}, m, \sigma, \rho), \mathsf{Verify}(\mathsf{ConvertPK}(\mathsf{pk}, \rho), m, \sigma') = 1$.

Change of message representative. $\forall\, m \in \mathcal{M}, \forall\, \sigma$ such that $\mathsf{Verify}(\mathsf{pk}, m, \sigma) = 1, \forall\, \mu \in \mathsf{sample}_\mu, \mathsf{Verify}(\mathsf{pk}, m, \sigma') = 1$, where $(m', \sigma') = \mathsf{ChgRep}(\mathsf{pk}, m, \sigma, \mu)$. Moreover, $m \in \mathcal{R}_m$.

*Definition 2.3 (**Unforgeability** [CL19]).* A MS scheme for parameterized equivalence relations $\mathcal{R}_m, \mathcal{R}_{\mathsf{pk}}, \mathcal{R}_{\mathsf{sk}}$ is unforgeable if for all polynomial-length parameters $\ell(\kappa)$ and all PPT adversary $\mathcal{A}$ having access to a signing oracle, the following probability is negligible,

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

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

*Definition 2.4 (**Class-Hiding** [CL19]).* 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 $\mathbf{Adv}^{\mathsf{MSG\text{-}CH}}_{\mathsf{MS},\mathcal{A}}(\kappa) := 2 \cdot \Pr\left[\mathbf{Exp}^{\mathsf{MSG\text{-}CH}}_{\mathsf{MS},\mathcal{A}}(\kappa) \Rightarrow \mathsf{true}\right] - 1 = \epsilon(\kappa)$, where $\mathbf{Exp}^{\mathsf{MSG\text{-}CH}}_{\mathsf{MS},\mathcal{A}}(\kappa)$ is shown in Fig. 1.

Public key class-hiding: if the advantage of any PPT adversary $\mathcal{A}$ defined by $\mathbf{Adv}^{\mathsf{PK\text{-}CH}}_{\mathsf{MS},\mathcal{A}}(\kappa) := 2 \cdot \Pr\left[\mathbf{Exp}^{\mathsf{PK\text{-}CH}}_{\mathsf{MS},\mathcal{A}}(\kappa) \Rightarrow \mathsf{true}\right] - 1 = \epsilon(\kappa)$, where $\mathbf{Exp}^{\mathsf{PK\text{-}CH}}_{\mathsf{MS},\mathcal{A}}(\kappa)$ is shown in Fig. 1.

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

$\mathsf{pp} \leftarrow\$ \mathsf{PGen}(1^\kappa); b \leftarrow\$ \{0,1\}; m_1 \leftarrow\$ \mathcal{M}; m_2^0 \leftarrow\$ \mathcal{M}; m_2^1 \leftarrow\$ [m]_{\mathcal{R}_m}$

$b' \leftarrow\$ \mathcal{A}(\mathsf{pp}, m_1, m_2^b)$; **return** $b = b'$

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

$\mathsf{pp} \leftarrow\$ \mathsf{PGen}(1^\kappa); b \leftarrow\$ \{0,1\}; \rho \leftarrow\$ \mathsf{sample}_\rho(\mathsf{pp})$

$(\mathsf{sk}_1, \mathsf{pk}_1) \leftarrow\$ \mathsf{KGen}(\mathsf{pp}, \ell(\kappa)); (\mathsf{sk}_2^0, \mathsf{pk}_2^0) \leftarrow\$ \mathsf{KGen}(\mathsf{pp}, \ell(\kappa))$

$\mathsf{pk}_2^1 \leftarrow \mathsf{ConvertPK}(\mathsf{pk}_1, \rho); \mathsf{sk}_2^1 \leftarrow \mathsf{ConvertSK}(\mathsf{sk}_1, \rho)$

$b' \leftarrow\$ \mathcal{A}^{\mathsf{Sign}(\mathsf{sk}_1, \cdot), \mathsf{Sign}(\mathsf{sk}_2^b, \cdot)}(\mathsf{pk}_1, \mathsf{pk}_2^b)$; **return** $b = b'$

Oracle $\mathsf{Sign}(\mathsf{sk}, m)$: **return** $\mathsf{Sign}(\mathsf{sk}, m)$

Figure 1: Class-hiding experiments from [CL19].

*Definition 2.5 (**Origin-hiding** [CL19]).* A MS scheme is origin-hiding if for all κ, $\mathsf{pp} \in \mathsf{PGen}(1^\kappa)$, pk^*, m, and σ, the following two properties hold:

(1) if $\mathsf{Verify}(\mathsf{pk}, m, \sigma) = 1$ and $\mu \leftarrow\$ \mathsf{sample}_\mu$, then $\mathsf{ChgRep}(\mathsf{pk}^*, m, \sigma, \mu)$ outputs a uniformly random $m' \in [m]_{\mathcal{R}_m}$ and uniformly random $\sigma' \in \{\hat{\sigma} | \mathsf{Verify}(\mathsf{pk}^*, m', \hat{\sigma}) = 1\}$.

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

2.3 Construction From [CL19]

The MS by Crites and Lysyanskaya [CL19] 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 $(\mathbb{G}_1^*)^\ell$ where ℓ is the length of the message vector. We recall that all elements of a vector $(M)_{i\in[\ell]} \in (\mathbb{G}_1^*)^\ell$ share different mutual ratios that depend on their discrete logarithms. Hence, it is possible to partition $(\mathbb{G}_1^*)^\ell$ into equivalence classes given by the following relation:

$$\mathcal{R} = \{(M, M') \in (\mathbb{G}_1^*)^\ell \times (\mathbb{G}_1^*)^\ell | \exists s \in \mathbb{Z}_p^* : M' = M^s\} \subseteq (\mathbb{G}_1^*)^\ell$$

The construction from [CL19], which we present below, is based on the observation that an analogous relation can be defined for the public keys, inducing equivalence classes on the key space as well.

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

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

$\mathsf{Sign}(\mathsf{pp}, \mathsf{sk}, M) \rightarrow \sigma$: $y \leftarrow\$ \mathbb{Z}_p^*$; $Z \leftarrow (\Pi_{i=1}^\ell M_i^{x_i})^y$; $Y \leftarrow P^{\frac{1}{y}}$; $\hat{Y} \leftarrow \hat{P}^{\frac{1}{y}}$; **return** $(Z, Y, \hat{Y})$.

$\mathsf{Verify}(\mathsf{pp}, M, \sigma, \mathsf{pk} = (\hat{X})_{i\in[\ell]}) \rightarrow 0/1$: **return** $\Pi_{i=1}^\ell e(M_i, \hat{X}_i) = e(Z, \hat{Y}) \wedge e(Y, \hat{P}) = e(P, \hat{Y})$.

$\mathsf{ConvertSK}(\mathsf{sk}, \rho) \rightarrow \mathsf{sk}'$: $\mathsf{sk}' \leftarrow \mathsf{sk}^\rho$; **return** sk'.

$\mathsf{ConvertPK}(\mathsf{pk}, \rho) \rightarrow \mathsf{pk}'$: $\mathsf{pk}' \leftarrow \mathsf{pk}^\rho$; **return** pk'.

$\mathsf{ConvertSig}(\mathsf{pk}, M, \sigma, \rho) \rightarrow \sigma'$: $\psi \leftarrow\$ \mathbb{Z}_p^*$; **return** $(Z^{\psi\rho}, Y^{\frac{1}{\psi}}, \hat{Y}^{\frac{1}{\psi}})$.

$\mathsf{ChgRep}(\mathsf{pk}, M, \sigma, \mu) \rightarrow (M', \sigma')$: $\psi \leftarrow\$ \mathbb{Z}_p^*; M' \leftarrow M^\mu; \sigma' \leftarrow (Z^{\psi\mu}, Y^{\frac{1}{\psi}}, \hat{Y}^{\frac{1}{\psi}})$; **return** (M', σ').

In the following, we recall the security statements of MS from [CL19] where the security of our TMS is reduced.

Theorem 2.6 (Unforgeability [CL19]). *The* MS *from [CL19] is unforgeable in the generic group model for Type-III bilinear groups.*

Theorem 2.7 (Class-hiding [CL19]). *The* MS *from [CL19] is message class-hiding for $\boldsymbol{G}$ and is public key class-hiding in the generic group model for Type-III bilinear groups if the DDH assumption holds in $\boldsymbol{G}$.*

Theorem 2.8 (Origin-hiding [CL19]). *The* MS *from [CL19] is origin-hiding in the generic group model for Type-III bilinear groups.*

3 THRESHOLD MERCURIAL SIGNATURES

3.1 Syntax

We present the syntax and security properties of (interactive) TMS following the notation from [CKP+23].

*Definition 3.1 (**Threshold mercurial signature**).* A TMS scheme is a MS scheme where KGen and Sign, are replaced with:

$\mathsf{TKGen}(\mathsf{pp}, \ell, t, n) \rightarrow (\vec{\mathsf{sk}}, \vec{\mathsf{pk}}, \mathsf{pk})$: On input the public parameters pp, a length parameter ℓ, and two integers $t, n \in \mathsf{poly}(1^\kappa)$ such that $1 \le t \le n$, this probabilistic algorithm outputs two vectors of size n of signing and public keys along with the global (threshold) public key pk. Both, the signing keys $\vec{\mathsf{sk}} = (\mathsf{sk}_1, ..., \mathsf{sk}_n)$ and the public keys $\vec{\mathsf{pk}} = (\mathsf{pk}_1, ..., \mathsf{pk}_n)$ are distributed among parties such that

party P_i gets $(\mathsf{sk}_i, \vec{\mathsf{pk}}, \mathsf{pk})$. The message space $\mathcal{M}$ is well-defined from pp and ℓ.

$\mathsf{TSign}(\mathsf{pp}, \{\mathsf{sk}_j\}_{j\in\mathcal{J}}, m) \to \sigma$: On input $\{sk_j\}_{j\in\mathcal{J}}$ for some $\mathcal{J} \subseteq [n]$ of size $\geq t$ and a message $m \in \mathcal{M}$, this probabilistic algorithm is an interactive protocol run by a set of parties in $\mathcal{J}$. At the end of the protocol 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).

3.2 Security Properties

Throughout the paper, the key generation is done by a single trusted party. Alternatively, decentralized discrete-log key generation protocols (DKGs), e.g., [Fel87, Ped91, AF04, CL24a], are available with various security properties. We refer to a recent survey on the long history and state of the art about secure DKG protocols [Kat23]. Since we consider static corruption, the security property we require for the key generation is the presence of a static simulator.

Definition 3.2 (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 $\mathsf{pp} \in \mathsf{PGen}(1^\kappa)$, $\ell \in \mathbb{N}$, $(\mathsf{pk}, \mathsf{sk}) \in \mathsf{KGen}(\mathsf{pp}, \ell)$, $t, n \in \mathbb{N}$, $C \subsetneq [n]$ of size $t-1$, $\mathsf{SimTKGen}(\mathsf{pk}, n, C)$ outputs $\{\mathsf{sk}_j\}_{j\in C}$ and $\{\mathsf{pk}_j\}_{j\in[n]}$. The joint distribution of $(\mathsf{pk}, \{\mathsf{pk}_j\}_{j\in[n]}, \{\mathsf{sk}_j\}_{j\in C})$ is indistinguishable from that of $\mathsf{TKGen}(\mathsf{pp}, \ell, t, n)$.

Correctness of TMS follows the usual notion (see Section 2). Below we present the corresponding definition for completeness.

*Definition 3.3 (**Correctness**).* A (n,t)-TMS is correct if it satisfies the following conditions for all κ, $\mathsf{pp} \in \mathsf{PGen}(1^\kappa)$, $\ell > 1$, $(\vec{\mathsf{sk}}, \vec{\mathsf{pk}}, \mathsf{pk}) \in \mathsf{TKGen}(\mathsf{pp}, \ell, t, n)$ and $\mathcal{J} \subseteq [n]$ of size $\geq t$:

Verification. $\forall\, m \in \mathcal{M}, \forall\, \sigma \in \mathsf{TSign}(\mathsf{pp}, \{(j, \mathsf{sk}_j)\}_{j\in\mathcal{J}}, M)$, $\mathsf{Verify}(\mathsf{pp}, m, \sigma, \mathsf{pk}) = 1$.

Key conversion. $\forall\, \rho \in \mathsf{sample}_\rho$, $(\mathsf{ConvertTSK}(\mathsf{sk}, \rho), \mathsf{ConvertTPK}(\mathsf{pk}, \rho), \mathsf{ConvertPK}(\mathsf{pk}, \rho)) \in \mathsf{TKGen}(\mathsf{pp}, \ell, t, n)$.

Signature conversion. $\forall\, m \in \mathcal{M}, \forall\, \sigma$ such that $\mathsf{Verify}(\mathsf{pp}, \mathsf{pk}, m, \sigma) = 1, \forall\, \rho \in \mathsf{sample}_\rho, \forall\, \sigma' \in \mathsf{ConvertSig}(\mathsf{pk}, m, \sigma, \rho), \mathsf{Verify}(\mathsf{ConvertPK}(\mathsf{pk}, \rho), m, \sigma') = 1$.

Change of message representative. $\forall\, m \in \mathcal{M}, \forall\, \sigma$ such that $\mathsf{Verify}(\mathsf{pp}, m, \sigma, \mathsf{pk}) = 1$, $\forall\, \mu \in \mathsf{sample}_\mu$, $\mathsf{Verify}(\mathsf{pp}, m, \sigma, \mathsf{pk}') = 1$, where $(m', \sigma') = \mathsf{ChgRep}(\mathsf{pk}, m, \sigma, \mu)$. Moreover, $m \in \mathcal{R}_m$.

For unforgeability and unlinkability (class-hiding) we follow the definitions from [CL19], adapting them to the threshold setting (the adversary could corrupt up to $t-1$ parties). Unlike [CKP+23] whose unforgeability definition is in the non-interactive setting and thus lets the adversary obtain partial signatures of honest signers, we let the adversary query signatures for a set of signers of her choice but assume at least one honest signer.

*Definition 3.4 (**Unforgeability**).* A TMS scheme is unforgeable w.r.t. equivalence classes if the advantage of any PPT adversary $\mathcal{A}$ defined by $\mathbf{Adv}^{\mathsf{UNF}}_{\mathsf{TMS},\ell,t,n}(1^\kappa, \mathcal{A}) := \Pr\left[\mathbf{Exp}^{\mathsf{UNF}}_{\mathsf{TMS},\ell,t,n}(1^\kappa, \mathcal{A}) \Rightarrow \mathsf{true}\right] \leq \epsilon(\kappa)$, where $\mathbf{Exp}^{\mathsf{UNF}}_{\mathsf{TMS},\ell,t,n}(1^\kappa, \mathcal{A})$ is shown in Fig. 2.

Unlike the original class-hiding definition for mercurial signatures (Fig. 1), 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. Our approach is to relax the definition from [CGH+23] and 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.

*Definition 3.5 (**Public Key Unlinkability**).* A TMS scheme is public key unlinkable if the advantage of any PPT adversary $\mathcal{A}$ defined by $\mathbf{Adv}^{\mathsf{PK\text{-}UNL}}_{\mathsf{TMS},\ell,t,n}(1^\kappa, \mathcal{A}) := 2\cdot\Pr\left[\mathbf{Exp}^{\mathsf{PK\text{-}UNL}}_{\mathsf{TMS},\ell,t,n}(1^\kappa, \mathcal{A}) \Rightarrow \mathsf{true}\right] - 1 \leq \epsilon(\kappa)$, where $\mathbf{Exp}^{t-\mathsf{PK\text{-}UNL}}_{\mathsf{TMS},\ell,t,n}(1^\kappa, \mathcal{A})$ is shown in Fig. 3.

Note that, at $n = t = 1$, the above seamlessly gives the notion of public key unlinkability for MS. It is implied by the public key class-hiding and fulfilled by the instantiation of MS in [CL19]. We will use these facts to prove public key unlinkability.

Experiment $\mathbf{Exp}^{\mathsf{UNF}}_{\mathsf{TMS},\ell,t,n}(1^\kappa, \mathcal{A})$

$\Sigma \leftarrow \emptyset$; $\mathsf{pp} \leftarrow\!\!\$\ \mathsf{PGen}(1^\kappa)$; $(C, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{pp})$

if $C \notin [n] \vee |C| > t-1$ **return** $\bot$

$\mathcal{H} \leftarrow [n] \setminus C$; $(\vec{\mathsf{sk}}, \vec{\mathsf{pk}}, \mathsf{pk}) \leftarrow\!\!\$\ \mathsf{TKGen}(\mathsf{pp}, \ell, t, n)$

$(m^*, \sigma^*, \rho^*) \leftarrow\!\!\$\ \mathcal{A}^{\mathsf{OTSign}(\mathsf{sk}, \cdot, \cdot)}(\mathsf{st}, \{\mathsf{sk}_i\}_{i\in C}, \vec{\mathsf{pk}}, \mathsf{pk})$

return $(m^*, \sigma^*) \notin \Sigma \wedge \forall\, m \in \Sigma : [m]_{\mathcal{R}_m} \neq [m^*]_{\mathcal{R}_m}$

$\wedge\ \mathsf{Verify}(m^*, \sigma^*, \mathsf{ConvertPK}(\mathsf{pk}, \rho^*))$

Oracle $\mathsf{OTSign}(\mathsf{sk}, m, \mathcal{T})$

if $|\mathcal{T}| \neq t \vee \mathcal{T} \cap \mathcal{H} = \emptyset$ **return** $\bot$

$\sigma \leftarrow\!\!\$\ \mathsf{TSign}(\{\vec{\mathsf{sk}}_j\}_{j\in\mathcal{T}}, m)$; $\Sigma \leftarrow \Sigma \cup \{(m, \sigma)\}$; **return** σ

Figure 2: Unforgeability experiment. C and $\mathcal{H}$ are the sets of corrupt and honest signers, respectively.

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

$\mathsf{st} \leftarrow \emptyset$; $b \leftarrow\!\!\$\ \{0,1\}$; $\rho \leftarrow\!\!\$\ \mathsf{sample}_\rho$; $\mathsf{pp} \leftarrow\!\!\$\ \mathsf{PGen}(1^\kappa)$

$(C, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{st}, \mathsf{pp})$; **if** $C \notin [n] \vee |C| > t-1$ **return** $\bot$

$(\vec{\mathsf{sk}}^{\,i}, \vec{\mathsf{pk}}^{\,i}, \mathsf{pk}^i) \leftarrow\!\!\$\ \mathsf{TKGen}(\mathsf{pp}, \ell, t, n)$ **for** $i \in \{0,1\}$

$\mathsf{pk}' \leftarrow \mathsf{ConvertPK}(\mathsf{pk}^b, \rho)$

$b' \leftarrow\!\!\$\ \mathcal{A}^{\mathsf{OTSign}_b(\cdot,\cdot,\cdot)}(\mathsf{st}, \{\vec{\mathsf{sk}}^{\,i}_j\}^{i\in\{0,1\}}_{j\in C}, \mathsf{pk}', \mathsf{pk}^0, \mathsf{pk}^1)$; **return** $b = b'$

Oracle $\mathsf{OTSign}_b(m, \mathsf{pk}, \mathcal{T})$

if $|\mathcal{T}| \neq t \vee \mathsf{pk} \notin \{\mathsf{pk}', \mathsf{pk}^0, \mathsf{pk}^1\}$ **return** $\bot$

if $\mathsf{pk} = \mathsf{pk}'$ **then**

  $\sigma^i \leftarrow \mathsf{TSign}(\{\vec{\mathsf{sk}}^{\,i}_j\}_{j\in\mathcal{T}}, M)$ **for** $i \in \{0,1\}$

  if $\sigma^0 = \bot$ or $\sigma^1 = \bot$ **return** $\bot$

  else return $\mathsf{ConvertSig}(\mathsf{pk}^b, M, \sigma^b, \rho)$

else if $\mathsf{pk} = \mathsf{pk}^i$ **return** $\mathsf{TSign}(\{\vec{\mathsf{sk}}^{\,i}_j\}_{j\in\mathcal{T}}, M)$

Figure 3: Public key unlinkability experiment.

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 at the end of the present section.

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. Most importantly, the resulting signature has the same structure as the one from [CL19] and it can work as a drop-in replacement. We assume at least one honest party to achieve stronger security guarantees.

4.1 Construction

We assume that $\mathsf{PGen}(1^\kappa)$ and $\mathsf{TKGen}(\mathsf{pp}, \ell, 2, 2)$ are run honestly. Every signer j is given $\mathsf{pp} := (\mathbb{G}_1, \mathbb{G}_2, P, \hat{P}, e, p)$, $\vec{\mathsf{pk}} := \{\hat{P}^{x_j^i}\}_{j\in[0,1]}^{i\in[\ell]}$, pk, and $\vec{sk}_j := \{x_j^i\}_{i\in[\ell]}$. 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. 4, 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 $Y = P^{\frac{1}{y}}$ and $\hat{Y} = \hat{P}^{\frac{1}{y}}$ for $y = y_0y_1$ is done straightforwardly in sequence. Since at least one of the parties is honest, y will be a random value as in the original scheme. Furthermore, we observe that because EQS can be publicly randomized, our signing protocol does not need to execute a DKG protocol to remove potential bias. Instead, at the end of the computation, each user can independently randomize the signature to refresh the randomness.

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 lift Z_0 with y_0 and y_1 in sequence. This seemingly works, but we do not know how to prove its security. The difficulty is that 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.

Our approach to getting around the above problem is to blind Z_1 by using $Y_0 = P^{\frac{1}{y_0}}$, 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 $(Y_0^r)^{y_0} = (P^{\frac{r}{y_0}})^{y_0} = P^r$. Thus, P_1, who holds r, can easily unblind Z_0 by multiplying P^{-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.

- $\pi_0^{(1)} := \mathsf{ZKPoK}[\ y_0 \ :\ P = Y_0^{y_0} \wedge \hat{P} = \hat{Y}_0{}^{y_0}]$: this ZKPoK can easily be implemented using the Chaum-Pedersen protocol [CP93] and in a non-interactive way via de Fiat-Shamir transform [FS87].
- $\pi_1^{(1)} := \mathsf{ZKPoK}[\ (r, \{x_1^i\}_{i\in\ell})\ :\ Z_1 = Y_0^r \prod_{i=1}^{\ell} M_i^{x_1^i} \wedge \hat{X}_1^i = \hat{P}^{x_1^i}]$: same as above.
- $\pi_0^{(2)} := \mathsf{ZKPoK}[\ (y_0, \{x_0^i\}_{i\in[\ell]})\ : Z_0 = (Z_1 \cdot \prod_{i=i}^{\ell} M_i^{x_0^i})^{y_0} \wedge\ P = Y_0^{y_0} \wedge_{i\in[\ell]}\ \hat{X}_0^i = \hat{P}^{x_0^i}]$: This statement has a witness product in the exponent so we cannot apply the previous approach. However, the following statement is equivalent and can easily be implemented with known techniques.

$$\mathsf{ZKPoK}[\ (\{x_{0,i}\}_{i\in[\ell]}, y_0, t)\ :\ U = Y_0^t Z_1 \prod_{i=1}^{l} M_i^{x_0^i} \wedge\ Z_0 = P^{-t}U^{y_0} \wedge\ P = Y_0^{y_0}\ \wedge_{i\in[\ell]}\ \hat{X}_0^i = \hat{P}^{x_0^i}]$$

- $\pi_1^{(2)} := \mathsf{ZKPoK}[\ (r, y_1)\ :\ Z = (Z_0P^{-r})^{y_1} \wedge Y = Y_0^{\frac{1}{y_1}}]$: in this case we also have a product of r and y_1. Fortunately, we can directly translate it into $\mathsf{ZKPoK}[\ (y_1', r)\ :\ Z^{y_1'}P^r = Z_0 \wedge\ Y = Y_0^{y_1'}\ ]$ for $y_1' = \frac{1}{y_1}$ without introducing an intermediate variable.

Each proof is verified by its 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 extra three exponentiations in $\mathbb{G}_1$ for blinding and unblinding.

Table 1 presents the computational and communication costs of each ZKPoK in terms of scalar values and group elements when instantiating them using sigma protocols. We defer the full presentation of each protocol to Appendix A. We note that ℓ is usually instantiated for short vectors. Considering the applications covered in Section 6, $\ell = 2$ 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

We consider *stand-alone*, allowing us to instantiate efficient NIZKPoK's in the ROM.Alternatively, the ZKPoK's can be instantiated via interactive five-round PoK's in the standard model [GK96]. We restrict ourselves to *static* corruptions. Security under *concurrent* executions or *adaptive* corruptions would require straight-line extraction. We leave the evaluation of potential alternatives, such as using Fichlin's transform [Fis05, CL24b] for future research.

Theorem 4.1 (Correctness). *For any* $\mathsf{pp} := (\mathbb{G}_1, \mathbb{G}_2, P, \hat{P}, e, p) \in \mathsf{PGen}(1^\kappa)$, $\ell > 1$, $\vec{\mathsf{sk}} := \{x_j^i\}_{j\in[0,1]}^{i\in[\ell]}$, $\vec{\mathsf{pk}} := \{P^{x_j^i}\}_{j\in[0,1]}^{i\in[\ell]}$, *and* $\mathsf{pk} := \{P^{(x_0^i+x_1^i)}\})_{i\in[\ell]}$ *generated by* $\mathsf{TKGen}(\mathsf{pp}, \ell, 2, 2)$, *and* $M \in \mathbb{G}_1^\ell$, $\mathsf{TSign}(\mathsf{pp}, \{x_j^i\}_{j\in\{0,1\}}^{i\in[\ell]}, M)$ *in Fig. 4 outputs signature* σ *that distributes the same as* $\mathsf{Sign}(\mathsf{pp}, \mathsf{sk}, M)$ *for* $\mathsf{sk} := (x_i)_{i\in[\ell]} = (x_0^i + x_1^i)_{i\in[\ell]}$ *if both* P_0 *and* P_1 *are honest.*

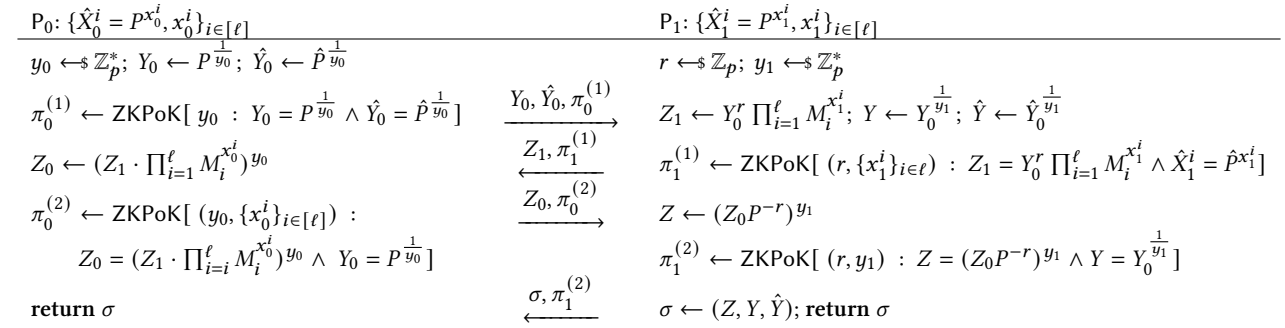

Figure 4: $\mathsf{TSign}(\mathsf{pp}, \{x_j^i\}_{j\in\{0,1\},i\in[\ell]}, M)$

	Exp.	Comm.
$\pi_0^{(1)}$	$3\lvert\mathbb{G}_1\rvert + 3\lvert\mathbb{G}_2\rvert$	$1\lvert\mathbb{G}_1\rvert + 1\lvert\mathbb{G}_2\rvert + 2\lvert\mathbb{Z}_p\rvert$
$\pi_1^{(1)}$	$(2\ell+3)\lvert\mathbb{G}_1\rvert + 3\ell\lvert\mathbb{G}_2\rvert$	$1\lvert\mathbb{G}_1\rvert + \ell\lvert\mathbb{G}_2\rvert + (\ell+1)\lvert\mathbb{Z}_p\rvert$
$\pi_0^{(2)}$	$(2\ell+11)\lvert\mathbb{G}_1\rvert + 3\ell\lvert\mathbb{G}_2\rvert$	$3\lvert\mathbb{G}_1\rvert + \ell\lvert\mathbb{G}_2\rvert + (\ell+2)\lvert\mathbb{Z}_p\rvert$
$\pi_1^{(2)}$	$8\lvert\mathbb{G}_1\rvert$	$2\lvert\mathbb{G}_1\rvert + 3\lvert\mathbb{Z}_p\rvert$

Table 1: Cost of each ZKPoK **protocol.**

Proof. Observe that $Y = P^{\frac{1}{y_0 y_1}}$, $\hat{Y} = \hat{P}^{\frac{1}{y_0 y_1}}$, and

$$\begin{aligned} Z = (Z_0 P^{-r})^{y_1} &= \{(Z_1 \cdot \textstyle\prod_{i=i}^{\ell} M_i^{x_{0,i}})^{y_0} P^{-r}\}^{y_1} \\ &= \{(Y_0^r \textstyle\prod_{i=1}^{\ell} M_i^{x_{1,i}} \cdot \prod_{i=i}^{\ell} M_i^{x_{0,i}})^{y_0} P^{-r}\}^{y_1} \\ &= \left\{\left(P^{\frac{1}{y_0} r} \textstyle\prod_{i=1}^{\ell} M_i^{(x_{0,i}+x_{1,i})}\right)^{y_0} P^{-r}\right\}^{y_1} \\ &= \left\{\left(P^{\frac{1}{y_0} r}\right)^{y_0} P^{-r} \left(\textstyle\prod_{i=1}^{\ell} M_i^{(x_{0,i}+x_{1,i})}\right)^{y_0}\right\}^{y_1} \\ &= \left(\textstyle\prod_{i=1}^{\ell} M_i^{(x_{0,i}+x_{1,i})}\right)^{y_0 y_1} \end{aligned}$$

hold. Thus, for $x_i = x_{0,i} + x_{1,i}$ and $y = y_0 y_1$, the resulting signature is $(Z, Y, \hat{Y}) = (\left(\prod_{i=1}^{\ell} M_i^{x_i}\right)^{y}, P^{1/y}, \hat{P}^{1/y})$. 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 the same as the original Mercurial signature generated with the stated global keys. □

In mercurial signatures [CL19], unforgeability (Definition 2.3) is proved by contradiction. If there was a PPT algorithm that can break unforgeability through accessing signing oracle, there is a reduction to one that can break 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 3.4 as it can be run between the adversary and the environment. We have three cases for any adversary $\mathcal{A}$:

(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].
(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 extracted through ZKPoK's.

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 4.2 (Unforgeability). *Our* $(2,2)$-TMS *scheme 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 2, we construct a simulator that plays the role of the adversary in the unforgeability game against MS as in Definition 2.3. Let $(\mathsf{sk}, \mathsf{pk}) := (\{x^i\}_{i\in[\ell]}, \{\hat{X}^i\}_{i\in[\ell]})$ be a key pair of MS generated by $\mathsf{KGen}(\mathsf{pp}, \ell)$. Given pp as input, the simulator first invokes $\mathcal{A}$ and outputs C obtained from $\mathcal{A}$. Here, C is either 0 or 1 meaning P_0 or P_1 is corrupted, respectively. Then, given pk as input, the simulator executes $\mathsf{SimTKGen}(\mathsf{pk}, 2, C)$ to obtain $\mathsf{sk}_j := \{x_j^i\}_{i\in[\ell]}$ for $j \in C$ and $\mathsf{pk}_j := \{\hat{X}_j^i\}_{i\in[\ell]}$ for $j \in [n]$. Shared signing keys $\{x_j^i\}_{i\in[\ell]}$ for $j \notin C$ 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 $\{\mathsf{sk}_j\}_{j\in C}$, $\{\mathsf{pk}_j\}_{j\in[n]}$, 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. 5. The case in which P_1 is corrupted is shown in Fig. 6.

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. 5), the real computation of Z_1 and the simulated one in the first round

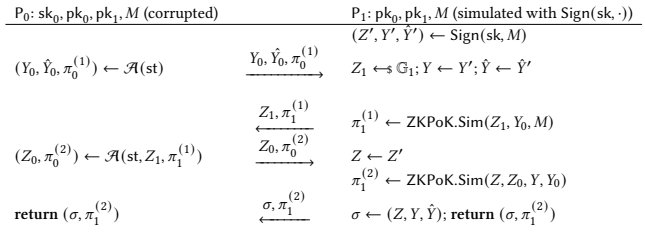

Figure 5: Simulator's algorithm considering corruption of P_0.

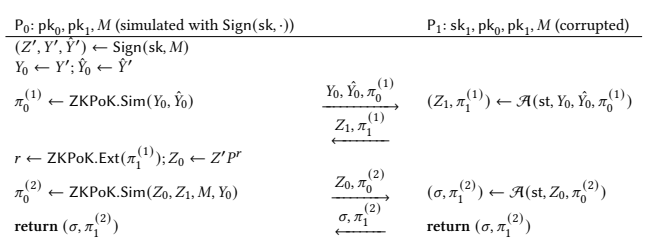

Figure 6: Simulator's algorithm considering corruption of P_1.

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 next analyze the second case where P_1 is corrupted (Fig. 6). During the first round, Y_0 and $\hat{Y}_0$ distributes 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 $Z_1 = Y_0^r \prod_{i=1}^{\ell} M_i^{x_1^i}$ with the extracted random factor 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 $y = y_0 = \log_P Y'$, we have: $Z_0 = Z'P^r = (\prod_{i=1}^{\ell} M_i^{x_0^i + x_1^i})^{y_0} P^r = (Y_0^r \prod_{i=1}^{\ell} M_i^{x_1^i})^{y_0} (\prod_{i=1}^{\ell} M_i^{x_0^i})^{y_0} = (Z_1 \prod_{i=1}^{\ell} M_i^{x_0^i})^{y_0}$.

Accordingly, the simulation of P_0 is perfect modulus the knowledge soundness of $\pi_1^{(1)}$ and zero-knowledge of $\pi_0^{(1)}$ and $\pi_0^{(2)}$.

Finally, the simulator outputs whatever $\mathcal{A}$ outputs at the end. As TKGen is assumed secure and the honest party within OTSign is correctly simulated, the view of $\mathcal{A}^{\mathsf{OTSign}}$ is indistinguishable from that of the real unforgeability game in the presence of corrupt party C. Thus, whenever $\mathcal{A}$ is successful in forging TMS, so does the simulator in forging MS. This concludes the proof. □

As for unforgeability, we prove unlinkability assuming at least one honest signer and considering a signing oracle in the presence of the corrupted party ($\mathcal{A}$ can corrupt any party of its choice).

Theorem 4.3 (Public Key Unlinkability). *Our* $(2, 2)$-TMS *scheme 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-hiding and public key class-hiding.*

Proof. The proof strategy is almost the same as that in the proof of Theorem 4.2. 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. Given pp as input, the simulator invokes $\mathcal{A}$ and outputs C obtained from $\mathcal{A}$. Then, given $(\mathsf{pk}', \mathsf{pk}^0, \mathsf{pk}^1)$ as input, the simulator runs SimTKGen twice for pk^0 and pk^1 with C) to obtain, for $i = \{0, 1\}$, sk_j^i for $j \in C$ and pk_j^i for $j \in [n]$. The simulator then invokes $\mathcal{A}$ with $\{\mathsf{sk}_j^i\}_{j\in C}^{i\in\{0,1\}}$, $\{\mathsf{pk}_j^i\}_{j\in[n]}^{i\in\{0,1\}}$, and pk' 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' and M, the simulator first makes two queries to its oracle on pk^0 and pk^1 with the same M. It uses the obtained MS signatures to simulate two invocations of TSign on pk^0 and pk^1 with M as explained in the proof of Theorem 4.2. The validity of this part of the simulation is due to the security of ZKPoK's as before. If either of TSign simulation results in $\perp$ (due to misbehavior of a corrupted party), it returns $\perp$. Otherwise, it queries its oracle on pk' with M and returns the obtained signature to $\mathcal{A}$. This part of the simulation is perfect due to the origin-hiding property of MS.

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. □

5 THRESHOLD CASE

This section describes our protocol for the threshold case where the keys are distributed in a (t, n)-threshold manner among n users $\mathsf{P}_{u_1}, \ldots, \mathsf{P}_{u_n}$. As before, we assume that the key generation is done by a single honest party and only describes the syntax and the relation satisfied by the keys. Our focus is the threshold signing protocol where a subset of users $\mathsf{P}_{u_1}, \ldots, \mathsf{P}_{u_t}$ engage to produce a signature. The overall structure of the protocol follows that of the two-party case; parties work in sequence, from P_{u_1} to P_{u_t}. The first P_{u_1} and the last P_{u_t} do the same as P_0 and P_1 did in the two-party case, respectively. However, each intermediate participant, $\mathsf{P}_{u_2}, \ldots, \mathsf{P}_{u_{t-1}}$, must independently take on the roles of P_0 and P_1, bringing forth new considerations to both the protocol and its security analysis. Importantly, our signing protocol assumes the existence of a broadcast channel and all communication between parties is done using said channel.

5.1 Construction

For given t and n that $1 \le t \le n$, TKGen generates local key pairs $\mathsf{sk}_j := (x_j^1, \ldots, x_j^\ell)$, $\mathsf{pk}_j := (\hat{P}^{x_j^1}, \ldots, \hat{P}^{x_j^\ell})$ for $j \in [n]$, and global verification key. The global signing key is implicitly set to $\mathsf{sk} := (x_1, \ldots, x_\ell)$ where each x_i is shared into $(x_1^i, \ldots, x_n^i)$ by (t, n)-threshold scheme over $\mathbb{Z}_p$. For any set of indices, $J \subseteq [n]$, of size t, it

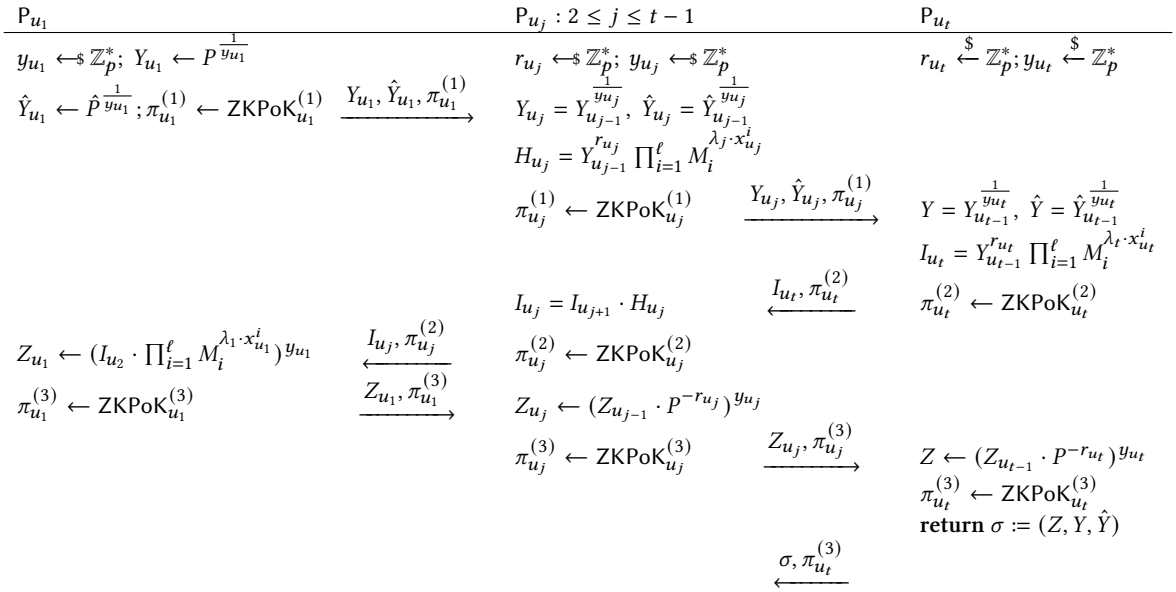

Figure 7: t-party protocol for $\mathsf{TSign}(\mathsf{pp}, \{\mathsf{sk}_j\}_{j\in J}, M)$.

holds that $x_i = \sum_{j\in J} \lambda_j x_j^i \bmod p$ where λ_j is a Lagrange coefficient defined as $\lambda_j := \prod_{t\in J\setminus\{j\}} \frac{t}{t-j} \bmod p$.

Let $J = (u_1, \ldots, u_t) \subseteq [n]$ be a size-t subset of signers engaging in $\mathsf{TSign}(\mathsf{pp}, \{\mathsf{sk}_j\}_{j\in J}, m)$, presented in Fig. 7. We follow the template of the two-party case, which operates sequentially. The initial party u_1 communicates with the first intermediate party (u_2), and all intermediate parties behave the same until the last one (u_{t-1}) communicates with the final party u_t. The protocol proceeds backward until the u_1 is reached. Subsequently, u_1 triggers the last round, which concludes when u_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:

- $\mathsf{ZKPoK}_{u_1}^{(1)}[y_{u_1} : Y_{u_1} = P^{\frac{1}{y_{u_1}}} \wedge \hat{Y_{u_1}} = \hat{P}^{\frac{1}{y_{u_1}}}]$
- $\mathsf{ZKPoK}_{u_j}^{(1)}[y_{u_j} : Y_{u_j} = Y_{u_{j-1}}^{\frac{1}{y_{u_j}}} \wedge \hat{Y}_{u_j} = \hat{Y}_{u_{j-1}}^{\frac{1}{y_{u_j}}}]$
- $\mathsf{ZKPoK}_{u_t}^{(2)}[(r_{u_t}, \{x_{u_t}^i\}_{i\in[\ell]}) : I_{u_t} = Y_{u_{t-1}}^{r_{u_t}} \prod_{i=1}^{\ell} M_i^{\lambda_t \cdot x_{u_t}^i} \wedge_{i\in[\ell]} X_{u_t}^i = \hat{P}^{x_{u_t}^i}]$
- $\mathsf{ZKPoK}_{u_j}^{(2)}[(r_{u_j}, \{x_{u_j}^i\}_{i\in[\ell]}) : I_{u_j} = I_{u_{j+1}} \cdot Y_{u_{j-1}}^{r_{u_j}} \prod_{i=1}^{\ell} M_i^{\lambda_j \cdot s_{u_j}^i} \wedge_{i\in[\ell]} X_{u_j}^i = \hat{P}^{x_{u_j}^i}]$
- $\mathsf{ZKPoK}_{u_1}^{(3)}[(\{x_{u_1}^i\}_{i\in[\ell]}, y_{u_1}) : Z_{u_1} = (I_{u_2} \cdot \prod_{i=i}^{\ell} M_i^{\lambda_1 \cdot x_{u_1}^i})^{y_{u_1}} \wedge Y_{u_1}^{y_{u_1}} = P \wedge_{i\in[\ell]} X_{u_1}^i = \hat{P}^{s_{u_1}^i}]$
- $\mathsf{ZKPoK}_{u_j}^{(3)}[y_{u_j} : Z_{u_j} = (Z_{u_{j-1}} \cdot P^{-r_{u_j}})^{y_{u_j}} \wedge Y_{u_j}^{y_{u_j}} = Y_{u_{j-1}}]$
- $\mathsf{ZKPoK}_{u_t}^{(3)}[y_{u_t} : Z = (Z_{u_{t-1}} \cdot P^{-r_{u_t}})^{y_{u_t}} \wedge Y^{y_{u_t}} = Y_{u_{t-1}}]$

An obvious difference from the two-party case is the presence of the intermediate parties, $\mathsf{P}_{u_2}, \ldots, \mathsf{P}_{u_{t-1}}$. Computing $Y = P^{\frac{1}{y}}$ and $\hat{Y} = \hat{P}^{\frac{1}{y}}$ is done sequentially from P_{u_1} to P_{u_t}, and y is defined by $y = \prod_{j=1}^{t} y_{u_j}$. If all parties are honest, the following holds for Z_{u_1}:

$$\begin{aligned} Z_{u_1} &= (I_{u_2} \cdot \prod_{i=1}^{\ell} M_i^{\lambda_1 \cdot x_{u_1}^i})^{y_{u_1}} \\ &= \left(\prod_{j=2}^{t} H_{u_j} \cdot \prod_{i=1}^{\ell} M_i^{\lambda_1 \cdot x_{u_1}^i}\right)^{y_{u_1}} \\ &= \left(P^{\frac{r_{u_2}}{y_{u_1}} + \frac{r_{u_3}}{y_{u_2} y_{u_1}} + \cdots + \frac{r_{u_t}}{y_{u_{t-1}} \cdots y_{u_1}}} \cdot \prod_{i=1}^{\ell} M_i^{\sum_{j=1}^{t} \lambda_j \cdot x_{u_j}^i}\right)^{y_{u_1}} \end{aligned} \tag{1}$$

The reason why Z_{u_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_{u_1} in the reverse order from P_{u_2} to P_{u_t}. To see that the blinding factors are canceled as expected, observe that

$$\begin{aligned} Z &= (Z_{u_{t-1}} \cdot P^{-r_{u_t}})^{y_{u_t}} \\ &= (\cdots (Z_{u_1} \cdot P^{-r_{u_2}})^{y_{u_2}} \cdots)^{y_{u_{t-1}}} \cdot P^{-r_{u_t}})^{y_{u_t}} \\ &= \left(\cdots \left(P^{\frac{r_{u_2}}{y_{u_1}} + \frac{r_{u_3}}{y_{u_2} y_{u_1}} + \cdots + \frac{r_{u_t}}{y_{u_{t-1}} \cdots y_{u_1}}} \cdot \prod_{i=1}^{\ell} M_i^{\sum_{j=1}^{t} \lambda_j \cdot x_{u_j}^i}\right)^{y_{u_1}} \right. \\ &\qquad \left. P^{-r_{u_2}})^{y_{u_2}} \cdots)^{y_{u_{t-1}}} \cdot P^{-r_{u_t}})^{y_{u_t}} \end{aligned} \tag{2}$$

holds. Concerning the exponent of P, we have:

$$\left(\cdots\left(\frac{r_{u_2}}{y_{u_1}}+\frac{r_{u_3}}{y_{u_2}y_{u_1}}+\cdots+\frac{r_{u_t}}{y_{u_{t-1}}\cdots y_{u_1}}\right)y_{u_1} - r_{u_2}\right)\cdots\Big)y_{u_{t-1}}-r_{u_t}\Big)y_{u_t}$$

$$=\left(\cdots\left(\frac{r_{u_t}}{y_{u_{t-1}}}+r_{u_{t-1}}\right)\frac{1}{y_{u_{t-2}}}\cdots\right)\frac{1}{y_{u_3}}+r_{u_3}\Big)\frac{1}{y_{u_2}} +r_{u_2}\Big)\frac{1}{y_{u_1}}\cdot y_{u_1}-r_{u_2}\Big)\cdots\Big)y_{u_{t-1}}-r_{u_t}\Big)y_{u_t} \tag{3}$$

$$=\big(0+\cdots+0\big)y_{u_t}=0.$$

Therefore:

$$Z=\left(\prod_{i=1}^{\ell}M_i^{\sum_{j=1}^{t}\lambda_j\cdot x_{u_j}^i}\right)^{\prod_{j=1}^{t}y_{u_j}}=\left(\prod_{i=1}^{\ell}M_i^{x_i}\right)^{y}. \tag{4}$$

5.2 Efficiency

The ZKPoK's are analogous to the two-party case and can be instantiated with the protocols discussed in Appendix A. We have $\pi_0^{(1)} = \mathsf{ZKPoK}_{u_1}^{(1)} = \mathsf{ZKPoK}_{u_j}^{(1)}$, $\pi_0^{(2)} = \mathsf{ZKPoK}_{u_1}^{(3)}$, $\pi_1^{(1)} = \mathsf{ZKPoK}_{u_t}^{(2)} = \mathsf{ZKPoK}_{u_j}^{(2)}$, and $\pi_1^{(2)} = \mathsf{ZKPoK}_{u_j}^{(3)} = \mathsf{ZKPoK}_{u_t}^{(3)}$. Therefore, communication and computation complexity increases linearly with t. However, in many cases, the number of signers does not grow beyond one order of magnitude so t can stay relatively small. Furthermore, considering the applications discussed in Section 6.4 which are multi-signatures and threshold ring signatures, ℓ will be small, meaning the ZKPoK's will all be efficient and short.

5.3 Security

Since the key observation for correctness is already given, we focus on unforgeability and unlinkability. For both proofs, overall strategies are unchanged from their two-party counterparts. The simulator has to deal with a situation in which at most $t-1$ signers are corrupted. The simulation strategy changes depending on which party remains honest. An essential difference from the two-party case is the presence of intermediate parties to simulate. Namely, when parties $(\mathsf{P}_{u_1},\ldots,\mathsf{P}_{u_t})\subseteq[n]$ engage in the signing protocol, we have three cases: The adversary corrupts everyone but

(1) the initial party P_{u_1} who starts the protocol, or
(2) the end party P_{u_t} who first obtains the signature, or
(3) an intermediate party P_{u_j} for $1<j<t$.

Below, we present simulation algorithms for each case individually.

Theorem 5.1 (Unforgeability). *Our (t,n)-TMS construction is unforgeable against static corruption of at most $t-1$ parties if* TKGen *is secure, all zero-knowledge proofs are secure, and the original* MS *is unforgeable.*

Proof sketch. In Fig. 8, simulators of the interactive signing oracle for all possible types of corruptions are presented. We argue why the simulator's algorithm works in each possible scenario.

- The first case simulates the initial party, P_{u_1}, in the same way as simulating P_0 in the two-party case. It however needs to extract all random factors from other parties. Provided that $\mathsf{ZKPoK}_{u_j}^{(2)}$ allows knowledge extraction, the simulation of the computation is perfect as we argued before.
- The second case simulates the end party, P_{u_t}, in the exactly same way as simulating P_1 in the two-party case. Thus, the simulation is perfect assuming the soundness of all relevant proofs from other parties.
- The last case simulates an intermediate party, P_{u_j}. It is a mixture of the above simulation strategies. Given random factors extracted from all descending parties, $\mathsf{P}_{u_{j+1}},\ldots,\mathsf{P}_{u_t}$, the simulation is perfect for the same reasons as above.

□

The following theorem can be proved in the same way as done in the two-party case except for the obvious changes in the simulation strategies addressed in the above proof of unforgeability.

Theorem 5.2 (Public Key Unlinkability). *Our (t,n)-TMS scheme is public key unlinkable against static corruption of at most $t-1$ partis if* TKGen *is secure, all* ZKPoK*'s are secure, and* MS *is origin-hiding and public key class-hiding.*

6 APPLICATIONS

We begin this section recalling that our TMS constructions work as a drop-in replacement of the original MS ([CL19]) as they share the same structure and verification. For example, in all delegatable anonymous credential schemes from MS [CL19, CL21], the root authority issues MS signatures and TMS can be used to distribute it. The same applies for EQS constructions based on [FHS19], which is the signature scheme underlying the MS construction from [CL19].

6.1 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). Subsequent works by Conolly *et al.* [CLPK22, CDLP22] extended it to consider issuer-hiding features [BEK+21, CLPK22], allowing users to even to hide the identity of their credential issuer. In particular, [CDLP22] uses the MS from [CL19] to instantiate the FHS19 framework with issuer-hiding features. Since the MS used only provides a weak issuer-hiding feature (*i.e.,* the issuers can recognize randomizations of their own public key), the ABC from [CDLP22] is limited to settings where partial trust can be tolerated.

In the following, we discuss how our $(2,2)$-TMS construction can overcome the above limitation in a setting with multiple issuers $A_1,\ldots,A_n$. Under a pseudonymous public key X_B, the idea is to let user B get a credential from issuer A_i with public key X_{A_i} by running TSign for a message m (a commitment to the user attributes as in FHS19). Let such credential be (σ,m,X_{A_iB}) (it verifies under $X_{A_iB}=X_{A_i}\cdot X_B=\hat{P}^{(x_{A_i}+x_B)}$). To validate σ, B needs to prove the correctness of X_{A_iB} with respect to X_{A_i} for some $i\in[n]$ without disclosing which one. Most importantly, even if A_i colludes with

Case 1: P_{u_1} (simulated)		$\mathsf{P}_{u_j}: 2 \le j \le t-1$ (corrupted)		P_{u_t} (corrupted)
$(Z', Y', \hat{Y}') \leftarrow \mathsf{Sign}(\mathsf{sk}, M)$				
$Y_{u_1} \leftarrow Y'; Y_{u_1} \leftarrow \hat{Y}'$				
$\pi_{u_1}^{(1)} \leftarrow \mathsf{ZKPoK.Sim}(Y_{u_1}, \hat{Y}_{u_1})$	$\xrightarrow{Y_{u_1}, \hat{Y}_{u_1}, \pi_{u_1}^{(1)}}$	$(Y_{u_j}, \hat{Y}_{u_j}, \pi_{u_j}^{(1)}) \leftarrow \mathcal{A}$	$\xrightarrow{Y_{u_j}, \hat{Y}_{u_j}, \pi_{u_j}^{(1)}}$	$(I_{u_t}, \pi_{u_t}^{(2)}) \leftarrow \mathcal{A}$
	$\xleftarrow{I_{u_j}, \pi_{u_j}^{(2)}}$	$(I_{u_j}, \pi_{u_j}^{(2)}) \leftarrow \mathcal{A}$	$\xleftarrow{I_{u_t}, \pi_{u_t}^{(2)}}$	
for $2 \le j \le k$ **do**				
$r'_{u_j} \leftarrow \mathsf{ZKPoK.Ext}(\pi_{u_j}^{(2)})$				
$r' \leftarrow \sum_{j=2}^{k} r'_{u_j}; Z_{u_1} \leftarrow Z' P^{r'}$				
$\pi_{u_1}^{(3)} \leftarrow \mathsf{ZKPoK.Sim}(Z_{u_1}, I_{u_2}, M, Y_{u_1})$				
	$\xrightarrow{Z_{u_1}, \pi_{u_1}^{(3)}}$	$(Z_{u_j}, \pi_{u_j}^{(3)}) \leftarrow \mathcal{A}$	$\xrightarrow{Z_{u_j}, \pi_{u_j}^{(3)}}$	$(\sigma, \pi_{u_t}^{(3)}) \leftarrow \mathcal{A}$; **return** (σ)
			$\xleftarrow{\sigma, \pi_{u_t}^{(3)}}$	

Case 2: P_{u_1} (corrupted)		$\mathsf{P}_{u_j}: 2 \le j \le t-1$ (corrupted)		P_{u_t} (simulated)
$(Y_{u_1}, \hat{Y}_{u_1}, \pi_{u_1}^{(1)}) \leftarrow \mathcal{A}$				$(Z', Y', \hat{Y}') \leftarrow \mathsf{Sign}(\mathsf{sk}, M)$
	$\xrightarrow{Y_{u_1}, \hat{Y}_{u_1}, \pi_{u_1}^{(1)}}$	$(Y_{u_j}, \hat{Y}_{u_j}, \pi_{u_j}^{(1)}) \leftarrow \mathcal{A}$	$\xrightarrow{Y_{u_j}, \hat{Y}_{u_j}, \pi_{u_j}^{(1)}}$	$Z \leftarrow Z'; Y \leftarrow Y'; \hat{Y} \leftarrow \hat{Y}'; I_{u_t} \leftarrow\!\!\$\ \mathbb{G}_1$
	$\xleftarrow{I_{u_j}, \pi_{u_j}^{(2)}}$	$(I_{u_j}, \pi_{u_j}^{(2)}) \leftarrow \mathcal{A}$	$\xleftarrow{I_{u_t}, \pi_{u_t}^{(2)}}$	$\pi_{u_t}^{(2)} \leftarrow \mathsf{ZKPoK.Sim}(I_{u_t}, Y_{u_{t-1}}, M))$
$(Z_{u_1}, \pi_{u_1}^{(3)}) \leftarrow \mathcal{A}$	$\xrightarrow{Z_{u_1}, \pi_{u_1}^{(3)}}$	$(Z_{u_j}, \pi_{u_j}^{(3)}) \leftarrow \mathcal{A}$	$\xrightarrow{Z_{u_j}, \pi_{u_j}^{(3)}}$	$\pi_{u_t}^{(3)} \leftarrow \mathsf{ZKPoK.Sim}(Z, Z_{u_{t-1}}, Y, Y_{u_{t-1}})$
				return $(\sigma := (Z, Y, \hat{Y}))$
			$\xleftarrow{\sigma, \pi_{u_t}^{(3)}}$	

Case 3: P_{u_1} (corrupted)		$\exists\ \mathsf{P}_{u_j}: 2 \le j \le t-1$ (simulated)		P_{u_t} (corrupted)
$(Y_{u_1}, \hat{Y}_{u_1}, \pi_{u_1}^{(1)}) \leftarrow \mathcal{A}$		$(Z', Y', \hat{Y}') \leftarrow \mathsf{Sign}(\mathsf{sk}, M)$		
	$\xrightarrow{Y_{u_1}, \hat{Y}_{u_1}, \pi_{u_1}^{(1)}}$	**if** P_{u_j} is simulated **then**		
		$Y_{u_j} \leftarrow Y'; \hat{Y}_{u_j} \leftarrow \hat{Y}'$		
		$\pi_{u_j}^{(1)} \leftarrow \mathsf{ZKPoK.Sim}(Y_{u_j}, \hat{Y}_{u_j})$		
		else $(Y_{u_j}, \hat{Y}_{u_j}, \pi_{u_j}^{(1)}) \leftarrow \mathcal{A}$		
			$\xrightarrow{Y_{u_j}, \hat{Y}_{u_j}, \pi_{u_j}^{(1)}}$	$(I_{u_t}, \pi_{u_t}^{(2)}) \leftarrow \mathcal{A}$
			$\xleftarrow{I_{u_t}, \pi_{u_t}^{(2)}}$	
		if P_{u_j} is simulated **then**		
		$I_{u_j} \leftarrow\!\!\$\ \mathbb{G}_1$		
		$\pi_{u_j}^{(2)} \leftarrow \mathsf{ZKPoK.Sim}(I_{u_j}, M)$		
		else $(I_{u_j}, \pi_{u_j}^{(2)}) \leftarrow \mathcal{A}$		
$(Z_{u_1}, \pi_{u_1}^{(3)}) \leftarrow \mathcal{A}$	$\xleftarrow{I_{u_j}, \pi_{u_j}^{(2)}}$			
	$\xrightarrow{Z_{u_1}, \pi_{u_1}^{(3)}}$	**if** P_{u_j} is simulated **then**		
		for $j+1 \le k \le t$ **do**		
		$r'_{u_k} \leftarrow \mathsf{ZKPoK.Ext}(\pi_{u_k}^{(2)})$		
		$r' \leftarrow \sum_{k=j+1}^{t} r'_{u_k}; Z_{u_j} \leftarrow Z' P^{r'}$		
		$\pi_{u_j}^{(3)} \leftarrow \mathsf{ZKPoK.Sim}(Z_{u_j}, Z_{u_{j-1}}, M, Y_{u_j})$		
		else $(Z_{u_j}, \pi_{u_j}^{(3)}) \leftarrow \mathcal{A}$		
			$\xrightarrow{Z_{u_j}, \pi_{u_j}^{(3)}}$	$(\sigma, \pi_{u_t}^{(3)}) \leftarrow \mathcal{A}$; **return** (σ)
			$\xleftarrow{\sigma, \pi_{u_t}^{(3)}}$	

Figure 8: Simulator's algorithm for each corruption case. Adversary $\mathcal{A}$ manages its internal state.

a verifier and gets to see (σ, m, X_{A_iB}), it should be infeasible to recognize such credential as one issued by A_i.

Our approach is to give a show proof consisting in randomizing the signature-message pair with μ (using ChgRep as in FHS19), ρ (using ConvertSig to hide X_{A_iB}), and giving a NIZK proof for statement $\{(\rho, x_B) : X_T = (X_{A_i} \cdot \hat{P}^{x_B})^{\rho}\}$. Such type of NIZK, whose idea we borrow from a very recent work on issuer-hiding anonymous credentials based on Pointcheval-Sanders signatures [ST23], can be efficiently implemented in the ROM from Schnorr proofs. Intuitively, it attest that X_B generated her credential with the authority and thus the signature is valid. Note that X_B 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 links the tuple with the issuer's public key X_{A_i}. 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, given the OR-Proof, issuers cannot link their public key with a randomized one (this was possible in all previous works from EQS[CDLP22]).

6.2 Blind Signatures

Our $(2, 2)$-TMS construction can be used to obtain blind and partially blind signatures in a black-box way using the ideas from [FHS15] and [FHKS16] since it can be seen as an EQS. Moreover, our interactive signing protocol is also compatible with (blind) signatures on random messages (parties can choose independent messages P^{m_0} and P^{m_1} to produce a signature on $P^{m_0+m_1}$). We leave it as an interesting future work exploring the advantages of instantiating blind signatures with our 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.

6.3 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.

6.4 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 6.1, allowing users to prove that the got a signature involving certain set of users.

7 EXPERIMENTAL EVALUATION

We prototyped our 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 [NPTA24]. 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 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 $\mathbf{G}_1$ and $\mathbf{G}_2$ are of size 48 and 96 bytes, respectively).

Table 2 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 small number of parties, and the additional features offered by TMS, we find the overhead compared to standard MS well justified.

A natural question that arises is to compare the performance of TMS with other primitives. In this regard, the work most closely related to ours would be the threshold SPS from [CKP+23], but, to the best of our knowledge, it has not been implemented. One could also 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.

8 CONCLUSION

In this work, we develop the notion of 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. Moreover, our experimental evaluation suggests that our constructions

Scheme	Parties	Sign($\ell = 2$)	Sign($\ell = 5$)	Sign($\ell = 10$)
MS	1	0.3	0.4	0.5
TMS	2	4.2	6.5	10.7
TMS	5	10.9	16.6	26.0
TMS	10	22.6	35.7	55.6

Table 2: Signing times in milliseconds for each scheme

.

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.

Compared to the existing threshold structure-preserving signatures for the generalized case, our construction is very competitive in terms of efficiency without requiring new assumptions, such as the indexed Diffie-Hellman message.

All in all, interactive TMS offer greater flexibility than standard mercurial signatures with relatively little overhead. Therefore, revising existing applications of mercurial signatures (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.

REFERENCES

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

[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. Springer, Heidelberg, May 2023.

[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. Springer, Heidelberg, August 2004.

[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. Springer, Heidelberg, August 2010.

[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. Springer, Heidelberg, August 2011.

[AHAN+22] Diego F. Aranha, Mathias Hall-Andersen, Anca Nitulescu, Elena Pagnin, and Sophia Yakoubov. 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. Springer, Heidelberg, March 2022.

[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. Springer, Heidelberg, August 2022.

[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. Springer, Heidelberg, December 2021.

[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. Springer, Heidelberg, September 2020.

[BF24] Balthazar Bauer and Georg Fuchsbauer. On security proofs of existing equivalence class signature schemes. Cryptology ePrint Archive, Paper 2024/183, 2024. https://eprint.iacr.org/2024/183.

[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. Springer, Heidelberg, May 2003.

[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. Springer, Heidelberg, December 2018.

[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. ACM Press, November 2019.

[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. ACM Press, November 2022.

[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. Springer, Heidelberg, April 2019.

[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. Springer, Heidelberg, December 2001.

[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. Springer, Heidelberg, January 2003.

[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. Springer, Heidelberg, August 2002.

[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. https://eprint.iacr.org/2023/1524.

[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. IEEE Computer Society Press, 2014.

[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. Springer, Heidelberg, August 2023.

[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. Springer, Heidelberg, August 2023.

[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. Springer, Heidelberg, March 2019.

[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. https://eprint.iacr.org/2024/031.

[CL24b] Yi-Hsiu Chen and Yehuda Lindell. Optimizing and implementing fischlin's transform for uc-secure zero-knowledge. Cryptology ePrint Archive, Paper 2024/526, 2024. https://eprint.iacr.org/2024/526.

[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. Springer, Heidelberg, March 2022.

[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. Springer, Heidelberg, August 1993.

[DEF+19] Manu Drijvers, Kasra Edalatnejad, Bryan Ford, Eike Kiltz, Julian Loss, Gregory Neven, and Igors Stepanovs. On the security of two-round multi-signatures. In *2019 IEEE Symposium on Security and Privacy*, pages 1084–1101. IEEE Computer Society Press, May 2019.

[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. Springer, Heidelberg, December 2015.

[DR23] Sourav Das and Ling Ren. Adaptively secure bls threshold signatures from ddh and co-cdh. Cryptology ePrint Archive, Paper 2023/1553, 2023. https://eprint.iacr.org/2023/1553.

[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. ACM Press, April 2018.

[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. IEEE Computer Society Press, October 1987.

[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. Springer, Heidelberg, August / September 2016.

[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. Springer, Heidelberg, August 2015.

[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. Springer, Heidelberg, August 2005.

[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. Springer, Heidelberg, August 1987.

[GJKR07] Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Secure distributed key generation for discrete-log based cryptosystems. *Journal of Cryptology*, 20(1):51–83, January 2007.

[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.

[Gol01] Oded Goldreich. *Foundations of Cryptography: Basic Tools*, volume 1. Cambridge University Press, Cambridge, UK, 2001.

[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. Springer, Heidelberg, April 2023.

[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. Springer, Heidelberg, March 2022.

[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. Springer, Heidelberg, September 2015.

[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. Springer, Heidelberg, December 2014.

[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. Springer, Heidelberg, May 2020.

[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. ACM Press, November 2021.

[Kat23] Jonathan Katz. Round optimal fully secure distributed key generation. Cryptology ePrint Archive, Paper 2023/1094, 2023. https://eprint.iacr.org/2023/1094.

[Lab20] Protocol Labs. Blast: Multilingual bls12-381 signature library. Online, 2020. https://github.com/supranational/blst.

[Lab21] Protocol Labs. High performance implementation of bls12 381. Online, 2021. https://github.com/filecoin-project/blstrs.

[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, October 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. Cryptology ePrint Archive, Paper 2024/445, 2024. https://eprint.iacr.org/2024/445.

[NIS23] Nist first call for multi-party threshold schemes. Online, 2023. https://csrc.nist.gov/pubs/ir/8214/c/ipd.

[NPTA24] Masaya Nanri, Octavio Perez Kempner, Mehdi Tibouchi, and Masayuki Abe. Implementation available upon request, 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 2021. Springer, Heidelberg.

[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. Springer, Heidelberg, April 1991.

[PS16] David Pointcheval and Olivier Sanders. Short randomizable signatures. In Kazue Sako, editor, *CT-RSA 2016*, volume 9610 of *LNCS*, pages 111–126. Springer, Heidelberg, February / March 2016.

[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. Springer, Heidelberg, April 2018.

[Sch91] Claus-Peter Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, January 1991.

[ST23] Olivier Sanders and Jacques Traoré. Efficient issuer-hiding authentication, application to anonymous credential. Cryptology ePrint Archive, Paper 2023/1845, 2023. https://eprint.iacr.org/2023/1845.

[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. Springer, Heidelberg, May / June 2022.

A ZERO-KNOWLEDGE PROOFS

We present protocols for each of the ZKPoK's from our two-party construction using known techniques [Sch91, CP93], which leads to efficient non-interactive instantiations in the ROM via de Fiat-Shamir transform [FS87]. Following the discussion from Section 4.1, we present $\pi_0^{(1)}(\mathsf{ZKPoK}[\ y_0\ :\ P = Y_0^{y_0} \wedge \hat{P} = \hat{Y}_0^{\ y_0}])$ in Fig. 9, $\pi_1^{(1)}(\mathsf{ZKPoK}[\ (r,$
$\{x_1^i\}_{i\in\ell})\ :\ Z_1 = Y_0^r \prod_{i=1}^{\ell} M_i^{x_1^i} \wedge \hat{X}_1^i = \hat{P}^{x_1^i}])$ in Fig. 10, $\pi_0^{(2)}(\mathsf{ZKPoK}[$

Prover: Z, Y, Z_0, Y_0, r, y'		Verifier: Z, Y, Z_0, Y_0
$a_0, a_1 \leftarrow\$ \mathbb{Z}_p; A_0 = Z^{a_0} P^{a_1}; A_1 = Y_0^{a_0}$	$\xrightarrow{A_0, A_1}$	
	$\xleftarrow{c}$	$c \leftarrow\$ \mathbb{Z}_p$
$q_0 = a_0 - cy'; q_1 = a_1 - cr$	$\xrightarrow{q_0, q_1}$	**return** $A_0 = Z^{q_0} P^{q_1} Z_0^c \wedge A_1 = Y_0^{q_0} Y^c$

Figure 12: ZKPoK **protocol for** $\pi_1^{(2)}$.

Prover: $Y_0, \hat{Y}_0, y_0$		Verifier: $Y_0, \hat{Y}_0$
$a \leftarrow\$ \mathbb{Z}_p; A = Y_0^a; \hat{A} = \hat{Y}_0{}^a$	$\xrightarrow{A, \hat{A}}$	
	$\xleftarrow{c}$	$c \leftarrow\$ \mathbb{Z}_p$
$q \leftarrow a - cy_0$	$\xrightarrow{q}$	**return** $A = Y_0^q P^c \wedge \hat{A} = \hat{Y}_0{}^q \hat{P}^c$

Figure 9: ZKPoK **protocol for** $\pi_0^{(1)}$.

Prover: $Z_1, Y_0, (M_i, \hat{X}_1^i, x_1^i)_{i\in[\ell]}, r$		Verifier: $Z_1, Y_0, (M_i, \hat{X}_1^i)_{i\in[\ell]}$
$a_0, a_1, \ldots, a_\ell \leftarrow\$ \mathbb{Z}_p^{\ell+1}$		
$A_0 = Y_0^{a_0} \prod_{i=i}^{\ell} M_i^{a_i}$		
$\hat{A}_1 = \hat{P}^{a_1}$		
$\vdots$		
$\hat{A}_\ell = \hat{P}^{a_\ell}$	$\xrightarrow{A_0, \hat{A}_1, \ldots, \hat{A}_\ell}$	
	$\xleftarrow{c}$	$c \leftarrow\$ \mathbb{Z}_p;$
$q_0 = a_0 - cr$		
$q_1 = a_1 - cx_1^1$		
$\vdots$		
$q_\ell = a_\ell - cx_1^\ell$	$\xrightarrow{q_0, q_1, \ldots, q_\ell}$	**return** $A_0 = (Y_0^{q_0} \prod_{i=i}^{\ell} M_i^{q_i}) Z_1^c$ $\wedge \hat{A}_1 = \hat{P}^{q_1} \hat{X}_1^{1^c}$ $\vdots$ $\wedge \hat{A}_\ell = \hat{P}^{q_\ell} \hat{X}_1^{\ell^c}$

Figure 10: ZKPoK **protocol for** $\pi_1^{(1)}$.

Prover: $Z_1, Z_0, Y_0, y_0, (M_i, X_0^i, x_0^i)_{i\in[\ell]}, U, t$		Verifier: $Z_1, Z_0, Y_0, (M_i, X_0^i)_{i\in[\ell]}, U$
$a_0, a_1, \ldots, a_\ell, a_{\ell+1} \leftarrow\$ \mathbb{Z}_p^{\ell+2}$		
$A_0 = Y_0^{a_0} Z_1 \prod_{i=i}^{\ell} M_i^{a_i}$		
$\hat{A}_1 = \hat{P}^{a_1}$		
$\vdots$		
$\hat{A}_\ell = \hat{P}^{a_\ell}$		
$A_{\ell+1} = P^{-a_0} U^{a_{\ell+1}}$		
$A_{\ell+2} = Y_0^{a_{\ell+1}}$	$\xrightarrow{A_0, \hat{A}_1, \ldots, \hat{A}_\ell, A_{\ell+1}, A_{\ell+2}}$	
	$\xleftarrow{c}$	$c \leftarrow\$ \mathbb{Z}_p$
$q_0 = a_0 - ct$		
$q_1 = a_1 - cx_0^1$		
$\vdots$		
$q_\ell = a_\ell - cx_0^\ell$		
$q_{\ell+1} = a_{\ell+1} - cy_0$	$\xrightarrow{q_0, q_1, \ldots, q_{\ell+1}}$	**return** $A_0 = (Y_0^{q_0} Z_1 \prod_{i=i}^{\ell} M_i^{q_i}) U^c Z^{-c}$ $\wedge \hat{A}_1 = \hat{P}^{q_1} \hat{X}_0^{1^c}$ $\vdots$ $\wedge \hat{A}_\ell = \hat{P}^{q_\ell} \hat{X}_0^{\ell^c}$ $\wedge A_{\ell+1} = P^{-q_0} U^{q_{\ell+1}} Z_0^c$ $\wedge A_{\ell+2} = Y_0^{q_{\ell+1}} P^c$

Figure 11: ZKPoK **protocol for** $\pi_0^{(2)}$.

$(\{x_{0,i}\}_{i\in[\ell]}, y_0, t) : U = Y_0^t Z_1 \prod_{i=1}^{l} M_i^{x_0^i} \wedge Z_0 = P^{-t} U^{y_0} \wedge P = Y_0^{y_0} \wedge_{i\in[\ell]} \hat{X}_0^i = \hat{P}^{x_0^i}]$), in Fig. 11, and $\pi_1^{(2)}(\mathsf{ZKPoK}[(r, y_1) : Z = (Z_0 P^{-r})^{y_1} \wedge Y = Y_0^{\frac{1}{y_1}}])$ in Fig. 12.