sciendo
Proceedings on Privacy Enhancing Technologies ; 2021 (4):441–463

Elizabeth C. Crites and Anna Lysyanskaya

Mercurial Signatures for Variable-Length Messages

Abstract: Mercurial signatures are a useful building block for privacy-preserving schemes, such as anonymous credentials, delegatable anonymous credentials, and related applications. They allow a signature σ on a message m under a public key pk to be transformed into a signature σ' on an *equivalent* message m' under an equivalent public key pk' for an appropriate notion of equivalence. For example, pk and pk' may be unlinkable pseudonyms of the same user, and m and m' may be unlinkable pseudonyms of a user to whom some capability is delegated. The only previously known construction of mercurial signatures suffers a severe limitation: in order to sign messages of length ℓ, the signer's public key must also be of length ℓ. In this paper, we eliminate this restriction and provide an interactive signing protocol that admits messages of any length. We prove our scheme existentially unforgeable under chosen open message attacks (EUF-CoMA) under a variant of the asymmetric bilinear decisional Diffie-Hellman assumption (ABDDH).

Keywords: Signature schemes, anonymous credentials.

DOI 10.2478/popets-2021-0079
Received 2021-02-28; revised 2021-06-15; accepted 2021-06-16.

1 Introduction

Suppose Alice is known by a public key pk_{Alice}, and Bob is known by a public key pk_{Bob}. Suppose also that Alice has a certificate on her public key and relevant attributes from some certification authority (CA). Attributes may include the expiration date of the certificate or information about resources to which a user has been granted access. Alice's certificate consists of her public key pk_{Alice} and attributes $attr_{Alice}$ and a signature on them from the CA: $\sigma_{CA\to Alice}$. Suppose Bob obtains a certificate from Alice, rather than directly from the CA. As a result, Bob's certificate consists of Alice's pk_{Alice} and $attr_{Alice}$ and certificate from the CA, $\sigma_{CA\to Alice}$, as well as his own public key pk_{Bob} and attributes $attr_{Bob}$ and certificate from Alice, $\sigma_{Alice\to Bob}$.

A conventional signature scheme allows Alice to certify Bob as above. However, a *mercurial signature* allows the signer, Alice, to sign a message, such as Bob's public key and attributes, with two important blinding features that make it attractive in privacy-preserving applications. The first feature is message-blinding: the original message m and its corresponding signature σ can be transformed into an equivalent message m' and corresponding signature σ'. The second feature is public key-blinding, which allows the original public key and corresponding signature to be transformed as well.

Let us see how these two privacy-preserving features may be used in the scenario above. Mercurial signatures allow Bob to transform the public keys on his certification chain and derive valid signatures for these transformed values. Specifically, he can transform pk_{Alice} into an equivalent pk'_{Alice}, where Alice's secret key will also correspond to this new public key. Bob can then adapt $\sigma_{CA\to Alice}$ into $\sigma'_{CA\to Alice}$, which is the CA's signature on the transformed public key pk'_{Alice} and attributes $attr_{Alice}$. This can be done using the message-blinding feature of mercurial signatures. Using the public key-blinding feature, Bob can also adapt $\sigma_{Alice\to Bob}$ into $\sigma'_{Alice\to Bob}$, which is a valid signature on pk_{Bob} and attributes $attr_{Bob}$ under pk'_{Alice}. He can then repeat the process to transform his own public key pk_{Bob} into an equivalent but unlinkable $\tilde{\mathsf{pk}}_{Bob}$ and derive the corresponding signature $\tilde{\sigma}_{Alice\to Bob}$. It is easy to see that this can be extended to longer certification chains. These blinding features are desirable because certificate holders do not have to disclose all of the information on their certification chains every time they use them. In particular, the public keys on certification chains are blinded, concealing the identities of the users operating under them.

Mercurial signatures were introduced in a recent paper by Crites and Lysyanskaya [15]. The construction consists of messages and public keys that are vectors of group elements of a certain fixed length. Specifically, messages and public keys are of the form $M = (M_1, \dots, M_\ell)$ and $\mathsf{pk} = (\hat{X}_1, \dots, \hat{X}_\ell)$ for a fixed length ℓ, where M and pk are defined over bilinear groups $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. Mercurial signatures allow a message M to be transformed into an equivalent message

Elizabeth C. Crites: IOHK, email: elizabeth_crites@alumni.brown.edu
Anna Lysyanskaya: Brown University, email: anna_lysyanskaya@brown.edu

$M' = (M_1^{\mu}, \ldots, M_\ell^{\mu})$ using a scalar μ, and public keys may be transformed similarly.

We present a construction of mercurial signatures that inherits this structure but allows for messages of unbounded length. A message space that consists of vectors of any length is very convenient because, in particular, it allows for signatures on public keys and any number of attributes. Consider anonymous credentials, wherein users receive credentials directly from the CA. (Such directly issued credentials are referred to as level-1 credentials.) Suppose Alice's public key is $\mathsf{pk}_{Alice} = (\hat{X}_1, \ldots, \hat{X}_\ell)$ and her attributes are some values $(a_1, \ldots, a_k)$ that represent access to a particular set of buildings at particular times. If Alice intends to reveal her attributes every time she uses her certificate, she may encode them as $(\hat{X}_1^{a_1}, \ldots, \hat{X}_1^{a_k})$ and simply append them to the vector representing her public key. (Of course, a limitation of encoding attributes this way is that they are exposed. In this paper, we do not address limited disclosure of attributes.) Her certificate is then the CA's signature on this combined vector $M = (\hat{X}_1, \ldots, \hat{X}_\ell, \hat{X}_1^{a_1}, \ldots, \hat{X}_1^{a_k})$ of length $\ell + k$. If the message is transformed into an equivalent message $M' = (\hat{X}_1^{\mu}, \ldots, \hat{X}_\ell^{\mu}, \hat{X}_1^{\mu \cdot a_1}, \ldots, \hat{X}_1^{\mu \cdot a_k})$, the attributes remain the same relative to the base $\hat{X}_1^{\mu}$, so Alice's certificate still authorizes access to the same buildings at the same times. A message space that consists of vectors of any length is also desirable because the CA does not need to know how many attributes a user has ahead of time.

Now consider *delegatable* anonymous credentials, wherein a user receives a level-L credential from a level-L-1 user. In particular, suppose Alice issues a level-2 credential to Bob that grants him access to the same buildings or a subset of the buildings to which she has access, potentially limiting the hours during which Bob is authorized. Under the mercurial signature scheme of [15], if Alices's public key is of length ℓ and her attributes are of length k, the CA's public key must be of length $\ell + k$. This, in turn, severely limits the kinds of key-attribute pairs that Alice can sign with a public key of length ℓ and Bob can sign with a public key of length $\ell - |attr_{Bob}|$ (and so on down the chain). Furthermore, while the construction of [15] permits this kind of delegation, the proofs of security do not. Delegatable anonymous credentials in [15] are proven secure only when all public keys and messages are of the same fixed length ℓ. The mercurial signature scheme presented in this work allows messages to include any number of elements.

1.1 Related Work and Applications

Our motivating application is anonymous credentials [5–8, 14, 24, 25]. In an anonymous credential system, users can obtain credentials anonymously as well as prove possession of credentials without revealing any other information (via zero-knowledge proofs). Anonymous credentials are well studied and have been incorporated into industry standards (such as the TCG standard [4]) and government policy (such as the NSTIC document released by the Obama administration[1]).

Mercurial signatures are a natural building block for anonymous credentials. In order to anonymously obtain a credential, Alice requests a signature from the CA on one of her many equivalent public keys. In order to anonymously use her credential, Alice blinds her public key and the CA's signature and gives a zero-knowledge proof of knowledge (ZKPoK) of the secret key corresponding to her public key. Crucially, it is difficult to distinguish whether or not a pair of public keys (and thus identities) are equivalent.

Mercurial signatures are used as a building block for even more interesting applications, such as delegatable anonymous credentials [15]. In this setting, a participant may use her credential anonymously as well as anonymously delegate it to others, all while remaining oblivious to the true identities of the users on her credential chain. All prior constructions of delegatable anonymous constructions relied on costly non-interactive zero-knowledge (NIZK) proofs [2, 12, 13], such as Groth-Sahai proofs [23], which made them too inefficient for practical use. (Some required hundreds of group elements to represent a chain of length two.) Mercurial signatures allow for modular constructions of delegatable anonymous credentials that do not require NIZKs and are substantially more efficient: in the construction of [15], only five group elements are needed to represent each link in a credential chain.

A user may in fact be in possession of several types of credentials: a credential issued by her employer, one issued by the government, and another issued by a service provider, for example. *Multi-authority* delegatable anonymous credentials allow users to anonymously obtain, demonstrate possession of, and delegate credentials under different certification authorities, all with the same underlying identity. For example, suppose Alice has a level-1 credential from her employer and a level-2 credential from the government. Under the mer-

1 https://obamawhitehouse.archives.gov/sites/default/files/rss_viewer/NSTICstrategy_041511.pdf

curial signature scheme of [15], Alice could not possess a single underlying secret key. To see why, suppose a credential has just one single attribute. Following [15], to give Alice a level-1 credential, her employer signs a representative of the equivalence class of her public keys. If her secret key is a vector of length ℓ, then her public key is also a vector of length ℓ. To issue a level-2 credential, Alice signs a representative of the equivalence class of Bob's public keys; however, using a length-ℓ key, she may only sign length-ℓ-1 vectors, so Bob's secret key must be shorter than Alice's. By the same logic, any user who has a level-2 credential must have a secret key of shorter length than that of a user with a level-1 credential (under the same CA). In particular, if Alice's level-2 government credential chain is government $\rightarrow$ Carol $\rightarrow$ Alice for some user Carol with a secret key also of length ℓ, then Alice's secret key would need to be of length $\ell - 1$. This is a contradiction since Alice's secret key is of length ℓ.

Mercurial signatures for variable-length messages allow users to have the same underlying secret key under different CAs as well as any number of attributes (although note that delegators at the same level must have the same number of attributes or else their signatures are trivially distinguishable). This is a first step towards achieving efficient multi-authority delegatable anonymous credentials.

Mercurial signatures were inspired by Fuchsbauer, Hanser and Slamanig's work on structure-preserving signatures on equivalence classes (SPS-EQ) [22], which introduces the idea of transforming a signature σ on a fixed-length message m into a signature σ' on an equivalent but unlinkable message m'. Mercurial signatures [15] additionally allow the fixed-length public key pk to be transformed into an equivalent public key pk', where pk and pk' are unlinkable even when given signatures under both keys. A related concept, signatures with flexible public key [1], allows blinding of the public key, but not the message.

1.2 Our Contribution

The only previously known construction of mercurial signatures [15] was restricted to messages of fixed length, which limits its use in applications. Thus, our goal was to construct mercurial signatures that allow messages of any length to be signed under public keys of a small, fixed length. This is desirable because public keys are pseudonyms, which users may wish to be shorter than the messages they are signing.

While the prior construction of mercurial signatures [15] achieved the standard notion of unforgeability, namely existential unforgeability under chosen message attacks (EUF-CMA), the construction presented here is unforgeable in a more limited sense. Instead of the adversary having access to the usual signing oracle that simply responds with its signature σ on input a message m, the adversary obtains signatures via a *signing protocol* in which it is required to prove knowledge of the discrete logarithm of each message vector component. This proof of knowledge is needed for proving unforgeability. (The reduction must use these discrete logarithms.) This variant of unforgeability was defined by Fuchsbauer and Gay [20] as existential unforgeability under chosen *open* message attacks (EUF-CoMA).

The construction of mercurial signatures presented here also differs from that of [15] as far as message-blinding is concerned. Recall that Bob needs to blind a message m signed by a potentially malicious Alice by transforming it into a new message m' and adapting her signature σ into σ' accordingly. A property of mercurial signatures called *origin-hiding* guarantees that the resulting signature σ' is distributed identically to what Bob would have received had m' been signed anew. Our construction guarantees origin-hiding if the signer follows the signing algorithm, but a malicious signer could issue improperly formed signatures that would allow it to tell whether σ' was adapted from σ or was freshly issued. To mitigate this, the signer convinces the recipient that the signature was formed properly via an efficient zero-knowledge proof as part of the signing protocol.

Though our construction satisfies a weaker notion of unforgeability and origin-hiding, for the purpose of anonymous credentials, our results constitute a success. This is because the protocol for issuing anonymous credentials typically requires that the recipient prove knowledge of her secret key anyway, so relaxing unforgeability to EUF-CoMA comes for free. Relaxing origin-hiding so it holds only when signatures were issued properly adds an additional step to the signing protocol; however, it can be executed efficiently and is therefore also a reasonable relaxation.

Our construction of variable-length mercurial signatures uses the fixed-length mercurial signature scheme of [15] as a building block and is proven secure (under the variants of unforgeability and origin-hiding above) assuming (1) the security of the underlying mercurial signature scheme and (2) the ABDDH$^+$ assumption, which was introduced in [21] and is reminiscent of the decisional Diffie-Hellman assumption for Type III bilinear pairings.

Towards constructing variable-length mercurial signatures. A naive approach to extending mercurial signatures to allow messages of any length would be to hash the messages down to the correct fixed length and use the fixed-length mercurial signature scheme of [15]. In general, this does not work because we do not readily have a hash function H such that $H(m)$ and $H(m')$ are equivalent when m and m' are equivalent.

In order to maintain the equivalence relation among messages, we instead break a message $m = (\hat{g}, u_1, \ldots, u_n)$, where $\hat{g}$ is a base group element and $u_i = \hat{g}^{m_i}$ for some $m_i \in \mathbb{Z}_p^*$, into its n constituent group elements u_i. Each u_i, together with powers of a base $\tilde{g}$ indicating the index i, is signed using the fixed-length mercurial signature scheme. However, an adversary may be able to mix and match elements of the n new messages being signed under the fixed-length scheme. To mitigate this, an additional group element is included in each of the n messages to link them together and to the original message m in an unforgeable way. We call this additional element the "glue" element. Specifically, we represent the message m to be signed as a sequence of n messages $M_1 = (\tilde{g}, \tilde{g}^1, \tilde{g}^n, \tilde{g}^s, u_1)$, $M_2 = (\tilde{g}, \tilde{g}^2, \tilde{g}^n, \tilde{g}^s, u_2)$,…, $M_n = (\tilde{g}, \tilde{g}^n, \tilde{g}^n, \tilde{g}^s, u_n)$, where $\tilde{g}^s$ is the glue element. This allows the message m to be transformed into an equivalent message $m' = m^\mu = (\hat{g}^\mu, u_1^\mu, \ldots, u_n^\mu)$, for any $\mu \in \mathbb{Z}_p^*$, by simply changing each M_i to $M_i' = M_i^\mu = (\tilde{g}^\mu, (\tilde{g}^\mu)^i, (\tilde{g}^\mu)^n, (\tilde{g}^\mu)^s, u_i^\mu)$ and invoking the underlying algorithm of the fixed-length scheme that updates the signature. The problem with this approach, however, is that different signatures receive different glue values, so origin-hiding does not hold in a statistical sense. In order to satisfy the origin-hiding property, the glue element $\tilde{g}^s$ must be computed (relative to $\tilde{g}$) as a function of the entire equivalence class to which the message belongs. That way, no matter which message in the class is signed, the glue element's discrete logarithm base $\tilde{g}$ is the same. Our main technical insight is how to compute the glue element such that it is a function of the entire equivalence class that a message represents, and not just the message itself.

2 Preliminaries

A function $\nu : \mathbb{N} \to \mathbb{R}$ is called *negligible* if for all $c > 0$, there exists a k_0 such that $\nu(k) < \frac{1}{k^c}$ for all $k > k_0$. Let $y \leftarrow A(x)$ denote running a probabilistic algorithm A on input x and assigning the output to y.

Definition 1 (Bilinear pairing)**.** Let $\mathbb{G}_1, \mathbb{G}_2$, and $\mathbb{G}_T$ be multiplicative groups of prime order p, and let P and $\hat{P}$ be generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. A *bilinear pairing* is a map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ that satisfies (1) bilinearity: $e(P^a, \hat{P}^b) = e(P, \hat{P})^{ab} = e(P^b, \hat{P}^a)\ \forall\ a, b \in \mathbb{Z}_p$; (2) non-degeneracy: $e(P, \hat{P}) \neq 1_{\mathbb{G}_T}$ (i.e., $e(P, \hat{P})$ generates $\mathbb{G}_T$); and (3) computability: there exists an efficient algorithm to compute e.

Bilinear pairings can be classified into three types. We consider Type III (asymmetric) pairings, where $\mathbb{G}_1 \neq \mathbb{G}_2$ and there is no efficiently computable homomorphism between them.

Definition 2 (Bilinear group generator)**.** A *bilinear group generator* BGGen is a (possibly probabilistic) polynomial-time algorithm that takes as input a security parameter 1^k and outputs a bilinear group description $\mathsf{BG} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, P, \hat{P}, e)$ with a Type III pairing.

Definition 3 (Discrete logarithm assumption (DL))**.** Let BGGen be a bilinear group generator that outputs $\mathsf{BG} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, P_1, P_2, e)$. For $i \in \{1, 2\}$, the *discrete logarithm assumption* holds in $\mathbb{G}_i$ for BGGen if for all probabilistic, polynomial-time (PPT) adversaries $\mathcal{A}$, there exists a negligible function ν such that: $\Pr[\mathsf{BG} \leftarrow \mathsf{BGGen}(1^k), x \leftarrow \mathbb{Z}_p, x' \leftarrow \mathcal{A}(\mathsf{BG}, P_i^x) : P_i^{x'} = P_i^x] \leq \nu(k)$.

Definition 4. (Decisional Diffie-Hellman assumption (DDH)). Let BGGen be a bilinear group generator that outputs $\mathsf{BG} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, P_1, P_2, e)$. For $i \in \{1, 2\}$, the *decisional Diffie-Hellman assumption* holds in $\mathbb{G}_i$ for BGGen if for all probabilistic, polynomial-time (PPT) adversaries $\mathcal{A}$, there exists a negligible function ν such that: $\Pr[b \leftarrow \{0, 1\}, \mathsf{BG} \leftarrow \mathsf{BGGen}(1^k), s, t, r \leftarrow \mathbb{Z}_p, b^* \leftarrow \mathcal{A}(\mathsf{BG}, P_i^s, P_i^t, P_i^{(1-b)\cdot r + b \cdot st}) : b^* = b] - \frac{1}{2} \leq \nu(k)$.

3 Definition

We begin with the definition of mercurial signatures. The following definition is mostly a restatement of [15] with a few adaptations to accommodate messages of any length. We denote by $\mathcal{M}_n$ the message space consisting of all message vectors of length n. The key generation algorithm KeyGen no longer takes as input a fixed length parameter, and the signature conversion algorithm $\mathsf{ConvertSig}$ now takes as input a message converter μ to transform (m, σ) into $(m', \tilde{\sigma})$. The original construction of mercurial signatures [15] satisfies this revised definition for a fixed-length message space.

Definition 5 (Mercurial signature)**.** A *mercurial signature scheme* for parameterized 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 stated otherwise:

$\mathsf{PPGen}(1^k) \to PP$: On input the security parameter 1^k, this probabilistic algorithm outputs the public parameters PP. This includes parameters for the parameterized equivalence relations $\mathcal{R}_m$, $\mathcal{R}_{\mathsf{pk}}$, $\mathcal{R}_{\mathsf{sk}}$ so they are all well defined. It also includes parameters for the algorithms sample_ρ and sample_μ, which sample key and message converters, respectively.

$\mathsf{KeyGen}(PP) \to (\mathsf{pk}, \mathsf{sk})$: On input the public parameters PP, this probabilistic algorithm outputs a key pair $(\mathsf{pk}, \mathsf{sk})$. This algorithm also defines a correspondence between public and secret keys: we write $(\mathsf{pk}, \mathsf{sk}) \in \mathsf{KeyGen}(PP)$ if there exists a set of random choices that KeyGen could make that would result in $(\mathsf{pk}, \mathsf{sk})$ as the output.

$\mathsf{Sign}(\mathsf{sk}, m) \to \sigma$: On input the signing key sk and a message $m \in \mathcal{M}$, this probabilistic algorithm outputs a signature σ.

$\mathsf{Verify}(\mathsf{pk}, m, \sigma) \to 0/1$: On input the public key pk, a message m, and a purported signature σ, output 0 or 1.

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

$\mathsf{ConvertPK}(\mathsf{pk}, \rho) \to \tilde{\mathsf{pk}}$: On input pk and a key converter $\rho \in \mathsf{sample}_\rho$, output a new public key $\tilde{\mathsf{pk}} \in [\mathsf{pk}]_{\mathcal{R}_{\mathsf{pk}}}$. (Correctness of this operation, defined below, will guarantee that if pk corresponds to sk, then $\tilde{\mathsf{pk}}$ corresponds to $\tilde{\mathsf{sk}} = \mathsf{ConvertSK}(\mathsf{sk}, \rho)$.)

$\mathsf{ChangeRep}(\mathsf{pk}, m, \sigma, \mu) \to (m', \sigma')$: On input pk, a message m, a signature σ, and a message converter $\mu \in \mathsf{sample}_\mu$, this probabilistic algorithm computes a new message representative $m' \in [m]_{\mathcal{R}_m}$ and a new signature σ' and outputs (m', σ'). (Correctness of this will require that whenever $\mathsf{Verify}(\mathsf{pk}, m, \sigma) = 1$, it will also be the case that $\mathsf{Verify}(\mathsf{pk}, m', \sigma') = 1$.)

$\mathsf{ConvertSig}(\mathsf{pk}, m, \sigma, \rho, \mu) \to (m', \tilde{\sigma})$: On input pk, a message m, a signature σ, a key converter $\rho \in \mathsf{sample}_\rho$, and a message converter $\mu \in \mathsf{sample}_\mu$, this probabilistic algorithm computes a new message representative $m' \in [m]_{\mathcal{R}_m}$ and a new signature $\tilde{\sigma}$ and outputs $(m', \tilde{\sigma})$. (Correctness of this will require that whenever $\mathsf{Verify}(\mathsf{pk}, m, \sigma) = 1$, it will also be the case that $\mathsf{Verify}(\tilde{\mathsf{pk}}, m', \tilde{\sigma}) = 1$, where $\tilde{\mathsf{pk}} = \mathsf{ConvertPK}(\mathsf{pk}, \rho)$.)

Definition 6 (Correctness)**.** A mercurial signature scheme $(\mathsf{PPGen}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{ConvertSK}, \mathsf{ConvertPK}, \mathsf{ChangeRep}, \mathsf{ConvertSig})$ 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 k, for all $PP \in \mathsf{PPGen}(1^k)$, and for all $(\mathsf{pk}, \mathsf{sk}) \in \mathsf{KeyGen}(PP)$:

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

Key conversion: For all $\rho \in \mathsf{sample}_\rho$, $(\mathsf{ConvertPK}(\mathsf{pk}, \rho), \mathsf{ConvertSK}(\mathsf{sk}, \rho)) \in \mathsf{KeyGen}(PP)$. Moreover, $\mathsf{ConvertSK}(\mathsf{sk}, \rho) \in [\mathsf{sk}]_{\mathcal{R}_{\mathsf{sk}}}$ and $\mathsf{ConvertPK}(\mathsf{pk}, \rho) \in [\mathsf{pk}]_{\mathcal{R}_{\mathsf{pk}}}$.

Change of message representative: For all $m \in \mathcal{M}$, for all σ such that $\mathsf{Verify}(\mathsf{pk}, m, \sigma) = 1$, for all $\mu \in \mathsf{sample}_\mu$, for all $(m', \sigma') \in \mathsf{ChangeRep}(\mathsf{pk}, m, \sigma, \mu)$, $\mathsf{Verify}(\mathsf{pk}, m', \sigma') = 1$, where $m' \in [m]_{\mathcal{R}_m}$.

Signature conversion: For all $m \in \mathcal{M}$, for all σ such that $\mathsf{Verify}(\mathsf{pk}, m, \sigma) = 1$, for all $\rho \in \mathsf{sample}_\rho$, for all $\mu \in \mathsf{sample}_\mu$, for all $(m', \tilde{\sigma}) \in \mathsf{ConvertSig}(\mathsf{pk}, m, \sigma, \rho, \mu)$, $\mathsf{Verify}(\mathsf{ConvertPK}(\mathsf{pk}, \rho), m', \tilde{\sigma}) = 1$, where $m' \in [m]_{\mathcal{R}_m}$.

Correct verification, key conversion, and change of message representative are exactly as in [15]. Correct signature conversion means that if a key converter ρ is applied to a public key pk to obtain an equivalent $\tilde{\mathsf{pk}}$, and the same ρ together with a message converter μ is applied to a valid message-signature pair (m, σ) to obtain $(m', \tilde{\sigma})$, then the signature $\tilde{\sigma}$ is valid on the equivalent message m' under the public key $\tilde{\mathsf{pk}}$.

Definition 7 (Unforgeability)**.** A mercurial signature scheme $(\mathsf{PPGen}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{ConvertSK}, \mathsf{ConvertPK}, \mathsf{ChangeRep}, \mathsf{ConvertSig})$ for parameterized equivalence relations $\mathcal{R}_m, \mathcal{R}_{\mathsf{pk}}, \mathcal{R}_{\mathsf{sk}}$ is *unforgeable* if for all probabilistic, polynomial-time (PPT) algorithms $\mathcal{A}$ having access to a signing oracle, there exists a negligible function ν such that:

$$\Pr[PP \leftarrow \mathsf{PPGen}(1^k); (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(PP); (Q, \mathsf{pk}^*, m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sign}(\mathsf{sk}, \cdot)}(\mathsf{pk}) : \forall\, \bar{m} \in Q, [m^*]_{\mathcal{R}_m} \neq [m]_{\mathcal{R}_m} \wedge [\mathsf{pk}^*]_{\mathcal{R}_{\mathsf{pk}}} = [\mathsf{pk}]_{\mathcal{R}_{\mathsf{pk}}} \wedge \mathsf{Verify}(\mathsf{pk}^*, m^*, \sigma^*) = 1] \leq \nu(k)$$

where Q is the set of discrete logarithms $\bar{m}$ of messages m that $\mathcal{A}$ has queried to the signing oracle.

This definition is similar to existential unforgeability under chosen open message attacks (EUF-CoMA) defined by Fuchsbauer and Gay [20]. EUF-CoMA differs from

EUF-CMA in that the adversary must provide the discrete logarithm $\bar{m}$ of the message m to be signed. This has the advantage that the adversary's success is efficiently verifiable [20]. Our notion of unforgeability is similar to EUF-CoMA, except the adversary's winning condition is slightly altered. As in the EUF-CoMA game, the adversary is given the public key pk and is allowed to query the signing oracle that knows the corresponding secret key sk. Eventually, the adversary outputs a public key pk^*, a message m^*, and a purported signature σ^*. Unlike the EUF-CoMA game, the adversary has the freedom to output a forgery under a different public key pk^*, as long as pk^* is in the same equivalence class as pk. This seemingly makes the adversary's task easier. At the same time, the adversary's forgery is not valid if the message m^* is in the same equivalence class as a previously queried message m, making the adversary's task harder. The definition of unforgeability for mercurial signatures in [15] allows a forgery under an equivalent public key, but does not require the adversary to provide the discrete logarithm of the message to be signed by the oracle.

Remark. In Section 4.1, we define an interactive signing protocol in which the recipient of the signature gives a zero-knowledge proof of knowledge (ZKPoK) of the discrete logarithm of the message.

Definition 8 (Class- and origin-hiding). A mercurial signature scheme (PPGen, $\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{ConvertSK}, \mathsf{ConvertPK}, \mathsf{ChangeRep}, \mathsf{ConvertSig}$) for parameterized equivalence relations $\mathcal{R}_m, \mathcal{R}_{\mathsf{pk}}, \mathcal{R}_{\mathsf{sk}}$ is *class-hiding* if it satisfies the following two properties:

Message class-hiding: For all polynomial-length parameters $n(k)$, and for all probabilistic, polynomial-time (PPT) algorithms $\mathcal{A}$, there exists a negligible function ν such that:

$$\begin{aligned}&\Pr[PP \leftarrow \mathsf{PPGen}(1^k);\ m_1 \leftarrow \mathcal{M}_{n(k)};\ m_2^0 \leftarrow \mathcal{M}_{n(k)};\\ &\quad m_2^1 \leftarrow [m_1]_{\mathcal{R}_m};\ b \leftarrow \{0,1\};\\ &\quad b' \leftarrow \mathcal{A}(PP, m_1, m_2^b)\ :\ b' = b] \le \tfrac{1}{2} + \nu(k)\end{aligned}$$

Public key class-hiding: For all probabilistic, polynomial-time (PPT) algorithms $\mathcal{A}$, there exists a negligible function ν such that:

$$\begin{aligned}&\Pr[PP \leftarrow \mathsf{PPGen}(1^k);\ (\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathsf{KeyGen}(PP);\\ &\quad (\mathsf{pk}_2^0, \mathsf{sk}_2^0) \leftarrow \mathsf{KeyGen}(PP);\ \rho \leftarrow \mathsf{sample}_\rho(PP);\\ &\quad \mathsf{pk}_2^1 = \mathsf{ConvertPK}(\mathsf{pk}_1, \rho);\ \mathsf{sk}_2^1 = \mathsf{ConvertSK}(\mathsf{sk}_1, \rho);\\ &\quad b \leftarrow \{0,1\};\ b' \leftarrow \mathcal{A}^{\mathsf{Sign}(\mathsf{sk}_1,\cdot),\mathsf{Sign}(\mathsf{sk}_2^b,\cdot)}(\mathsf{pk}_1, \mathsf{pk}_2^b)\\ &\quad :\ b' = b] \le \tfrac{1}{2} + \nu(k)\end{aligned}$$

A mercurial signature is also *origin-hiding* if the following two properties hold:

Origin-hiding of $\mathsf{ChangeRep}$: For all k, for all $PP \in \mathsf{PPGen}(1^k)$, for all pk^* (in particular, adversarially generated ones), for all m, σ, if $\mathsf{Verify}(\mathsf{pk}^*, m, \sigma) = 1$, if $\mu \leftarrow \mathsf{sample}_\mu$, then with overwhelming probability $\mathsf{ChangeRep}(\mathsf{pk}^*, m, \sigma, \mu)$ outputs a uniformly random $m' \in [m]_{\mathcal{R}_m}$ and a uniformly random $\sigma' \in \{\hat{\sigma} \mid \mathsf{Verify}(\mathsf{pk}^*, m', \hat{\sigma}) = 1\}$.

Origin-hiding of $\mathsf{ConvertSig}$: For all k, for all $PP \in \mathsf{PPGen}(1^k)$, for all pk^* (in particular, adversarially generated ones), for all m, σ, if $\mathsf{Verify}(\mathsf{pk}^*, m, \sigma) = 1$, if $\rho \leftarrow \mathsf{sample}_\rho$ and $\mu \leftarrow \mathsf{sample}_\mu$, then with overwhelming probability $\mathsf{ConvertSig}(\mathsf{pk}^*, m, \sigma, \rho, \mu)$ outputs a uniformly random $m' \in [m]_{\mathcal{R}_m}$ and a uniformly random $\tilde{\sigma} \in \{\hat{\sigma} \mid \mathsf{Verify}(\mathsf{ConvertPK}(\mathsf{pk}^*, \rho), m', \hat{\sigma}) = 1)\}$. $\mathsf{ConvertPK}(\mathsf{pk}^*, \rho)$ outputs a uniformly random element of $[\mathsf{pk}^*]_{\mathcal{R}_{\mathsf{pk}}}$.

Remark. This definition of origin-hiding is a relaxation of the prior definition [15] in that there is a small probability that the outputs of $\mathsf{ChangeRep}$ and $\mathsf{ConvertSig}$ are not distributed correctly. It will become clear why in Section 4.1.

4 Construction

Let $\mathbb{G}_1, \mathbb{G}_2$, and $\mathbb{G}_T$ be multiplicative groups of prime order p with a Type III bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. Similar to the prior mercurial signature scheme [15], the message space for our new mercurial signature scheme consists of vectors of group elements from $\mathbb{G}_1^*$, where $\mathbb{G}_1^* = \mathbb{G}_1 \backslash \{1_{\mathbb{G}_1}\}$. Unlike the prior scheme, these can be vectors of any length. The message space is $\mathcal{M}_n = \{(\hat{g}, u_1, \ldots, u_n) \in (\mathbb{G}_1^*)^{n+1}\}$, where $\hat{g}$ is a generator of $\mathbb{G}_1$, and for all $1 \le i \le n$, $u_i = \hat{g}^{m_i}$ for some $m_i \in \mathbb{Z}_p^*$. The space of secret keys consists of vectors of elements from $\mathbb{Z}_p^*$. The space of public keys, similar to the message space, consists of vectors of group elements from $\mathbb{G}_2^*$. A scheme with messages over $\mathbb{G}_2^*$ and public keys over $\mathbb{G}_1^*$ can be obtained by simply switching $\mathbb{G}_1^*$ and $\mathbb{G}_2^*$ throughout. Once the prime p, $\mathbb{G}_1$, and $\mathbb{G}_2$ are well defined, for a length parameter $n \in \mathbb{N}$ the equivalence relations are as follows:

$$\mathcal{R}_m = \{(m, m') \in (\mathbb{G}_1^*)^{n+1} \times (\mathbb{G}_1^*)^{n+1} \mid \exists\, \mu \in \mathbb{Z}_p^* \text{ s.t. } m' = m^\mu\}$$
$$\mathcal{R}_{\mathsf{sk}} = \{(\mathsf{sk}_\mathsf{X}, \tilde{\mathsf{sk}}_\mathsf{X}) \in (\mathbb{Z}_p^*)^{10} \times (\mathbb{Z}_p^*)^{10} \mid \exists\, \rho \in \mathbb{Z}_p^* \text{ s.t. } \tilde{\mathsf{sk}} = \rho \cdot \mathsf{sk}\}$$
$$\mathcal{R}_{\mathsf{pk}} = \{(\mathsf{pk}_\mathsf{X}, \tilde{\mathsf{pk}}_\mathsf{X}) \in (\mathbb{G}_2^*)^{10} \times (\mathbb{G}_2^*)^{10} \mid \exists\, \rho \in \mathbb{Z}_p^* \text{ s.t. } \tilde{\mathsf{pk}} = \mathsf{pk}^\rho\}$$

Our variable-length mercurial signature scheme, denoted MS_X, is an extension of the prior fixed-length scheme, denoted MS_f [15], which can be found in Appendix A. The subscript X, for extension, is used to denote all keys and algorithms associated with the variable-length scheme $\mathsf{MS_X}$.

Let us discuss the security properties of the fixed-length scheme MS_f. It satisfies the definition of security in Section 3, but only for the fixed-length message space $\mathcal{M}_5 = (\mathbb{G}_1^*)^5$. If given as input a message $m \notin \mathcal{M}_5$, the signing algorithm rejects. Correspondingly, correctness only holds for messages of the correct length. MS_f satisfies the definition of unforgeability in Section 3 as well as message and public key class-hiding. As for origin-hiding, $\mathsf{ChangeRep}_f(\mathsf{pk}, m, \sigma, \mu)$ outputs (m', σ'), where $m' = m^\mu \in [m]_{\mathcal{R}_m}$ for a message converter $\mu \in \mathbb{Z}_p^*$ and σ' is a valid signature on m' under pk, and $\mathsf{ConvertSig}_f(\mathsf{pk}, m, \sigma, \rho)$ outputs $\tilde{\sigma}$, where $\tilde{\sigma}$ is a valid signature on m under $\tilde{\mathsf{pk}} = \mathsf{pk}^\rho \in [\mathsf{pk}]_{\mathcal{R}_{\mathsf{pk}}}$ for a key converter $\rho \in \mathbb{Z}_p^*$. Both $\mathsf{ChangeRep}_f$ and $\mathsf{ConvertSig}_f$ satisfy origin-hiding with probability 1. The following theorem summarizes the security properties of MS_f.

Theorem 1. [15]. *The mercurial signature scheme MS_f is correct for fixed-length messages, unforgeable, and satisfies class- and origin-hiding in the generic group model for Type III bilinear groups.*

$\mathsf{MS_X}$ can be constructed from MS_f on messages of length $\ell = 5$ as follows. A message m is written as $m = (\hat{g}, u_1, \ldots, u_n) \in (\mathbb{G}_1^*)^{n+1}$, where $\hat{g}$ is a generator of $\mathbb{G}_1$ and for all $1 \le i \le n$, $u_i = \hat{g}^{m_i}$ for some $m_i \in \mathbb{Z}_p^*$. For a generator $\tilde{g}$ of $\mathbb{G}_1$ and "glue" element $\tilde{h} \in \mathbb{G}_1^*$ (discussed shortly), the message m can be represented as a set of n messages that are in the message space of the mercurial signature scheme MS_f as follows, where $\tilde{u}_i = \tilde{g}^{m_i}$ for all $1 \le i \le n$:

$$\begin{aligned} M_1 &= (\tilde{g}, \tilde{g}^1, \tilde{g}^n, \tilde{h}, \tilde{u}_1) \\ M_2 &= (\tilde{g}, \tilde{g}^2, \tilde{g}^n, \tilde{h}, \tilde{u}_2) \\ &\vdots \\ M_n &= (\tilde{g}, \tilde{g}^n, \tilde{g}^n, \tilde{h}, \tilde{u}_n) \end{aligned}$$

Each message $M_i = (\tilde{g}, \tilde{g}^i, \tilde{g}^n, \tilde{h}, \tilde{u}_i)$ is signed using the mercurial signature scheme MS_f, resulting in a signature σ_i. The verification consists of checking the n message-signature pairs (M_i, σ_i) using the prior mercurial signature Verify_f algorithm.

How might we form the glue element $\tilde{h}$? As discussed in the introduction, it is important for the origin-hiding property that $\tilde{h}$ for a message $m = (\hat{g}, u_1, \ldots, u_n)$, where $u_i = \hat{g}^{m_i}$, be a function of the m_i's so that if another representative $m' \in [m]_{\mathcal{R}_m}$ gets signed, the corresponding M_i''s are in the same equivalence classes as the original M_i's for the original m (i.e., $M_i' \in [M_i]_{\mathcal{R}_m}$ for all $1 \le i \le n$). Computing $\tilde{h}$ as $\tilde{g}^{R(m_1,\ldots,m_n)}$ for a random function R of the m_i's would work, but how would the signer compute such a value? A pseudorandom function could be used instead, but it is not obvious how to compute it since the signer has the group elements $u_1, \ldots, u_n$, but not their discrete logarithms $m_1, \ldots, m_n$.

Our solution is as follows. Consider a polynomial $p_m(x)$ parameterized by the m_i's: $p_m(x) = m_1 + m_2 x + m_3 x^2 \cdots + m_n x^{n-1}$. The signer evaluates this polynomial at a secret value $\hat{x}$ known only to him: $p_m(\hat{x})$. The glue element could be computed by the signer as $\hat{h} = \hat{g}^{p_m(\hat{x})}$; however, to ensure that it is pseudorandom, the signer picks a uniformly random $w \leftarrow \mathbb{Z}_p^*$, sets $\tilde{g} = g^w$, and computes the glue element as $\tilde{h} = \tilde{g}^{p_m(\hat{x})}$. Additionally, the signer picks a uniformly random $y \leftarrow \mathbb{Z}_p^*$ and raises $\tilde{g}^{p_m(\hat{x})}$ to y, resulting in the following:

$$\tilde{h} = \left(\tilde{g}^{p_m(\hat{x})}\right)^y = \left(\tilde{g}^{\sum_{i=1}^n m_i \hat{x}^{i-1}}\right)^y = \left(\prod_{i=1}^n \tilde{g}^{m_i \hat{x}^{i-1}}\right)^y \quad (1)$$

Note that w is fresh for each signature, but y is the same for all signatures issued by the same signer. In reality, the signer does not know the m_i's required to form the polynomial $p_m(\hat{x})$; however, he is given as input the original u_i's, which have the relationship $u_i = \hat{g}^{m_i}$, so $\tilde{h}$ can be computed directly as follows, where $\tilde{u}_i = u_i^w = \tilde{g}^{m_i}$: $\tilde{h} = \left(\prod_{i=1}^n \tilde{u}_i^{\hat{x}^{i-1}}\right)^y$. This is exactly Equation (1).

We now describe our construction formally. We first provide a non-interactive construction that satisfies the input-output specification in the definition of mercurial signatures. The final construction (Section 4.1) involves an interactive signing protocol carried out between the signer and the recipient of the signature.

Construction. The following algorithms are invoked from the fixed-length mercurial signature scheme MS_f : $\mathsf{ChangeRep}_f(\mathsf{pk}, m, \sigma, \mu) \rightarrow (m', \sigma')$, where $m' = m^\mu \in [m]_{\mathcal{R}_m}$ for a message converter $\mu \in \mathbb{Z}_p^*$ and $\mathsf{Verify}_f(\mathsf{pk}, m', \sigma') = 1$, and $\mathsf{ConvertSig}_f(\mathsf{pk}, m, \sigma, \rho) \rightarrow \tilde{\sigma}$, where $\mathsf{Verify}_f(\tilde{\mathsf{pk}}, m, \tilde{\sigma}) = 1$ and $\tilde{\mathsf{pk}} = \mathsf{pk}^\rho \in [\mathsf{pk}]_{\mathcal{R}_{\mathsf{pk}}}$ for a key converter $\rho \in \mathbb{Z}_p^*$.

$\mathsf{PPGen_X}(1^k) \rightarrow PP_\mathsf{X}$: Run $PP \leftarrow \mathsf{PPGen}_f(1^k)$ and output $PP_\mathsf{X} = PP = \mathsf{BG} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, P, \hat{P}, e)$.

$\mathsf{KeyGen_X}(PP_\mathsf{X}) \rightarrow (\mathsf{pk_X}, \mathsf{sk_X})$: Run $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}_f(PP, \ell = 5)$, where $\mathsf{sk} = (x_1, x_2, x_3, x_4, x_5) \in (\mathbb{Z}_p^*)^5$ and $\mathsf{pk} = (\hat{X}_1, \hat{X}_2, \hat{X}_3, \hat{X}_4, \hat{X}_5) \in (\mathbb{G}_2^*)^5$ for $\hat{X}_i = \hat{P}^{x_i}$. Pick uniformly at random a secret point

$\hat{x} \leftarrow \mathbb{Z}_p^*$ and secret seeds $y_1, y_2 \leftarrow \mathbb{Z}_p^*$. Also pick $x_6, x_8 \leftarrow \mathbb{Z}_p^*$ and set $x_7 = x_6 \cdot \hat{x}$ and $x_9 = x_8 \cdot y_1$ and $x_{10} = x_8 \cdot y_2$. Set $\mathsf{sk_X} = (\mathsf{sk}, x_6, x_7, x_8, x_9, x_{10})$ and $\mathsf{pk_X} = (\mathsf{pk}, \hat{X}_6, \hat{X}_7, \hat{X}_8, \hat{X}_9, \hat{X}_{10})$, where $\hat{X}_i = \hat{P}^{x_i}$, and output $(\mathsf{pk_X}, \mathsf{sk_X})$.

$\mathsf{Sign_X}(\mathsf{sk_X}, m) \to (\hat{h}, \sigma)$: On input $\mathsf{sk_X} = (\mathsf{sk}, x_6, x_7, x_8, x_9, x_{10})$ and a message $m = (\hat{g}, u_1, \ldots, u_n) \in (\mathbb{G}_1^*)^{n+1}$, where $\hat{g}$ is a generator of $\mathbb{G}_1$, compute $\hat{x} = x_7 \cdot x_6^{-1}$ and $y_1 = x_9 \cdot x_8^{-1}$ and $y_2 = x_{10} \cdot x_8^{-1}$. Then, compute $y := y_1 \cdot y_2$ and $\hat{h} = \left(\prod_{i=1}^{n} u_i^{\hat{x}^{i-1}} \right)^y$. Compute $\hat{g}^2, \ldots, \hat{g}^n$. For all $1 \le i \le n$, form the message $M_i = (\hat{g}, \hat{g}^i, \hat{g}^n, \hat{h}, u_i)$ and run $\sigma_i \leftarrow \mathsf{Sign}_f(\mathsf{sk}, M_i)$. Output the signature $(\hat{h}, \sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_n\})$.

$\mathsf{Verify_X}(\mathsf{pk_X}, m, (\hat{h}, \sigma)) \to 0/1$: On input $\mathsf{pk_X} = (\mathsf{pk}, \hat{X}_6, \hat{X}_7, \hat{X}_8, \hat{X}_9, \hat{X}_{10})$, $m = (\hat{g}, u_1, \ldots, u_n)$, and a signature $(\hat{h}, \sigma = \{\sigma_1, \ldots, \sigma_n\})$, compute $\hat{g}^2, \ldots, \hat{g}^n$. For all $1 \le i \le n$, form the message $M_i = (\hat{g}, \hat{g}^i, \hat{g}^n, \hat{h}, u_i)$ and check whether $\mathsf{Verify}_f(\mathsf{pk}, M_i, \sigma_i) = 1$. If these checks hold, output 1; otherwise output 0.

$\mathsf{ConvertSK_X}(\mathsf{sk_X}, \rho) \to \tilde{\mathsf{sk}}_\mathsf{X}$: On input $\mathsf{sk_X} = (\mathsf{sk}, x_6, x_7, x_8, x_9, x_{10})$ and $\rho \in \mathbb{Z}_p^*$, run $\tilde{\mathsf{sk}} \leftarrow \mathsf{ConvertSK}_f(\mathsf{sk}, \rho)$, where $\tilde{\mathsf{sk}} = \rho \cdot \mathsf{sk}$, compute $\tilde{x}_i = \rho \cdot x_i$ for all $6 \le i \le 10$, and output the new secret key $\tilde{\mathsf{sk}}_\mathsf{X} = (\tilde{\mathsf{sk}}, \tilde{x}_6, \tilde{x}_7, \tilde{x}_8, \tilde{x}_9, \tilde{x}_{10})$.

$\mathsf{ConvertPK_X}(\mathsf{pk_X}, \rho) \to \tilde{\mathsf{pk}}_\mathsf{X}$: On input $\mathsf{pk_X} = (\mathsf{pk}, \hat{X}_6, \hat{X}_7, \hat{X}_8, \hat{X}_9, \hat{X}_{10})$ and $\rho \in \mathbb{Z}_p^*$, run $\tilde{\mathsf{pk}} \leftarrow \mathsf{ConvertPK}_f(\mathsf{pk}, \rho)$, where $\tilde{\mathsf{pk}} = \mathsf{pk}^\rho$, compute $\tilde{X}_i = \hat{X}_i^\rho$ for all $6 \le i \le 10$, and output the new public key $\tilde{\mathsf{pk}}_\mathsf{X} = (\tilde{\mathsf{pk}}, \tilde{X}_6, \tilde{X}_7, \tilde{X}_8, \tilde{X}_9, \tilde{X}_{10})$.

$\mathsf{ChangeRep_X}(\mathsf{pk_X}, m, (\hat{h}, \sigma), \mu) \to (m', (\hat{h}', \sigma'))$: On input $\mathsf{pk_X} = (\mathsf{pk}, \hat{X}_6, \hat{X}_7, \hat{X}_8, \hat{X}_9, \hat{X}_{10})$, $m = (\hat{g}, u_1, \ldots, u_n)$, $(\hat{h}, \sigma = \{\sigma_1, \ldots, \sigma_n\})$, and $\mu \in \mathbb{Z}_p^*$, compute $\hat{g}^2, \ldots, \hat{g}^n$. For all $1 \le i \le n$, form the message $M_i = (\hat{g}, \hat{g}^i, \hat{g}^n, \hat{h}, u_i)$ and run $(M_i', \sigma_i') \leftarrow \mathsf{ChangeRep}_f(\mathsf{pk}, M_i, \sigma_i, \mu)$, where $M_i' = (\hat{g}^\mu, (\hat{g}^\mu)^i, (\hat{g}^\mu)^n, \hat{h}^\mu, u_i^\mu)$. Set $m' = (\hat{g}', u_1', \ldots, u_n') = (\hat{g}^\mu, u_1^\mu, \ldots, u_n^\mu)$ and $\hat{h}' = \hat{h}^\mu$ and output $(m', (\hat{h}', \sigma' = \{\sigma_1', \ldots, \sigma_n'\}))$.

$\mathsf{ConvertSig_X}(\mathsf{pk_X}, m, (\hat{h}, \sigma), \rho, \mu) \to (m', (\hat{h}', \tilde{\sigma}))$: On input $\mathsf{pk_X}$, m, $(\hat{h}, \sigma)$, and $\rho, \mu \in \mathbb{Z}_p^*$, run $(m', (\hat{h}', \sigma')) \leftarrow \mathsf{ChangeRep_X}(\mathsf{pk_X}, m, (\hat{h}, \sigma), \mu)$, where $m' = (\hat{g}', u_1', \ldots, u_n')$ and $\sigma' = \{\sigma_1', \ldots, \sigma_n'\}$. Compute $(\hat{g}')^2, \ldots, (\hat{g}')^n$. For all $1 \le i \le n$, form the message $M_i' = (\hat{g}', (\hat{g}')^i, (\hat{g}')^n, \hat{h}', u_i')$ and run $\tilde{\sigma}_i \leftarrow \mathsf{ConvertSig}_f(\mathsf{pk}, M_i', \sigma_i', \rho)$. Output $(m', (\hat{h}', \tilde{\sigma} = \{\tilde{\sigma}_1, \ldots, \tilde{\sigma}_n\}))$.

4.1 Signing Protocol

Our construction satisfies the input-output specification in the definition of mercurial signatures; however, unfortunately, our proofs of unforgeability and origin-hiding do not allow a signer to simply sign any message given to it as input. Instead, the signer must run a signing protocol with the receiver of the signature. When a signature is queried on a message $m = (\hat{g}, u_1, \ldots, u_n) \in (\mathbb{G}_1^*)^{n+1}$, the signer first has the recipient give a ZKPoK that, for all $1 \le i \le n$, the recipient knows m_i such that $u_i = \hat{g}^{m_i}$. This ZKPoK is requisite for proving unforgeability, as the reduction's algorithm must use the exponent m_i's. The signer then carries out the signing algorithm $\mathsf{Sign_X}$ as specified in the construction above, with one modification: the signer picks a uniformly random $w \leftarrow \mathbb{Z}_p^*$, sets $\tilde{g} = \hat{g}^w$, and computes the glue element $\tilde{h}$ relative to base $\tilde{g}$. The additional randomness w ensures that the glue element is pseudorandom, as discussed in Section 4.

In addition to the usual unforgeability property that protects the signer, mercurial signatures also have the origin-hiding property that protects the privacy of the signature recipient. Intuitively, origin-hiding means that a message-signature pair (m, σ) is distributed exactly the same way whether (1) the signature σ on m was issued directly by the signer, or (2) (m, σ) was obtained by running $\mathsf{ChangeRep}(\mathsf{pk}, m', \sigma')$ on an equivalent m'. The reason it protects the signature recipient is that the resulting (m, σ) is not linkable to the specific point in time when this recipient was issued this signature.

In order to satisfy the origin-hiding property, the glue element $\tilde{h}$ must be computed (relative to $\tilde{g}$) as a function of the entire equivalence class to which the message belongs. That way, no matter which message in the class is signed, the glue element's discrete logarithm base $\tilde{g}$ is the same. A dishonest signer might try to compute the glue element incorrectly, depriving the recipient of the benefits that origin-hiding confers. Thus, as a final step in the signing protocol, the recipient verifies that the glue element was indeed computed correctly via a ZKPoK, so origin-hiding holds for all signers, not just honest ones.

Signing Protocol: This is an interactive protocol between a Signer, who runs the Sign side of the protocol, and a Receiver, who runs the $\mathsf{Receive}$ side.

$[\mathsf{Sign_X}(\mathsf{sk_X}, m) \leftrightarrow \mathsf{Receive_X}(\mathsf{pk_X}, m, (m_1, \ldots, m_n))] \to (\tilde{m}, (\tilde{h}, \sigma))$: The Signer takes as input his signing key $\mathsf{sk_X} = (\mathsf{sk}, x_6, x_7, x_8, x_9, x_{10})$ and a message $m = (\hat{g}, u_1, \ldots, u_n)$. The Receiver takes as input the corresponding public key $\mathsf{pk_X} = (\mathsf{pk}, \hat{X}_6, \hat{X}_7, \hat{X}_8, \hat{X}_9, \hat{X}_{10})$, the message m, and a vector $(m_1, \ldots, m_n) \in (\mathbb{Z}_p^*)^n$.

0. The Receiver checks that in fact $u_i = \hat{g}^{m_i}$ for all $1 \le i \le n$.
1. The Signer acts as the verifier while the Receiver gives a ZKPoK that, for all $1 \le i \le n$, he knows m_i such that $u_i = \hat{g}^{m_i}$. If the verification fails, the Signer denies the Receiver the signature.
2. The Signer computes $\hat{h}$ as in the construction above. He then picks uniformly at random $w \leftarrow \mathbb{Z}_p^*$ and computes $\tilde{h} = \hat{h}^w$ and $\tilde{m} = (\tilde{g}, \tilde{u}_1, \ldots, \tilde{u}_n) = (\hat{g}^w, u_i^w, \ldots, u_n^w)$. He also computes $\tilde{g}^2, \ldots, \tilde{g}^n$. For all $1 \le i \le n$, he forms the message $M_i = (\tilde{g}, \tilde{g}^i, \tilde{g}^n, \tilde{h}, \tilde{u}_i)$ and runs $\sigma_i \leftarrow \mathsf{Sign}_f(\mathsf{sk}, M_i)$. The Signer sends the message $\tilde{m}$ and signature $(\tilde{h}, \sigma = \{\sigma_1, \ldots, \sigma_n\})$ to the Receiver.
3. The Receiver acts as the verifier while the Signer gives a ZKPoK that he has computed the glue element $\tilde{h}$ correctly. If verification of the glue and signature passes, the Receiver outputs the message $\tilde{m}$ and signature $(\tilde{h}, \sigma)$.

The algorithms $\mathsf{Verify_X}$, $\mathsf{ChangeRep_X}$, and $\mathsf{ConvertSig_X}$ must be modified to take as input the message $\tilde{m} = (\tilde{g}, \tilde{u}_1, \ldots, \tilde{u}_n)$:

$\mathsf{Verify_X}(\mathsf{pk_X}, \tilde{m}, (\tilde{h}, \sigma)) \to 0/1$: Form $M_i = (\tilde{g}, \tilde{g}^i, \tilde{g}^n, \tilde{h}, \tilde{u}_i)$ and check whether $\mathsf{Verify}_f(\mathsf{pk}, M_i, \sigma_i) = 1$ for all $1 \le i \le n$.

$\mathsf{ChangeRep_X}(\mathsf{pk_X}, \tilde{m}, (\tilde{h}, \sigma), \mu) \to (\tilde{m}', (\tilde{h}', \sigma'))$: Form $M_i = (\tilde{g}, \tilde{g}^i, \tilde{g}^n, \tilde{h}, \tilde{u}_i)$ and run $(M_i', \sigma_i') \leftarrow \mathsf{ChangeRep}_f(\mathsf{pk}, M_i, \sigma_i, \mu)$ for all $1 \le i \le n$. Output $(\tilde{m}' = (\tilde{g}^\mu, \tilde{u}_1^\mu, \ldots, \tilde{u}_n^\mu), (\tilde{h}' = \tilde{h}^\mu, \sigma' = \{\sigma_1', \ldots, \sigma_n'\}))$.

$\mathsf{ConvertSig_X}(\mathsf{pk_X}, \tilde{m}, (\tilde{h}, \sigma), \rho, \mu) \to (\tilde{m}', (\tilde{h}', \tilde{\sigma}))$: Run $(\tilde{m}', (\tilde{h}', \sigma' = \{\sigma_1', \ldots, \sigma_n'\})) \leftarrow \mathsf{ChangeRep_X}(\mathsf{pk_X}, \tilde{m}, (\tilde{h}, \sigma), \mu)$. Form $M_i' = (\tilde{g}', (\tilde{g}')^i, (\tilde{g}')^n, \tilde{h}', \tilde{u}_i')$ and run $\tilde{\sigma}_i \leftarrow \mathsf{ConvertSig}_f(\mathsf{pk}, M_i', \sigma_i', \rho)$ for all $1 \le i \le n$. Output $(\tilde{m}', (\tilde{h}', \tilde{\sigma} = \{\tilde{\sigma}_1, \ldots, \tilde{\sigma}_n\}))$.

Remark. While the elements $\hat{X}_6 = \hat{P}^{x_6}, \hat{X}_7 = \hat{P}^{x_6 \cdot \hat{x}}, \hat{X}_8 = \hat{P}^{x_8}, \hat{X}_9 = \hat{P}^{x_8 \cdot y_1}, \hat{X}_{10} = \hat{P}^{x_8 \cdot y_2}$ of the public key are not used in signature verification, they are used in Step 3 of the signing protocol. The secret values $\hat{x}, y_1$, and y_2 are defined relative to random bases in order for the hiding proofs to go through. The secret value y is broken into two components, y_1 and y_2, in order for the proof of unforgeability to go through (specifcally, Claim 3).

Efficiency analysis. Group operations and elements for the construction of $\mathsf{MS_X}$ can be found in Figure 4.1. The ZKPoKs are not part of the signature itself and are therefore not counted in the group operations. They can be formed using a combination of Σ-protocols, which can be compiled into efficient non-interactive zero-knowledge proofs using the Fiat-Shamir transform with a reliance on the random oracle model [18]. See Appendix B.

KeyGen	**exp: 10; grp: 10**
Sign	**exp: $9n + 1$; mult: $6n - 2$; grp: $4n + 2$**
Verify	**pair: $8n$; mult: $5n - 1$**
ConvertSK	**field: 10**
ConvertPK	**exp: 10; grp: 10**
ChangeRep	**exp: $4n + 2$; grp: $4n + 2$**
ConvertSig	**exp: $4n + 2$; grp: $4n + 2$**

Fig. 1. Table of efficiency for $\mathsf{MS_X}$. Here, exp denotes the number of group exponentiations, mult denotes the number of group multiplications, and pair denotes the number of pairings. Field operations are ignored. Group and field elements, grp and field, are given as the total number of elements output.

Theorem 2 (Correctness). *Let MS_f be a mercurial signature scheme on message space $(\mathbb{G}_1^*)^5$ as in Theorem 1, and let $\mathsf{MS_X}$ be the variable-length mercurial signature scheme on message space $(\mathbb{G}_1^*)^{n+1}$ constructed above, where all signatures are issued via the interactive signing protocol. Then, $\mathsf{MS_X}$ is correct.*

Correct verification and key conversion can be seen by inspection. We show correct change of message representative, and signature conversion is similar.

Change of message representative: We wish to show that for all messages $m \in \mathcal{M}_n$, for all signatures $(\tilde{h}, \sigma)$ such that $\mathsf{Verify_X}(\mathsf{pk_X}, \tilde{m}, (\tilde{h}, \sigma)) = 1$, for all $\mu \in \mathsf{sample}_\mu$, for all $(\tilde{m}', (\tilde{h}', \sigma')) \in \mathsf{ChangeRep_X}(\mathsf{pk_X}, \tilde{m}, (\tilde{h}, \sigma), \mu)$, it holds that $\mathsf{Verify_X}(\mathsf{pk_X}, \tilde{m}', (\tilde{h}', \sigma')) = 1$, where $\tilde{m}' \in [\tilde{m}]_{\mathcal{R}_m}$. First, observe that the M_i's corresponding to $(\tilde{m}, (\tilde{h}, \sigma = \{\sigma_1, \ldots, \sigma_n\}))$ are $M_i = (\tilde{g}, \tilde{g}^i, \tilde{g}^n, \tilde{h}, \tilde{u}_i)$. $\mathsf{ChangeRep_X}$ invokes $\mathsf{ChangeRep}_f$ as follows: for all $1 \le i \le n$, $\mathsf{ChangeRep}_f(\mathsf{pk}, M_i, \sigma_i, \mu)$ outputs (M_i', σ_i'), where $M_i' = (\tilde{g}^\mu, (\tilde{g}^\mu)^i, (\tilde{g}^\mu)^n, \tilde{h}^\mu, \tilde{u}_i^\mu)$. By correct change of message representative of $\mathsf{ChangeRep}_f$ (Theorem 1), we have that $\mathsf{Verify}_f(\mathsf{pk}, M_i', \sigma_i') = 1$ for all $1 \le i \le n$, which implies that $\mathsf{Verify_X}(\mathsf{pk_X}, \tilde{m}', (\tilde{h}', \sigma')) = 1$, where $\tilde{m}' = (\tilde{g}^\mu, \tilde{u}_1^\mu, \ldots, \tilde{u}_n^\mu) \in [\tilde{m}]_{\mathcal{R}_m}$.

4.2 Origin-hiding

Theorem 3 (Origin-hiding). *Let MS_f be a mercurial signature scheme on message space $(\mathbb{G}_1^*)^5$ as in Theo-*

rem 1, and let $\mathsf{MS_X}$ *be the variable-length mercurial signature scheme on message space* $(\mathbb{G}_1^*)^{n+1}$ *constructed above. Suppose all signatures are issued via the interactive signing protocol described in Section 4.1, where the proof system used in Step 3 is sound under sequential (or concurrent) composition. Then,* $\mathsf{MS_X}$ *is origin-hiding under sequential (or concurrent) composition.*

Origin-hiding of $\mathsf{ChangeRep_X}$**:** Let $\mathsf{pk}_\mathsf{X}^*, \tilde{m}, (\tilde{h}, \sigma = \{\sigma_1, \ldots, \sigma_n\})$ be such that $\mathsf{Verify_X}(\mathsf{pk}_\mathsf{X}^*, \tilde{m}, (\tilde{h}, \sigma)) = 1$, where pk_X^* is possibly adversarially generated. $\mathsf{ChangeRep_X}(\mathsf{pk}_\mathsf{X}^*, \tilde{m}, (\tilde{h}, \sigma), \mu)$ outputs $(\tilde{m}', (\tilde{h}', \sigma')) = (\tilde{m}^\mu, (\tilde{h}^\mu, \{\sigma_1', \ldots, \sigma_n'\}))$, where $\tilde{m}^\mu$ is shorthand for $\tilde{m}^\mu = (\tilde{g}^\mu, \tilde{u}_1^\mu, \ldots, \tilde{u}_n^\mu)$. By soundness of the ZKPoK in Step 3 of the signing protocol, the glue element $\tilde{h}$ is computed correctly with overwhelming probability. The M_i's corresponding to $(\tilde{m}, (\tilde{h}, \sigma))$ are $M_i = (\tilde{g}, \tilde{g}^i, \tilde{g}^n, \tilde{h}, \tilde{u}_i)$. $\mathsf{ChangeRep_X}$ invokes $\mathsf{ChangeRep}_f$ as follows: for all $1 \leq i \leq n$, $\mathsf{ChangeRep}_f(\mathsf{pk}, M_i, \sigma_i, \mu)$ outputs (M_i', σ_i'), where $M_i' = (\tilde{g}^\mu, (\tilde{g}^\mu)^i, (\tilde{g}^\mu)^n, \tilde{h}^\mu, \tilde{u}_i^\mu)$. By origin-hiding of $\mathsf{ChangeRep}_f$ (Theorem 1), σ_i' is distributed the same as a fresh signature on M_i' for all $1 \leq i \leq n$. Note that the glue element $\tilde{h}^\mu$ is correct if $\tilde{h}$ is correct, and $\tilde{h}^\mu$ is distributed the same as a fresh glue element for a fresh signature on $\tilde{m}^\mu$. Thus, $\tilde{m}^\mu$ is a uniformly random element of $[\tilde{m}]_{\mathcal{R}_m}$, and $(\tilde{h}^\mu, (\sigma_1', \ldots, \sigma_n'))$ is a uniformly random element in the space of signatures $(\bar{h}, \bar{\sigma})$ satisfying $\mathsf{Verify_X}(\mathsf{pk}_\mathsf{X}^*, \tilde{m}^\mu, (\bar{h}, \bar{\sigma})) = 1$ with overwhelming probability.

Origin-hiding of $\mathsf{ConvertSig_X}$**:** Let $\mathsf{pk}_\mathsf{X}^*, \tilde{m}, (\tilde{h}, \sigma = \{\sigma_1, \ldots, \sigma_n\})$ be such that $\mathsf{Verify_X}(\mathsf{pk}_\mathsf{X}^*, \tilde{m}, (\tilde{h}, \sigma)) = 1$, where pk_X^* is possibly adversarially generated. $\mathsf{ConvertSig_X}(\mathsf{pk}_\mathsf{X}^*, \tilde{m}, (\tilde{h}, \sigma), \rho, \mu)$ outputs $(\tilde{m}', (\tilde{h}', \tilde{\sigma})) = (\tilde{m}^\mu, (\tilde{h}^\mu, (\tilde{\sigma}_1, \ldots, \tilde{\sigma}_n)))$, where $\tilde{m}^\mu$ is shorthand for $\tilde{m}^\mu = (\tilde{g}^\mu, \tilde{u}_1^\mu, \ldots, \tilde{u}_n^\mu)$. By soundness of the ZKPoK in Step 3 of the signing protocol, the glue element $\tilde{h}$ is computed correctly with overwhelming probability. The M_i's corresponding to $(\tilde{m}, (\tilde{h}, \sigma))$ are $M_i = (\tilde{g}, \tilde{g}^i, \tilde{g}^n, \tilde{h}, \tilde{u}_i)$. The output of $\mathsf{ConvertSig_X}$ is computed in two steps. First, $\mathsf{ChangeRep_X}(\mathsf{pk}_\mathsf{X}^*, \tilde{m}, (\tilde{h}, \sigma), \mu)$ outputs $(\tilde{m}', (\tilde{h}', \sigma')) = (\tilde{m}^\mu, (\tilde{h}^\mu, (\sigma_1', \ldots, \sigma_n')))$. Then, $\mathsf{ConvertSig}_f(\mathsf{pk}^*, M_i', \sigma_i', \rho)$ outputs $\tilde{\sigma}_i$ for all $1 \leq i \leq n$. $\mathsf{ChangeRep_X}$ is origin-hiding, as shown above, and $\mathsf{ConvertSig}_f$ is origin-hiding by Theorem 1. Thus, $\tilde{m}^\mu$ is a uniformly random element of $[\tilde{m}]_{\mathcal{R}_m}$, and $(\tilde{h}^\mu, (\tilde{\sigma}_1, \ldots, \tilde{\sigma}_n))$ is a uniformly random element in the space of signatures $(\bar{h}, \bar{\sigma})$ satisfying $\mathsf{Verify_X}(\mathsf{ConvertPK_X}(\mathsf{pk}_\mathsf{X}^*, \rho), \tilde{m}^\mu, (\bar{h}, \bar{\sigma})) = 1$ with overwhelming probability (where $\mathsf{ConvertPK_X}(\mathsf{pk}_\mathsf{X}^*, \rho) = (\mathsf{pk}_\mathsf{X}^*)^\rho$ is a uniformly random element of $[\mathsf{pk}_\mathsf{X}^*]_{\mathcal{R}_\mathsf{pk}}$). Note that origin-hiding does not hold if $\tilde{h} = 1$, but this occurs with negligible probability.

4.3 Unforgeability

Unforgeability of $\mathsf{MS_X}$ holds under a variant of the asymmetric bilinear decisional Diffie-Hellman assumption (ABDDH$^+$) introduced by Fuchsbauer et al. [21].

Definition 9 (ABDDH$^+$ assumption [21])**.** Let BGGen be a bilinear group generator that outputs $\mathsf{BG} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, P, \hat{P}, e)$. The *ABDDH$^+$ assumption* holds in $\mathbb{G}_1$ if for all probabilistic, polynomial-time (PPT) algorithms $\mathcal{A}$, there exists a negligible function ν such that:

$$\begin{aligned}&\Pr[b \leftarrow \{0,1\}; \mathsf{BG} \leftarrow \mathsf{BGGen}(1^k); u, v, w, r \leftarrow \mathbb{Z}_p^*;\\ &\quad b^* \leftarrow \mathcal{A}(\mathsf{BG}, \hat{P}^u, \hat{P}^v, P^u, P^{uv}, P^w, P^{(1-b)\cdot r + b\cdot(wuv)})\\ &\quad : b^* = b] - \tfrac{1}{2} \leq \nu(k)\end{aligned}$$

Proposition 1. [21] *The ABDDH$^+$ assumption holds in generic groups.*

Theorem 4 (Unforgeability)**.** *Let* MS_f *be a mercurial signature scheme on message space* $(\mathbb{G}_1^*)^5$ *as in Theorem 1, and let* $\mathsf{MS_X}$ *be the variable-length mercurial signature scheme on message space* $(\mathbb{G}_1^*)^{n+1}$ *constructed above. Suppose all signatures are issued via the interactive signing protocol described in Section 4.1, where the proof system used in Step 1 is extractable under sequential (or concurrent) composition. Then, unforgeability of* $\mathsf{MS_X}$ *holds sequentially (or concurrently) under the discrete logarithm (DL) assumption in* $\mathbb{G}_2$ *and the ABDDH$^+$ assumption in* $\mathbb{G}_1$*. The same holds when* $\mathbb{G}_1$ *and* $\mathbb{G}_2$ *are swapped.*

Proof. We wish to show that if there exists a probabilistic, polynomial-time (PPT) adversary $\mathcal{A}$ that breaks unforgeability of $\mathsf{MS_X}$ with non-negligible probability, then we can construct a PPT adversary $\mathcal{A}'$ that breaks unforgeability of MS_f with non-negligible probability, or the discrete logarithm (DL) or ABDDH$^+$ assumption doesn't hold.

Suppose there exists such a PPT adversary $\mathcal{A}$. Then, we construct a PPT adversary $\mathcal{A}'$ as a reduction $\mathcal{B}_{\mathsf{MS}_f}$ running $\mathcal{A}$ as a subroutine. We construct the reduction $\mathcal{B}_{\mathsf{MS}_f}$ for breaking unforgeability of MS_f as follows. $\mathcal{B}_{\mathsf{MS}_f}$ receives as input public parameters $PP = \mathsf{BG} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, P, \hat{P}, e)$ and a fixed public key $\mathsf{pk} = (\hat{X}_1, \hat{X}_2, \hat{X}_3, \hat{X}_4, \hat{X}_5)$ for the mercurial signature scheme MS_f on messages of length $\ell = 5$ for which he will try to produce a forgery. He chooses uniformly at random a secret point $\hat{x} \leftarrow \mathbb{Z}_p^*$ and secret seeds $y_1, y_2 \leftarrow \mathbb{Z}_p^*$. He also picks $x_6, x_8 \leftarrow \mathbb{Z}_p^*$ and sets $x_7 = x_6 \cdot \hat{x}$ and $x_9 = x_8 \cdot y_1$ and $x_{10} = x_8 \cdot y_2$. He then sets $\mathsf{pk}_\mathsf{X} = (\mathsf{pk}, \hat{X}_6, \hat{X}_7, \hat{X}_8, \hat{X}_9, \hat{X}_{10})$, where $\hat{X}_i = \hat{P}^{x_i}$. $\mathcal{B}_{\mathsf{MS}_f}$ forwards $PP_\mathsf{X} = PP$ and pk_X to $\mathcal{A}$ and acts as $\mathcal{A}$'s challenger $\mathcal{C}$. As in the unforgeability game for MS_f, $\mathcal{B}_{\mathsf{MS}_f}$ has access to a signing oracle $\mathsf{Sign}_f(\mathsf{sk}, \cdot)$, where sk is the secret key corresponding to pk. $\mathcal{A}$ proceeds to make signature queries on messages of the form $m = (\hat{g}, u_1, \ldots, u_n) \in (\mathbb{G}_1^*)^{n+1}$. For each signature query, $\mathcal{B}_{\mathsf{MS}_f}$ acts as the verifier while $\mathcal{A}$ gives a ZKPoK that, for all $1 \le i \le n$, he knows m_i such that $u_i = \hat{g}^{m_i}$. If the verification fails, $\mathcal{B}_{\mathsf{MS}_f}$ denies $\mathcal{A}$ the signature; otherwise, $\mathcal{B}_{\mathsf{MS}_f}$ computes $y = y_1 \cdot y_2$ and $\hat{h} = \left(\prod_{i=1}^n u_i^{\hat{x}^{i-1}}\right)^y$. $\mathcal{B}_{\mathsf{MS}_f}$ picks uniformly at random $w \leftarrow \mathbb{Z}_p^*$ and computes $\tilde{g} = \hat{g}^w, \tilde{h} = \hat{h}^w$, and $\tilde{u}_i = u_i^w$ for all $1 \le i \le n$. He also computes $\tilde{g}^2, \ldots, \tilde{g}^n$. He forwards n messages of the form $M_i = (\tilde{g}, \tilde{g}^i, \tilde{g}^n, \tilde{h}, \tilde{u}_i)$ to his signing oracle $\mathsf{Sign}_f(\mathsf{sk}, \cdot)$ and receives n signatures $\sigma_1, \ldots, \sigma_n$. He sends the message $\tilde{m} = (\tilde{g}, \tilde{u}_1, \ldots, \tilde{u}_n)$ and the signature $(\tilde{h}, \sigma = \{\sigma_1, \ldots, \sigma_n\})$ to $\mathcal{A}$, along with a ZKPoK that $\tilde{h}$ was computed correctly.

After some polynomial number of signature queries, $\mathcal{A}$ produces a forgery $(\mathsf{pk}_\mathsf{X}^*, \tilde{m}^*, (\tilde{h}^*, \sigma^*))$, where $\mathsf{pk}_\mathsf{X}^* = (\mathsf{pk}^*, \hat{X}_6^*, \hat{X}_7^*, \hat{X}_8^*, \hat{X}_9^*, \hat{X}_{10}^*)$, $\tilde{m}^* = (\tilde{g}^*, \tilde{u}_1^*, \ldots, \tilde{u}_n^*)$, and $\sigma^* = \{\sigma_1^*, \ldots, \sigma_n^*\}$. $\mathcal{A}$'s forgery can be represented as a set of messages that are in the message space of MS_f: $M_1^* = (\tilde{g}^*, (\tilde{g}^*)^1, (\tilde{g}^*)^n, \tilde{h}^*, \tilde{u}_1^*), M_2^* = (\tilde{g}^*, (\tilde{g}^*)^2, (\tilde{g}^*)^n, \tilde{h}^*, \tilde{u}_2^*), \ldots, M_n^* = (\tilde{g}^*, (\tilde{g}^*)^n, (\tilde{g}^*)^n, \tilde{h}^*, \tilde{u}_n^*)$. $\mathcal{B}_{\mathsf{MS}_f}$ chooses $i \leftarrow \{1, \ldots, n\}$ uniformly at random and outputs $(\mathsf{pk}^*, M_i^*, \sigma_i^*)$ as his forgery. Let us analyze $\mathcal{B}_{\mathsf{MS}_f}$'s success probability.

Suppose $\mathcal{A}$'s forgery $(\mathsf{pk}_\mathsf{X}^*, \tilde{m}^*, (\tilde{h}^*, \sigma^*))$ is successful. Then, by definition, it satisfies $[\mathsf{pk}_\mathsf{X}^*]_{\mathcal{R}_\mathsf{pk}} = [\mathsf{pk}_\mathsf{X}]_{\mathcal{R}_\mathsf{pk}}$ and $\forall\ \bar{m} \in Q, [\tilde{m}^*]_{\mathcal{R}_m} \neq [m]_{\mathcal{R}_m}$ and $\mathsf{Verify}_\mathsf{X}(\mathsf{pk}_\mathsf{X}^*, \tilde{m}^*, (\tilde{h}^*, \sigma^*)) = 1$, where Q is the set of discrete logarithms $\bar{m} = \{m_1, \ldots, m_n\} \in (\mathbb{Z}_p^*)^n$ of messages m that $\mathcal{A}$ has queried to the signing oracle. Note that the forged $\tilde{g}^*$ and $\tilde{h}^*$ must be repeated for each message M_i^* because the verification algorithm accepts the signature $(\tilde{h}^*, \sigma^* = \{\sigma_1^*, \ldots, \sigma_n^*\})$.

There are two ways in which the forged message $\tilde{m}^*$ could have been derived by $\mathcal{A}$:

(1) Good Case: There exists some $i \in \{1, \ldots, n\}$ for which $[M_i^*]_{\mathcal{R}_m} \neq [M]_{\mathcal{R}_m}$ for any M previously queried by $\mathcal{B}_{\mathsf{MS}_f}$ to his signing oracle. We will see that with overwhelming probability, the Good Case is the way in which $\mathcal{A}$ forms his forgery.

(2) Bad Case: Every M_i^* is such that $[M_i^*]_{\mathcal{R}_m} = [M]_{\mathcal{R}_m}$ for some M previously queried by $\mathcal{B}_{\mathsf{MS}_f}$ to his signing oracle. In this case, $\mathcal{A}$ is able to "mix and match" m_i's from different messages for which signatures have been issued. We claim that $\mathcal{A}$ cannot do this, except with negligible probability, or the DL or ABDDH⁺ assumption doesn't hold.

First, note that if a glue element $\tilde{h}$ is formed as $\tilde{g}^{R(m_1,\ldots,m_n)}$ for some random function $R : (\mathbb{Z}_p^*)^n \to \mathbb{Z}_p^*$, then $\mathcal{A}$ cannot mix and match. This is because if the vectors $(m_1, \ldots, m_n)$ are distinct, then the values $R(m_1, \ldots, m_n)$ are distinct as well as the glue elements $\tilde{g}^{R(m_1,\ldots,m_n)}$. Our goal is to demonstrate that a glue element formed as $\tilde{g}^{R(m_1,\ldots,m_n)}$ is indistinguishable from a real glue element $\tilde{g}^{y\cdot q}$, where $q = p(\hat{x}) = \sum_{i=1}^n m_i \hat{x}^{i-1}$. Then, $\mathcal{A}$ can't mix and match when real glue elements are used, except with negligible probability.

We achieve this goal in two steps. We first demonstrate that $\tilde{g}^{R(m_1,\ldots,m_n)}$ is indistinguishable from $\tilde{g}^{R(q)}$, where $R : \mathbb{Z}_p^* \to \mathbb{Z}_p^*$ is a random function, under the DL assumption. We then demonstrate that $\tilde{g}^{R(q)}$ is indistinguishable from a real glue element $\tilde{g}^{y\cdot q}$ under the ABDDH⁺ assumption. This gives the desired result.

Consider the following set of games. In Game 0, the real signing game, the glue element is computed directly, without extraction of the m_i's or simulated proofs. Game 1 includes simulated proofs. In Games 2-5, the challenger acts as the zero-knowledge extractor to extract the m_i's necessary to compute the glue element and provides a simulated proof that it was computed correctly. The overall proof structure is as follows. Arrows indicate why consecutive games are indistinguishable.

Game 0. $\tilde{h} = \tilde{g}^{y\cdot q}$. No extraction or simulation. This is the real signing game.

$\updownarrow$ Claim 1: zero-knowledge property

Game 1. $\tilde{h} = \tilde{g}^{y\cdot q}$. No extraction, but simulation.

$\updownarrow$ Claim 2: knowledge extractor property

Game 2. $\tilde{h} = \tilde{g}^{y\cdot q}$, $q = p(\hat{x})$. Extraction and simulation henceforth.

$\updownarrow$ Claim 3: ABDDH⁺ assumption in $\mathbb{G}_1$

Game 3. $\tilde{h} = \tilde{g}^{R(q)}$, $q = p(\hat{x})$, $R : \mathbb{Z}_p^* \to \mathbb{Z}_p^*$ random.

$\updownarrow$ Claim 4: polynomial collision argument / Claim 5: DL assumption in $\mathbb{G}_2$

Game 4. $\tilde{h} = \tilde{g}^{R(\dot{q})}$, $\dot{q} = p(\alpha)$ for "fake" secret $\alpha \in \mathbb{Z}_p^*$.

$\updownarrow$ Claim 6: polynomial collision argument

Game 5. $\tilde{h} = \tilde{g}^{R(m_1,\ldots,m_n)}$, $R : (\mathbb{Z}_p^*)^n \to \mathbb{Z}_p^*$ random.

We now provide descriptions of the games and proofs of the claims.

Game 0. In this real signing game, the glue element is $\tilde{h} = \tilde{g}^{y \cdot q}$. There is no extraction or zero-knowledge simulation.

The challenger $\mathcal{C}$ computes the public parameters PP and keys $(\mathsf{pk}, \mathsf{sk}) = ((\hat{X}_1,\ \hat{X}_2,\ \hat{X}_3,\ \hat{X}_4,\ \hat{X}_5), (x_1, x_2, x_3, x_4, x_5))$ for a mercurial signature scheme MS_f on messages of length $\ell = 5$. $\mathcal{C}$ chooses uniformly at random a secret point $\hat{x} \leftarrow \mathbb{Z}_p^*$ and secret seeds $y_1, y_2 \leftarrow \mathbb{Z}_p^*$. He also picks $x_6, x_8 \leftarrow \mathbb{Z}_p^*$ and sets $x_7 = x_6 \cdot \hat{x}$ and $x_9 = x_8 \cdot y_1$ and $x_{10} = x_8 \cdot y_2$. He then sets $\mathsf{pk}_\mathsf{X} = (\mathsf{pk}, \hat{X}_6, \hat{X}_7, \hat{X}_8, \hat{X}_9, \hat{X}_{10})$, where $\hat{X}_i = \hat{P}^{x_i}$. $\mathcal{C}$ forwards PP_X and pk_X to $\mathcal{A}$.

$\mathcal{A}$ proceeds to make signature queries on messages of the form $m = (\hat{g}, u_1, \ldots, u_n) \in (\mathbb{G}_1^*)^{n+1}$. For each signature query, $\mathcal{C}$ acts as the verifier while $\mathcal{A}$ gives a ZKPoK that, for all $1 \leq i \leq n$, he knows m_i such that $u_i = \hat{g}^{m_i}$. If the verification fails, $\mathcal{C}$ denies $\mathcal{A}$ the signature; otherwise, $\mathcal{C}$ computes $y = y_1 \cdot y_2$ and $\hat{h} = \left(\prod_{i=1}^n u_i^{\hat{x}^{i-1}} \right)^y$. $\mathcal{C}$ picks uniformly at random $w \leftarrow \mathbb{Z}_p^*$ and computes $\tilde{g} = \hat{g}^w, \tilde{h} = \hat{h}^w$, and $\tilde{u}_i = u_i^w\ \forall i$. He also computes $\tilde{g}^2, \ldots, \tilde{g}^n$. He then signs n messages of the form $M_i = (\tilde{g}, \tilde{g}^i, \tilde{g}^n, \tilde{h}, \tilde{u}_i)$ using his secret key sk for MS_f and sends $\tilde{m} = (\tilde{g}, \tilde{u}_1, \ldots, \tilde{u}_n)$ and $(\tilde{h}, \sigma = \{\sigma_1, \ldots, \sigma_n\})$ to $\mathcal{A}$, along with a ZKPoK that $\tilde{h}$ was computed correctly. $\mathcal{A}$ issues queries for signatures on messages a polynomial number of times. The game ends when $\mathcal{A}$ produces a forgery or terminates without producing a forgery.

Game 1. In this game, the glue element remains $\tilde{h} = \tilde{g}^{y \cdot q}$. There is no extraction, but now there is simulation.

Game 1 is the same as Game 0, except the challenger $\mathcal{C}$ simulates the ZKPoK that $\tilde{h}$ was computed correctly.

Claim 1. *A PPT adversary cannot distinguish Game 0 from Game 1, except with negligible probability.*

The only difference between the two games is zero-knowledge simulation. In Game 1, the challenger simulates the ZKPoK that the glue $\tilde{h}$ was computed correctly, whereas in Game 0, the challenger gives a real ZKPoK. If an adversary could distinguish the two games, it would break the zero-knowledge property.

Game 2. In this game, the glue element remains $\tilde{h} = \tilde{g}^{y \cdot q}$, where $q = p(\hat{x})$. There is now extraction and simulation (and for all games henceforth).

The challenger $\mathcal{C}$ computes the public parameters PP and keys $(\mathsf{pk}, \mathsf{sk}) = ((\hat{X}_1,\ \hat{X}_2,\ \hat{X}_3,\ \hat{X}_4,\ \hat{X}_5), (x_1, x_2, x_3, x_4, x_5))$ for a mercurial signature scheme MS_f on messages of length $\ell = 5$. $\mathcal{C}$ chooses uniformly at random a secret point $\hat{x} \leftarrow \mathbb{Z}_p^*$ and secret seeds $y_1, y_2 \leftarrow \mathbb{Z}_p^*$. He also picks $x_6, x_8 \leftarrow \mathbb{Z}_p^*$ and sets $x_7 = x_6 \cdot \hat{x}$ and $x_9 = x_8 \cdot y_1$ and $x_{10} = x_8 \cdot y_2$. He then sets $\mathsf{pk}_\mathsf{X} = (\mathsf{pk}, \hat{X}_6, \hat{X}_7, \hat{X}_8, \hat{X}_9, \hat{X}_{10})$, where $\hat{X}_i = \hat{P}^{x_i}$. $\mathcal{C}$ forwards PP_X and pk_X to $\mathcal{A}$.

$\mathcal{A}$ proceeds to make signature queries on messages of the form $m = (\hat{g}, u_1, \ldots, u_n) \in (\mathbb{G}_1^*)^{n+1}$. For each signature query, $\mathcal{C}$ acts as the extractor while $\mathcal{A}$ gives a ZKPoK that, for all $1 \leq i \leq n$, he knows m_i such that $u_i = \hat{g}^{m_i}$. $\mathcal{C}$ extracts the m_i's, or if the extraction fails, $\mathcal{C}$ denies $\mathcal{A}$ the signature. Otherwise, $\mathcal{C}$ computes the polynomial $p(x) = m_1 + m_2 x + \cdots + m_n x^{n-1}$ and evaluates $p(x)$ at the secret point $\hat{x}$. Let $q = p(\hat{x})$ denote this evaluation. $\mathcal{C}$ computes $y = y_1 \cdot y_2$ and $\hat{h} = \hat{g}^{y \cdot q}$. $\mathcal{C}$ picks uniformly at random $w \leftarrow \mathbb{Z}_p^*$ and computes $\tilde{g} = \hat{g}^w, \tilde{h} = \hat{h}^w$, and $\tilde{u}_i = u_i^w\ \forall i$. He also computes $\tilde{g}^2, \ldots, \tilde{g}^n$. He then signs n messages of the form $M_i = (\tilde{g}, \tilde{g}^i, \tilde{g}^n, \tilde{h}, \tilde{u}_i)$ using his secret key sk for MS_f and sends $\tilde{m} = (\tilde{g}, \tilde{u}_1, \ldots, \tilde{u}_n)$ and $(\tilde{h}, \sigma = \{\sigma_1, \ldots, \sigma_n\})$ to $\mathcal{A}$, along with a simulated ZKPoK that $\tilde{h}$ was computed correctly. $\mathcal{A}$ issues queries for signatures on messages a polynomial number of times. The game ends when $\mathcal{A}$ produces a forgery or terminates without producing a forgery.

Claim 2. *A PPT adversary cannot distinguish Game 1 from Game 2, except with negligible probability.*

In Game 2, the challenger $\mathcal{C}$ extracts the m_i's from the message m, forms the polynomial $p(x) = m_1 + m_2 x + \cdots + m_n x^{n-1}$, and evaluates $q = p(\hat{x})$. $\mathcal{C}$ then forms the glue element as $\tilde{h} = \tilde{g}^{y \cdot q}$, where $\tilde{g} = \hat{g}^w$ for some uniformly random $w \leftarrow \mathbb{Z}_p^*$. In Game 1, the challenger $\mathcal{C}$ forms the glue element as $\tilde{h} = \left(\prod_{i=1}^n (u_i^w)^{\hat{x}^{i-1}} \right)^y$ for a uniformly random $w \leftarrow \mathbb{Z}_p^*$. But note that $\tilde{h} = \left(\prod_{i=1}^n (\hat{g}^{m_i \cdot w})^{\hat{x}^{i-1}} \right)^y = \left(\prod_{i=1}^n \tilde{g}^{y \cdot m_i \cdot \hat{x}^{i-1}} \right) = \tilde{g}^{y \cdot q}$.

Thus, the glue elements $\tilde{h}$ in both games are identical. The only difference between the two games is extraction. In Game 2, the challenger extracts the m_i's to compute the glue $\tilde{h}$, whereas in Game 1, the challenger computes the correct $\tilde{h}$ directly from the u_i's, without extracting the m_i's. If an adversary could distinguish the two games, it would break the knowledge extractor property.

Game 3. In this game, the glue element is $\tilde{h} = \tilde{g}^{R(q)}$, where $q = p(\hat{x})$ and $R : \mathbb{Z}_p^* \to \mathbb{Z}_p^*$ is a random function.

Game 3 is the same as Game 2, except the challenger $\mathcal{C}$ chooses a random function $R : \mathbb{Z}_p^* \to \mathbb{Z}_p^*$ and for each signature computes $\hat{h} = \hat{g}^{R(q)}$, where $q = p(\hat{x})$. The rest of the signing protocol is carried out as in Game 2.

Claim 3. *A PPT adversary cannot distinguish Game 2 from Game 3 under the ABDDH⁺ assumption in $\mathbb{G}_1$.*

Consider the following decisional problem related to the ABDDH⁺ assumption.

Definition 10. (ABDDH† problem). Let BGGen be a bilinear group generator that outputs $\mathsf{BG} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, P, \hat{P}, e)$. The ABDDH† problem in $\mathbb{G}_1$ is to distinguish between the distributions D_0 and D_1 defined by:

$$\begin{aligned} D_0 =& \{\mathsf{BG} \leftarrow \mathsf{BGGen}(1^k);\ \alpha, \beta, u, v, \omega \leftarrow \mathbb{Z}_p^*; \\ & (\mathsf{BG}, \hat{P}^{\alpha}, \hat{P}^{\alpha u}, \hat{P}^{\alpha v}, P^{\beta}, P^{\beta uv}, P^{\omega}, P^{\omega uv})\} \\ D_1 =& \{\mathsf{BG} \leftarrow \mathsf{BGGen}(1^k);\ \alpha, \beta, u, v, \omega, r \leftarrow \mathbb{Z}_p^*; \\ & (\mathsf{BG}, \hat{P}^{\alpha}, \hat{P}^{\alpha u}, \hat{P}^{\alpha v}, P^{\beta}, P^{\beta uv}, P^{\omega}, P^{r}) \end{aligned}$$

Lemma 1. *If the ABDDH⁺ assumption holds for a bilinear group generator* BGGen*, then the ABDDH† problem is also hard for* BGGen.

Indeed, a reduction $\mathcal{B}$ given an ABDDH⁺ instance

$$(\mathsf{BG}, \hat{P}^{u}, \hat{P}^{v}, P^{u}, P^{uv}, P^{\omega}, P^{(1-b)\cdot r + b\cdot(\omega uv)})$$

can pick uniformly at random $\alpha, \beta \leftarrow \mathbb{Z}_p^*$ and provide an ABDDH† instance

$$(\mathsf{BG}, \hat{P}^{\alpha}, (\hat{P}^{u})^{\alpha}, (\hat{P}^{v})^{\alpha}, P^{\beta}, (P^{uv})^{\beta}, P^{\omega}, P^{(1-b)\cdot r + b\cdot(\omega uv)})$$

to an adversary $\mathcal{A}$ whose non-negligible advantage in distinguishing ABDDH† tuples becomes $\mathcal{B}$'s non-negligible advantage in breaking ABDDH⁺.

We now prove Claim 3 via a hybrid argument. Let $\Gamma(k)$ be a polynomial. For $0 \leq i \leq \Gamma(k)$, let $\mathcal{H}_i$ be the hybrid experiment defined as the following game. The challenger $\mathcal{C}$ computes the public parameters PP and keys $(\mathsf{pk}, \mathsf{sk}) = ((\hat{X}_1, \hat{X}_2, \hat{X}_3, \hat{X}_4, \hat{X}_5), (x_1, x_2, x_3, x_4, x_5))$ for a mercurial signature scheme MS_f on messages of length $\ell = 5$. $\mathcal{C}$ chooses uniformly at random a secret point $\hat{x} \leftarrow \mathbb{Z}_p^*$ and secret seeds $y_1, y_2 \leftarrow \mathbb{Z}_p^*$. He also picks $x_6, x_8 \leftarrow \mathbb{Z}_p^*$ and sets $x_7 = x_6 \cdot \hat{x}$ and $x_9 = x_8 \cdot y_1$ and $x_{10} = x_8 \cdot y_2$. He then sets $\mathsf{pk}_{\mathsf{X}} = (\mathsf{pk}, \hat{X}_6, \hat{X}_7, \hat{X}_8, \hat{X}_9, \hat{X}_{10})$, where $\hat{X}_i = \hat{P}^{x_i}$. He also chooses a random function $R : \mathbb{Z}_p^* \to \mathbb{Z}_p^*$ and forwards PP_{X} and pk_{X} to $\mathcal{A}$.

Let $\mathcal{A}$'s j^{th} signature query be on message $m_j = (\hat{g}_j, u_{j,1}, \ldots, u_{j,n})$. $\mathcal{C}$ acts as the extractor while $\mathcal{A}$ gives a ZKPoK that, for all $1 \leq i \leq n$, he knows $m_{j,i}$ such that $u_{j,i} = \hat{g}_j^{m_{j,i}}$. $\mathcal{C}$ extracts the $m_{j,i}$'s, or if the extraction fails, $\mathcal{C}$ denies $\mathcal{A}$ the signature. Otherwise, $\mathcal{C}$ computes the polynomial $p_j(x) = m_{j,1} + m_{j,2}x + \cdots + m_{j,n}x^{n-1}$ and evaluates $q_j = p_j(\hat{x})$. $\mathcal{C}$ also computes $y = y_1 \cdot y_2$.

(1) If $j \leq i$, $\mathcal{C}$ computes $R(q_j)$ and $\hat{h}_j = \hat{g}_j^{R(q_j)}$. $\mathcal{C}$ picks uniformly at random $w_j \leftarrow \mathbb{Z}_p^*$ and computes $\tilde{g}_j = \hat{g}_j^{w_j}, \tilde{h}_j = \hat{h}_j^{w_j}$, and $\tilde{u}_{j,i} = u_{j,i}^{w_j}$ $\forall i$. He also computes $\tilde{g}_j^2, \ldots, \tilde{g}_j^n$. He then signs n messages of the form $M_{j,i} = (\tilde{g}_j, \tilde{g}_j^i, \tilde{g}_j^n, \tilde{h}_j, \tilde{u}_{j,i})$ using his secret key sk for MS_f and sends $\tilde{m}_j = (\tilde{g}_j, \tilde{u}_{j,1}, \ldots, \tilde{u}_{j,n})$ and $(\tilde{h}_j, \sigma_j = \{\sigma_{j,1}, \ldots, \sigma_{j,n}\})$ to $\mathcal{A}$, along with a simulated ZKPoK that $\tilde{h}_j$ was computed correctly.

(2) If $j > i$, $\mathcal{C}$ computes: $\hat{h}_j = \hat{g}_j^{y\cdot q_j}$. $\mathcal{C}$ picks uniformly at random $w_j \leftarrow \mathbb{Z}_p^*$ and computes $\tilde{g}_j = \hat{g}_j^{w_j}, \tilde{h}_j = \hat{h}_j^{w_j}$, and $\tilde{u}_{j,i} = u_{j,i}^{w_j}$ $\forall i$. He also computes $\tilde{g}_j^2, \ldots, \tilde{g}_j^n$. He then signs n messages of the form $M_{j,i} = (\tilde{g}_j, \tilde{g}_j^i, \tilde{g}_j^n, \tilde{h}_j, \tilde{u}_{j,i})$ using his secret key sk for MS_f and sends $\tilde{m}_j = (\tilde{g}_j, \tilde{u}_{j,1}, \ldots, \tilde{u}_{j,n})$ and $(\tilde{h}_j, \sigma_j = \{\sigma_{j,1}, \ldots, \sigma_{j,n}\})$ to $\mathcal{A}$, along with a simulated ZKPoK that $\tilde{h}_j$ was computed correctly.

By definition, $\mathcal{H}_0$ corresponds to the game in which all glue elements are formed as $\tilde{h}_j = \tilde{g}_j^{y\cdot q_j}$ (Game 2), while $\mathcal{H}_{\Gamma(k)}$ corresponds to the game in which all glue elements are formed as $\tilde{h}_j = \tilde{g}_j^{R(q_j)}$ (Game 3).

Let $\mathcal{A}$ be an adversary, let $\Gamma(k)$ be the number of queries $\mathcal{A}$ makes, and let $0 \leq i \leq \Gamma(k) - 1$. We wish to show that $\mathcal{A}$'s advantage $\epsilon = \mathsf{Adv}(\mathcal{A}, k, i)$ in distinguishing $\mathcal{H}_i$ from $\mathcal{H}_{i+1}$ is negligible; in fact, $\epsilon \leq \nu$, where ν is the best advantage in distinguishing ABDDH† tuples.

Suppose not; that is, suppose $\epsilon = \mathsf{Adv}(\mathcal{A}, k, i) > \nu$ for some $\mathcal{A}, k, i$. Then, let us show that there exists a probabilistic, polynomial-time $\mathcal{B}$ that can distinguish between the distributions D_0 and D_1.

We construct $\mathcal{B}$ as a reduction running $\mathcal{A}$ as a subroutine. $\mathcal{B}$ serves as the challenger for $\mathcal{A}$ in the hybrid game and as the adversary for his own challenger in the ABDDH† game. $\mathcal{B}$ receives as input $(\mathsf{BG}, \hat{A}_0, \hat{A}_1, \hat{A}_2, B_1, C, B_2, D)$, where implicitly $\hat{A}_0 = \hat{P}^{\alpha}, \hat{A}_1 = \hat{P}^{\alpha u}, \hat{A}_2 = \hat{P}^{\alpha v}, B_1 = P^{\beta}, C = P^{\beta uv}, B_2 = P^{\omega}$, and $D = P^{\omega uv}$ or P^{r} for some uniformly random $\alpha, \beta, u, v, \omega, r \in \mathbb{Z}_p^*$.

$\mathcal{B}$ computes the public parameters PP and keys $(\mathsf{pk}, \mathsf{sk}) = ((\hat{X}_1, \hat{X}_2, \hat{X}_3, \hat{X}_4, \hat{X}_5), (x_1, x_2, x_3, x_4, x_5))$ for a mercurial signature scheme MS_f on messages of length $\ell = 5$. $\mathcal{B}$ chooses uniformly at random a secret point $\hat{x} \leftarrow \mathbb{Z}_p^*$ but does not know the secret seeds y_1, y_2. He also picks $x_6 \leftarrow \mathbb{Z}_p^*$ and sets $x_7 = x_6 \cdot \hat{x}$. He then sets $\mathsf{pk}_{\mathsf{X}} = (\mathsf{pk}, \hat{X}_6, \hat{X}_7, \hat{A}_0, \hat{A}_1, \hat{A}_2)$, where $\hat{X}_i = \hat{P}^{x_i}$. $\mathcal{B}$ chooses a random function $R : \mathbb{Z}_p^* \to \mathbb{Z}_p^*$ and forwards PP_{X} and pk_{X} to $\mathcal{A}$.

$\mathcal{A}$ proceeds to make queries to the signing oracle. Acting as the challenger for $\mathcal{A}$, $\mathcal{B}$ is responsible for computing the responses to the signature queries and forwarding them to $\mathcal{A}$. $\mathcal{B}$ responds to the signature queries as follows.

Let $\mathcal{A}$'s j^{th} signature query be on message $m_j = (\hat{g}_j, u_{j,1}, \ldots, u_{j,n})$. $\mathcal{B}$ acts as the extractor while $\mathcal{A}$ gives a ZKPoK that, for all $1 \leq i \leq n$, he knows $m_{j,i}$ such that $u_{j,i} = \hat{g}_j^{m_{j,i}}$. $\mathcal{B}$ extracts the $m_{j,i}$'s, or if the extraction fails, $\mathcal{B}$ denies $\mathcal{A}$ the signature. Otherwise, $\mathcal{B}$ computes the polynomial $p_j(x) = m_{j,1} + m_{j,2}x + \cdots + m_{j,n}x^{n-1}$ and evaluates $q_j = p_j(\hat{x})$.

(1) If $j \leq i$, $\mathcal{B}$ computes $R(q_j)$ and $\hat{h}_j = \hat{g}_j^{R(q_j)}$. $\mathcal{B}$ picks uniformly at random $w_j \leftarrow \mathbb{Z}_p^*$ and computes $\tilde{g}_j = \hat{g}_j^{w_j}, \tilde{h}_j = \hat{h}_j^{w_j}$, and $\tilde{u}_{j,i} = u_{j,i}^{w_j}$ $\forall i$. He also computes $\tilde{g}_j^2, \ldots, \tilde{g}_j^n$. He then signs n messages of the form $M_{j,i} = (\tilde{g}_j, \tilde{g}_j^i, \tilde{g}_j^n, \tilde{h}_j, \tilde{u}_{j,i})$ using his secret key sk for MS_f and sends $\tilde{m}_j = (\tilde{g}_j, \tilde{u}_{j,1}, \ldots, \tilde{u}_{j,n})$ and $(\tilde{h}_j, \sigma_j = \{\sigma_{j,1}, \ldots, \sigma_{j,n}\})$ to $\mathcal{A}$, along with a simulated ZKPoK that $\tilde{h}_j$ was computed correctly.

(2) If $j = i+1$, $\mathcal{B}$ computes $\hat{h}_j = D^{q_j}$. $\mathcal{B}$ sets $\tilde{g}_j = B_2, \tilde{h}_j = \hat{h}_j$, and $\tilde{u}_{j,i} = B_2^{m_{j,i}}$ $\forall i$. He also computes $B_2^2, \ldots, B_2^n$. He then signs n messages of the form $M_{j,i} = (B_2, B_2^i, B_2^n, D^{q_j}, B_2^{m_{j,i}})$ using his secret key sk for MS_f and sends $\tilde{m}_j = (B_2, B_2^{m_{j,1}}, \ldots, B_2^{m_{j,n}})$ and $(D^{q_j}, \sigma_j = \{\sigma_{j,1}, \ldots, \sigma_{j,n}\})$ to $\mathcal{A}$, along with a simulated ZKPoK that $\tilde{h}_j$ was computed correctly.

(3) If $j > i+1$, $\mathcal{B}$ computes: $\hat{h}_j = C^{q_j}$. $\mathcal{B}$ picks uniformly at random $w_j \leftarrow \mathbb{Z}_p^*$ and computes $\tilde{g}_j = B_1^{w_j}, \tilde{h}_j = \hat{h}_j^{w_j}$, and $\tilde{u}_{j,i} = (B_1^{w_j})^{m_{j,i}}$ $\forall i$. He also computes $(B_1^{w_j})^2, \ldots, (B_1^{w_j})^n$. He then signs n messages of the form $M_{j,i} = (B_1^{w_j}, (B_1^{w_j})^i, (B_1^{w_j})^n, C^{q_j \cdot w_j}, (B_1^{w_j})^{m_{j,i}})$ using his secret key sk for MS_f and sends $\tilde{m}_j = (B_1^{w_j}, (B_1^{w_j})^{m_{j,1}}, \ldots, (B_1^{w_j})^{m_{j,n}})$ and $(C^{q_j \cdot w_j}, \sigma_j = \{\sigma_{j,1}, \ldots, \sigma_{j,n}\})$ to $\mathcal{A}$, along with a simulated ZKPoK that $\tilde{h}_j$ was computed correctly.

Finally, when $\mathcal{A}$ terminates, without loss of generality he outputs either 0 or 1. He outputs 0 if he thinks he has observed $\mathcal{H}_i$ and 1 if he thinks he has observed $\mathcal{H}_{i+1}$. If $\mathcal{A}$ outputs 0, $\mathcal{B}$ outputs 0; otherwise, $\mathcal{B}$ outputs 1. Let us analyze $\mathcal{B}$'s success probability.

First, note that in the public key $\mathsf{pk_X}$, the values $\hat{X}_8, \hat{X}_9, \hat{X}_{10}$ can't be computed as $\hat{P}^{x_8}, \hat{P}^{x_9}, \hat{P}^{x_{10}}$, where $x_9 = x_8 \cdot y_1$ and $x_{10} = x_8 \cdot y_2$, because $\mathcal{B}$ does not know y_1 or y_2; however, $(\hat{A}_0, \hat{A}_1, \hat{A}_2)$ is implicitly $(\hat{P}^{\alpha}, \hat{P}^{\alpha u}, \hat{P}^{\alpha v})$, which is distributed the same as $(\hat{P}^{x_8}, \hat{P}^{x_8 \cdot y_1}, \hat{P}^{x_8 \cdot y_2})$ for uniformly random $x_8, y_1, y_2 \in \mathbb{Z}_p^*$. Thus, $\mathsf{pk_X}$ is distributed correctly.

The case $j \leq i$ is exactly as in the hybrid game. For the case $j > i+1$, B_1 is implicitly P^{β}, so $\tilde{g}_j = P^{\beta w_j}$, which is distributed the same as $\hat{g}_j^{w_j}$ because w_j is uniformly random in $\mathbb{Z}_p^*$. C is implicitly $P^{\beta uv}$, so $\tilde{h}_j = C^{q_j \cdot w_j} = (P^{\beta w_j})^{uv \cdot q_j} = (B_1^{w_j})^{uv \cdot q_j} = \tilde{g}_j^{uv \cdot q_j}$, which is distributed the same as $\tilde{g}_j^{y_1 y_2 \cdot q_j} = \tilde{g}_j^{y \cdot q_j}$ for uniformly random $y_1, y_2 \in \mathbb{Z}_p^*$. For the case $j = i+1$, B_2 is implicitly P^{ω}, so $\tilde{g}_j = P^{\omega}$, which is distributed the same as $\hat{g}_j^{w_j}$ for a uniformly random $w_j \in \mathbb{Z}_p^*$. D is implicitly $P^{\omega uv}$ or P^r for the uniformly random $u, v, \omega, r \in \mathbb{Z}_p^*$ given as input to the reduction. If $D = P^{\omega uv}$, then $\tilde{h}_j = D^{q_j} = B_2^{uv \cdot q_j}$, which is distributed the same as $\tilde{g}_j^{y_1 y_2 \cdot q_j} = \tilde{g}_j^{y \cdot q_j}$ for uniformly random $y_1, y_2 \in \mathbb{Z}_p^*$. If $D = P^r$, then $\tilde{h}_j = D^{q_j} = P^{r \cdot q_j}$, which is distributed the same as $\tilde{g}_j^{R(q_j)}$ since the r given as input to the reduction is uniformly random in $\mathbb{Z}_p^*$. Thus, $D = P^{\omega uv}$ corresponds to hybrid $\mathcal{H}_i$ and $D = P^r$ corresponds to hybrid $\mathcal{H}_{i+1}$.

The above description of $\mathcal{B}$'s responses to $\mathcal{A}$'s oracle queries demonstrates that $\mathcal{B}$ is able to emulate the appropriate hybrid and compute each step of $\mathcal{A}$'s oracle queries exactly as $\mathcal{A}$'s challenger in the game would. If $\mathcal{A}$ outputs 0, it means the input looks like it came from $\mathcal{H}_i$, so $\mathcal{B}$ outputs 0 to indicate the distribution D_0. If $\mathcal{A}$ outputs 1, it means the input looks like it came from $\mathcal{H}_{i+1}$, so $\mathcal{B}$ outputs 1 to indicate the distribution D_1. Then, $\mathcal{A}$'s advantage translates into $\mathcal{B}$'s advantage: if $\mathcal{A}$ is able to distinguish $\mathcal{H}_i$ from $\mathcal{H}_{i+1}$ with non-negligible probability ϵ, then $\mathcal{B}$ is able to distinguish ABDDH† tuples with the same non-negligible probability.

Game 4. In this game, the glue element is $\tilde{h} = \tilde{g}^{R(\dot{q})}$, where $\dot{q} = p(\alpha)$ for a "fake" secret point $\alpha \in \mathbb{Z}_p^*$ and $R : \mathbb{Z}_p^* \to \mathbb{Z}_p^*$ is a random function.

Game 4 is the same as Game 3, except in addition to the secret point $\hat{x}$, the challenger $\mathcal{C}$ also chooses a "fake" secret point uniformly at random $\alpha \leftarrow \mathbb{Z}_p^*$ and computes $\hat{h} = \hat{g}^{R(\dot{q})}$, where $\dot{q} = p(\alpha)$. The rest of the signing protocol is carried out as in Game 3.

Claim 4. *A PPT adversary can distinguish Game 3 from Game 4 only if a collision $p_i(\hat{x}) = p_j(\hat{x})$ occurs in Game 3 with non-negligible probability.*

Let a PPT adversary $\mathcal{A}$'s j^{th} signature query be on message $m_j = (\hat{g}_j, u_{j,1}, \ldots, u_{j,n})$, where $u_{j,i} = \hat{g}_j^{m_{j,i}}$ $\forall i$. In Game 3, the challenger $\mathcal{C}$ extracts the $m_{j,i}$'s, forms the polynomial $p_j(x) = m_{j,1} + m_{j,2}x + \cdots + m_{j,n}x^{n-1}$, and evaluates $q_j = p_j(\hat{x})$. He then computes $R(q_j)$ for some random function $R : \mathbb{Z}_p^* \to \mathbb{Z}_p^*$ and forms the glue element as $\tilde{h}_j = \tilde{g}_j^{R(q_j)}$.

In Game 4, the challenger $\mathcal{C}$ extracts the $m_{j,i}$'s, forms the polynomial $p_j(x) = m_{j,1} + m_{j,2}x + \cdots + m_{j,n}x^{n-1}$, and evaluates $\dot{q}_j = p_j(\alpha)$ at the "fake" secret point $\alpha \in \mathbb{Z}_p^*$. He then computes $R(\dot{q}_j)$ and forms the glue element as $\tilde{h}_j = \tilde{g}_j^{R(\dot{q}_j)}$.

The only difference between the two games is that in Game 4, the polynomials $p_j(x)$ are evaluated at α, which is independent of the true secret point $\hat{x}$. If $\dot{q}_i = \dot{q}_j$ for some $p_i(x) \neq p_j(x)$, then $R(\dot{q}_i) = R(\dot{q}_j)$, so $\mathcal{A}$ learns that $p_i(\alpha) = p_j(\alpha)$. The value α is independent of the adversary's view unless such a collision occurs. We will show that a collision occurs with negligible probability by induction on the number of queries.

For the base case, suppose $\dot{q}_1 = \dot{q}_2$. Then, α is a root of the difference polynomial $p_1(x) - p_2(x)$. $\mathcal{A}$'s probability of successfully constructing a difference polynomial with root α is maximized by choosing $n-1$ distinct roots for it. The probability that one of these $n-1$ distinct roots is α is $(n-1)/p$. Thus, the probability that $\dot{q}_1 = \dot{q}_2$ is at most $(n-1)/p$, which is negligible. For the induction step, suppose $\forall i \leq t, \forall j \leq t, \dot{q}_i \neq \dot{q}_j$. The probability that $\dot{q}_{t+1}$ collides with one of the first t $\dot{q}_i$'s, conditioned on the fact that there are no collisions among the first t $\dot{q}_i$'s, is at most $(t+1)(n-1)/p$, which is negligible, completing the induction step.

Thus, $\mathcal{A}$ can distinguish Game 4 from Game 3 only if a collision $p_i(\hat{x}) = p_j(\hat{x})$ occurs in Game 3 with non-negligible probability. We now show that such a collision occurs in Game 3 with negligible probability or the DL assumption doesn't hold.

Claim 5. *A collision $p_i(\hat{x}) = p_j(\hat{x})$ occurs in Game 3 with negligible probability under the DL assumption in $\mathbb{G}_2$.*

We wish to show that if there exists a PPT adversary $\mathcal{A}$ that produces a collision $p_i(\hat{x}) = p_j(\hat{x})$ for some polynomials $p_i(x) \neq p_j(x)$ with non-negligible probability, then we can construct a PPT adversary $\mathcal{A}'$ that breaks the DL assumption.

Suppose there exists such a PPT algorithm $\mathcal{A}$. Then, we construct a PPT adversary $\mathcal{A}'$ as a reduction $\mathcal{B}$ running $\mathcal{A}$ as a subroutine. We construct the reduction $\mathcal{B}$ for breaking the DL assumption as follows.

$\mathcal{B}$ receives as input $(\hat{A}, \hat{B}) \in \mathbb{G}_2^*$, where implicitly $\hat{B} = \hat{A}^{\hat{x}}$ for some uniformly random $\hat{x} \in \mathbb{Z}_p^*$. (Note that this variant of the DL assumption is equivalent to the one in which $\hat{x}$ is drawn from $\mathbb{Z}_p$.)

$\mathcal{B}$ computes the public parameters PP and keys $(\mathsf{pk}, \mathsf{sk}) = ((\hat{X}_1, \hat{X}_2, \hat{X}_3, \hat{X}_4, \hat{X}_5), (x_1, x_2, x_3, x_4, x_5))$ for a mercurial signature scheme MS_f on messages of length $\ell = 5$. $\mathcal{B}$ chooses uniformly at random secret values $y_1, y_2 \leftarrow \mathbb{Z}_p^*$ but does not know the secret point $\hat{x}$. He also picks $x_8 \leftarrow \mathbb{Z}_p^*$ and sets $x_9 = x_8 \cdot y_1$ and $x_{10} = x_8 \cdot y_2$. He then sets $\mathsf{pk}_\mathsf{X} = (\mathsf{pk}, \hat{A}, \hat{B}, \hat{X}_8, \hat{X}_9, \hat{X}_{10})$, where $\hat{X}_i = \hat{P}^{x_i}$, and forwards PP_X and pk_X to $\mathcal{A}$.

$\mathcal{A}$ proceeds to make queries to the signing oracle. Acting as the challenger for $\mathcal{A}$, $\mathcal{B}$ is responsible for computing the responses to the signature queries and forwarding them to $\mathcal{A}$. $\mathcal{B}$ responds to the signature queries as follows.

Let $\mathcal{A}$'s j^{th} signature query be on message $m_j = (\hat{g}_j, u_{j,1}, \ldots, u_{j,n})$. $\mathcal{B}$ acts as the extractor while $\mathcal{A}$ gives a ZKPoK that, for all $1 \leq i \leq n$, he knows $m_{j,i}$ such that $u_{j,i} = \hat{g}_j^{m_{j,i}}$. $\mathcal{B}$ extracts the $m_{j,i}$'s, or if the extraction fails, $\mathcal{B}$ denies $\mathcal{A}$ the signature. Otherwise, $\mathcal{B}$ computes the polynomial $p_j(x) = m_{j,1} + m_{j,2}x + \cdots + m_{j,n}x^{n-1}$.

For all $1 \leq t < j$, $\mathcal{B}$ computes the difference polynomial $p_j(x) - p_t(x)$ and finds its $n-1$ roots $r_{t,1}, \ldots, r_{t,n-1}$. Since $\mathcal{B}$ knows $\hat{A}$, he can compute $\hat{A}^{r_{t,i}}$ $\forall t, \forall i$ and check if $\hat{A}^{r_{t,i}} = \hat{B}$. If this holds for some $r_{t,i}$, then $r_{t,i} = \hat{x}$ and $\mathcal{B}$ wins the DL game. If this does not hold, $\mathcal{B}$ picks uniformly at random $\tilde{R}_j \leftarrow \mathbb{Z}_p^*$ and computes $\hat{h}_j = \hat{g}_j^{\tilde{R}_j}$ since he cannot correctly compute $\hat{g}^{R(q_j)}$; however, note that $\mathcal{A}$'s view is identical because he receives random values. $\mathcal{B}$ picks uniformly at random $w_j \leftarrow \mathbb{Z}_p^*$ and computes $\tilde{g}_j = \hat{g}_j^{w_j}, \tilde{h}_j = \hat{h}_j^{w_j}$, and $\tilde{u}_{j,i} = u_{j,i}^{w_j}$ $\forall i$. He also computes $\tilde{g}_j^2, \ldots, \tilde{g}_j^n$. He then signs n messages of the form $M_{j,i} = (\tilde{g}_j, \tilde{g}_j^i, \tilde{g}_j^n, \tilde{h}_j, \tilde{u}_{j,i})$ using his secret key sk for MS_f and sends $\tilde{m}_j = (\tilde{g}_j, \tilde{u}_{j,1}, \ldots, \tilde{u}_{j,n})$ and $(\tilde{h}_j, \sigma_j = \{\sigma_{j,1}, \ldots, \sigma_{j,n}\})$ to $\mathcal{A}$, along with a simulated ZKPoK that $\tilde{h}_j$ was computed correctly. $\mathcal{A}$ issues queries for signatures on messages a polynomial number of times. $\mathcal{A}$'s success in producing a difference polynomial $p_j(x) - p_t(x)$ with root $\hat{x}$ with non-negligible probability translates into $\mathcal{B}$'s success in breaking the DL assumption.

From Claim 4 and Claim 5, we can conclude that a PPT adversary $\mathcal{A}$ cannot distinguish Game 3 from Game 4, except with negligible probability.

Game 5. In this game, the glue element is $\tilde{h} = \tilde{g}^{R(m_1,\ldots,m_n)}$, where $R : (\mathbb{Z}_p^*)^n \to \mathbb{Z}_p^*$ is a random function.

Game 5 is the same as Game 4, except the challenger $\mathcal{C}$ does not choose a "fake" secret point $\alpha \in \mathbb{Z}_p^*$ and does not compute or evaluate the polynomial $p(x)$. Instead, $\mathcal{C}$ chooses a random function $R : (\mathbb{Z}_p^*)^n \to \mathbb{Z}_p^*$ and for each signature computes $\hat{h} = \hat{g}^{R(m_1,\ldots,m_n)}$ The rest of the signing protocol is carried out as in Game 4.

Claim 6. *An adversary's view in Game 4 is the same as it is in Game 5, except with negligible probability.*

Let a (possibly unbounded) adversary $\mathcal{A}$'s j^{th} signature query be on message $m_j = (\hat{g}_j, u_{j,1}, \ldots, u_{j,n})$, where $u_{j,i} = \hat{g}_j^{m_{j,i}}$ $\forall i$. In Game 5, the challenger $\mathcal{C}$ extracts the $m_{j,i}$'s, computes $R(m_{j,1}, \ldots, m_{j,n})$ for some random function R, and forms the glue element as $\tilde{h}_j = \tilde{g}_j^{R(m_{j,1},\ldots,m_{j,n})}$, where $\tilde{g}_j = \hat{g}_j^{w_j}$ for some uniformly random $w_j \leftarrow \mathbb{Z}_p^*$.

In Game 4, the challenger $\mathcal{C}$ extracts the $m_{j,i}$'s, forms the polynomial $p_j(x) = m_{j,1} + m_{j,2}x + \cdots + m_{j,n}x^{n-1}$, and evaluates $\dot{q}_j = p_j(\alpha)$ at the "fake" secret point $\alpha \in \mathbb{Z}_p^*$. He then computes $R(\dot{q}_j)$ for some random function R and forms the glue element as $\tilde{h}_j = \tilde{g}_j^{R(\dot{q}_j)}$, where $\tilde{g}_j = \hat{g}_j^{w_j}$ for some uniformly random $w_j \leftarrow \mathbb{Z}_p^*$.

If $\dot{q}_i = \dot{q}_j$ for some $p_i(x) \neq p_j(x)$, then $R(\dot{q}_i) = R(\dot{q}_j)$, so $\mathcal{A}$ learns that $p_i(\alpha) = p_j(\alpha)$. The value α is independent of the adversary's view unless such a collision occurs. We showed in Claim 4 that a collision $p_i(\alpha) = p_j(\alpha)$ occurs in Game 4 with negligible probability. If there are no such collisions, $\mathcal{A}$'s view is identical in both games because he receives random values.

This completes the proof of unforgeability for $\mathsf{MS_X}$ (Theorem 4). $\square$

4.4 Class-hiding

Message class-hiding states that given two messages m_1 and m_2, it is hard to tell if $m_2 \in [m_1]_{\mathcal{R}_m}$. Public key class-hiding states that given two public keys $\mathsf{pk}_{\mathsf{X},1}$ and $\mathsf{pk}_{\mathsf{X},2}$ and oracle access to the signing algorithm for both of them, it is hard to tell if $\mathsf{pk}_{\mathsf{X},2} \in [\mathsf{pk}_{\mathsf{X},1}]_{\mathcal{R}_{\mathsf{pk}}}$.

Theorem 5 (Message class-hiding). *Let MS_f be a mercurial signature scheme on message space $(\mathbb{G}_1^*)^5$ as in Theorem 1, and let $\mathsf{MS_X}$ be the variable-length mercurial signature scheme on message space $(\mathbb{G}_1^*)^{n+1}$ constructed above. Then, message class-hiding of $\mathsf{MS_X}$ holds under the decisional Diffie-Hellman assumption (DDH) in $\mathbb{G}_1$. The same holds when $\mathbb{G}_1$ and $\mathbb{G}_2$ are swapped.*

The proof is a straightforward hybrid argument inherited from Fuchsbauer et al. [22].

Theorem 6 (Public key class-hiding). *Let MS_f be a mercurial signature scheme on message space $(\mathbb{G}_1^*)^5$, and let $\mathsf{MS_X}$ be the variable-length mercurial signature scheme on message space $(\mathbb{G}_1^*)^{n+1}$ constructed above. Suppose all signatures are issued via the interactive signing protocol described in Section 4.1, where the proof system used in Step 1 is extractable under sequential (or concurrent) composition. Then, public key class-hiding of $\mathsf{MS_X}$ holds sequentially (or concurrently) under the DL assumption in $\mathbb{G}_2$, the ABDDH$^+$ assumption in $\mathbb{G}_1$, and the DDH assumption in $\mathbb{G}_1$ and $\mathbb{G}_2$. The same holds when $\mathbb{G}_1$ and $\mathbb{G}_2$ are swapped.*

For the proof, consider two public keys for $\mathsf{MS_X}$:

$$\mathsf{pk}_{\mathsf{X},1} = (\mathsf{pk}_1, \hat{P}^{x_{1,6}}, \hat{P}^{x_{1,6}\cdot\hat{x}_1}, \hat{P}^{x_{1,8}}, \hat{P}^{x_{1,8}\cdot y_1^{(1)}}, \hat{P}^{x_{1,8}\cdot y_2^{(1)}})$$
$$\mathsf{pk}_{\mathsf{X},2} = (\mathsf{pk}_2, \hat{P}^{x_{2,6}}, \hat{P}^{x_{2,6}\cdot\hat{x}_2}, \hat{P}^{x_{2,8}}, \hat{P}^{x_{2,8}\cdot y_1^{(2)}}, \hat{P}^{x_{2,8}\cdot y_2^{(2)}})$$

where $x_{\delta,6}, x_{\delta,8}, \hat{x}_\delta, y_1^{(\delta)}, y_2^{(\delta)} \in \mathbb{Z}_p^*$ for $\delta \in \{1,2\}$. They are independent if these values are sampled uniformly at random from $\mathbb{Z}_p^*$ and equivalent if $\mathsf{pk}_{\mathsf{X},2} = \mathsf{pk}_{\mathsf{X},1}^\beta$ for some $\beta \in \mathbb{Z}_p^*$. They are said to be $1/2$ independent and $1/2$ equivalent if $\mathsf{pk}_2 = \mathsf{pk}_1^\beta$, but the remaining elements are independent.

We construct a sequence of games beginning with the real signing game in which $\mathsf{pk}_{\mathsf{X},1}, \mathsf{pk}_{\mathsf{X},2}$ are independent (Game 0). In the real signing game, a signature query on a message m under chosen public key $\mathsf{pk}_{\mathsf{X},\delta}$ for $\delta \in \{1,2\}$ results in a glue element computed as $\tilde{h} = \tilde{g}^{y^{(\delta)}\cdot q_\delta}$, where $q_\delta = p(\hat{x}_\delta)$ and $y^{(\delta)} := y_1^{(\delta)} \cdot y_2^{(\delta)}$. The sequence of games ends with the real signing game in which $\mathsf{pk}_{\mathsf{X},1}, \mathsf{pk}_{\mathsf{X},2}$ are equivalent (Game 13). We show that Game 0 and Game 13 are indistinguishable via a sequence of intermediate games. These games cycle through public keys $\mathsf{pk}_{\mathsf{X},1}, \mathsf{pk}_{\mathsf{X},2}$ that are independent, $1/2$ independent and $1/2$ equivalent, and equivalent, as well as glue elements that are computed in the various ways specified in the proof of unforgeability. Since the real signing game in which $\mathsf{pk}_{\mathsf{X},1}, \mathsf{pk}_{\mathsf{X},2}$ are independent (Game 0) is indistinguishable from the real signing game in which $\mathsf{pk}_{\mathsf{X},1}, \mathsf{pk}_{\mathsf{X},2}$ are equivalent (Game 13), $\mathsf{MS_X}$ satisfies public key class-hiding. The proof can be found in Appendix C.

Acknowledgements

The first author was supported by EPSRC Grant EP/N028104/1 and NSF Grant 1422361. The second author was supported by NSF Grant 1422361.

References

[1] M. Backes, L. Hanzlik, K. Kluczniak, and J. Schneider. Signatures with flexible public key: Introducing equivalence classes for public keys. In T. Peyrin and S. D. Galbraith, editors, *ASIACRYPT 2018, Brisbane, QLD, Australia, December 2-6, 2018, Part II*, volume 11273 of *LNCS*, pages 405–434. Springer, 2018.

[2] M. Belenkiy, J. Camenisch, M. Chase, M. Kohlweiss, A. Lysyanskaya, and H. Shacham. Randomizable proofs and delegatable anonymous credentials. In S. Halevi, editor, *CRYPTO 2009, Santa Barbara, CA, USA, August 16-20, 2009*, volume 5677, pages 108–125. Springer, 2009.

[3] D. Bernhard, M. Fischlin, and B. Warinschi. Adaptive proofs of knowledge in the random oracle model. In J. Katz, editor, *PKC 2015, Gaithersburg, MD, USA, March 30 - April 1, 2015*, volume 9020 of *LNCS*, pages 629–649. Springer, 2015.

[4] E. F. Brickell, J. Camenisch, and L. Chen. Direct anonymous attestation. In V. Atluri, B. Pfitzmann, and P. D. McDaniel, editors, *CCS 2004, Washington, DC, USA, October 25-29, 2004*, pages 132–145. ACM, 2004.

[5] J. Camenisch, M. Dubovitskaya, K. Haralambiev, and M. Kohlweiss. Composable and modular anonymous credentials: Definitions and practical constructions. In T. Iwata and J. H. Cheon, editors, *ASIACRYPT 2015, Auckland, New Zealand, November 29 - December 3, 2015, Part II*, volume 9453 of *LNCS*, pages 262–288. Springer, 2015.

[6] J. Camenisch, S. Krenn, A. Lehmann, G. L. Mikkelsen, G. Neven, and M. Ø. Pedersen. Formal treatment of privacy-enhancing credential systems. *IACR Cryptol. ePrint Arch.*, 2014:708, 2014.

[7] J. Camenisch and A. Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In B. Pfitzmann, editor, *EUROCRYPT 2001, Innsbruck, Austria, May 6-10, 2001, Proceeding*, volume 2045 of *LNCS*, pages 93–118. Springer, 2001.

[8] J. Camenisch and A. Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In M. K. Franklin, editor, *CRYPTO 2004, Santa Barbara, California, USA, August 15-19, 2004*, volume 3152 of *LNCS*, pages 56–72. Springer, 2004.

[9] J. Camenisch and M. Michels. Proving in zero-knowledge that a number is the product of two safe primes. In J. Stern, editor, *Advances in Cryptology - EUROCRYPT 1999, Prague, Czech Republic, May 2-6, 1999*, volume 1592 of *LNCS*, pages 107–122. Springer, 1999.

[10] J. Camenisch and V. Shoup. Practical verifiable encryption and decryption of discrete logarithms. In D. Boneh, editor, *CRYPTO 2003, Santa Barbara, California, USA, August 17-21, 2003*, volume 2729 of *LNCS*, pages 126–144. Springer, 2003.

[11] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA*, pages 136–145. IEEE Computer Society, 2001.

[12] M. Chase, M. Kohlweiss, A. Lysyanskaya, and S. Meiklejohn. Malleable signatures: Complex unary transformations and delegatable anonymous credentials. *IACR Cryptol. ePrint Arch.*, 2013:179, 2013.

[13] M. Chase and A. Lysyanskaya. On signatures of knowledge. In C. Dwork, editor, *CRYPTO 2006, California, USA, August 20-24, 2006*, volume 4117 of *LNCS*, pages 78–96. Springer, 2006.

[14] D. Chaum. Showing credentials without identification: Signatures transferred between unconditionally unlinkable pseudonyms. In F. Pichler, editor, *EUROCRYPT '85, Linz, Austria, April 1985*, volume 219 of *LNCS*, pages 241–244. Springer, 1985.

[15] E. C. Crites and A. Lysyanskaya. Delegatable anonymous credentials from mercurial signatures. In M. Matsui, editor, *CT-RSA 2019, San Francisco, CA, USA, March 4-8, 2019*, volume 11405 of *LNCS*, pages 535–555. Springer, 2019.

[16] I. Damgård. On sigma-protocols, 2002.

[17] Y. Dodis, V. Shoup, and S. Walfish. Efficient constructions of composable commitments and zero-knowledge proofs. In D. A. Wagner, editor, *CRYPTO 2008, Santa Barbara, CA, USA, August 17-21, 2008.*, volume 5157 of *LNCS*, pages 515–535. Springer, 2008.

[18] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In A. M. Odlyzko, editor, *Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986*, volume 263 of *LNCS*, pages 186–194. Springer, 1986.

[19] M. Fischlin. Communication-efficient non-interactive proofs of knowledge with online extractors. In V. Shoup, editor, *CRYPTO 2005, Santa Barbara, California, USA, August 14-18, 2005*, volume 3621 of *LNCS*, pages 152–168. Springer, 2005.

[20] G. Fuchsbauer and R. Gay. Weakly secure equivalence-class signatures from standard assumptions. In M. Abdalla and R. Dahab, editors, *PKC 2018, Rio de Janeiro, Brazil, March 25-29, 2018, Part II*, volume 10770 of *LNCS*, pages 153–183. Springer, 2018.

[21] G. Fuchsbauer, C. Hanser, C. Kamath, and D. Slamanig. Practical round-optimal blind signatures in the standard model from weaker assumptions. In V. Zikas and R. D. Prisco, editors, *SCN 2016, Amalfi, Italy, August 31 - September 2, 2016*, volume 9841 of *LNCS*, pages 391–408. Springer, 2016.

[22] G. Fuchsbauer, C. Hanser, and D. Slamanig. Structure-preserving signatures on equivalence classes and constant-size anonymous credentials. *J. Cryptol.*, 32(2):498–546, 2019.

[23] J. Groth and A. Sahai. Efficient non-interactive proof systems for bilinear groups. In N. P. Smart, editor, *Advances in Cryptology - EUROCRYPT 2008, Istanbul, Turkey, April 13-17, 2008*, volume 4965 of *LNCS*, pages 415–432. Springer, 2008.

[24] A. Lysyanskaya. *Signature schemes and applications to cryptographic protocol design*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 2002.

[25] A. Lysyanskaya, R. L. Rivest, A. Sahai, and S. Wolf. Pseudonym systems. In H. M. Heys and C. M. Adams, editors, *SAC 1999, Kingston, Ontario, Canada, August 9-10, 1999*, volume 1758 of *LNCS*, pages 184–199. Springer, 1999.

[26] S. Meiklejohn, C. C. Erway, A. Küpçü, T. Hinkle, and A. Lysyanskaya. ZKPDL: A language-based system for efficient zero-knowledge proofs and electronic cash. In *19th USENIX Security Symposium, Washington, DC, USA, August 11-13, 2010*, pages 193–206. USENIX Association, 2010.

A Prior Construction [15]

The prior construction of mercurial signatures is as follows [15]. The message space consists of vectors of group elements from $\mathbb{G}_1^*$, the space of secret keys consists of vectors of elements from $\mathbb{Z}_p^*$, and the space of public keys consists of vectors of group elements from $\mathbb{G}_2^*$. Once the prime p, $\mathbb{G}_1$, $\mathbb{G}_2$, and a fixed length parameter ℓ are well defined, the equivalence relations are as follows:

$$\mathcal{R}_M = \{(M, M') \in (\mathbb{G}_1^*)^\ell \times (\mathbb{G}_1^*)^\ell \mid \exists\, \mu \in \mathbb{Z}_p^* \text{ s.t. } M' = M^\mu\}$$
$$\mathcal{R}_{\mathsf{sk}} = \{(\mathsf{sk}, \tilde{\mathsf{sk}}) \in (\mathbb{Z}_p^*)^\ell \times (\mathbb{Z}_p^*)^\ell \mid \exists\, \rho \in \mathbb{Z}_p^* \text{ s.t. } \tilde{\mathsf{sk}} = \rho \cdot \mathsf{sk}\}$$
$$\mathcal{R}_{\mathsf{pk}} = \{(\mathsf{pk}, \tilde{\mathsf{pk}}) \in (\mathbb{G}_2^*)^\ell \times (\mathbb{G}_2^*)^\ell \mid \exists\, \rho \in \mathbb{Z}_p^* \text{ s.t. } \tilde{\mathsf{pk}} = \mathsf{pk}^\rho\}$$

The message space for this mercurial signature scheme is $(\mathbb{G}_1^*)^\ell$, but a mercurial signature scheme with message space $(\mathbb{G}_2^*)^\ell$ can be obtained by simply switching $\mathbb{G}_1^*$ and $\mathbb{G}_2^*$ throughout. The algorithms are as follows:

$\mathsf{PPGen}(1^k) \to PP$: Compute $\mathsf{BG} \leftarrow \mathsf{BGGen}(1^k)$. Output $PP = \mathsf{BG} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, P, \hat{P}, e)$.

$\mathsf{KeyGen}(PP, \ell) \to (\mathsf{pk}, \mathsf{sk})$: For $1 \le i \le \ell$, pick $x_i \leftarrow \mathbb{Z}_p^*$, set $\mathsf{sk} = (x_1, \dots, x_\ell)$, $\mathsf{pk} = (\hat{X}_1, \dots, \hat{X}_\ell)$, where $\hat{X}_i = \hat{P}^{x_i}$ for $1 \le i \le \ell$. Output $(\mathsf{pk}, \mathsf{sk})$.

$\mathsf{Sign}(\mathsf{sk}, M) \to \sigma$: On input $\mathsf{sk} = (x_1, \dots, x_\ell)$, $M = (M_1, \dots, M_\ell) \in (\mathbb{G}_1^*)^\ell$, sample $y \leftarrow \mathbb{Z}_p^*$, output $\sigma = (Z, Y, \hat{Y})$, where $Z = \left(\prod_{i=1}^\ell M_i^{x_i}\right)^y, Y = P^{\frac{1}{y}}$, and $\hat{Y} = \hat{P}^{\frac{1}{y}}$.

$\mathsf{Verify}(\mathsf{pk}, M, \sigma) \to 0/1$: On input $\mathsf{pk} = (\hat{X}_1, \dots, \hat{X}_\ell)$, $M = (M_1, \dots, M_\ell)$, and $\sigma = (Z, Y, \hat{Y})$, check $\prod_{i=1}^\ell e(M_i, \hat{X}_i) = e(Z, \hat{Y}) \wedge e(Y, \hat{P}) = e(P, \hat{Y})$. If this holds, output 1; otherwise, output 0.

$\mathsf{ConvertSK}(\mathsf{sk}, \rho) \to \tilde{\mathsf{sk}}$: On input $\mathsf{sk} = (x_1, \dots, x_\ell)$ and key converter $\rho \in \mathbb{Z}_p^*$, output new $\tilde{\mathsf{sk}} = \rho \cdot \mathsf{sk}$.

$\mathsf{ConvertPK}(\mathsf{pk}, \rho) \to \tilde{\mathsf{pk}}$: On input $\mathsf{pk} = (\hat{X}_1, \dots, \hat{X}_\ell)$ and key converter $\rho \in \mathbb{Z}_p^*$, output new $\tilde{\mathsf{pk}} = \mathsf{pk}^\rho$.

$\mathsf{ConvertSig}(\mathsf{pk}, M, \sigma, \rho) \to \tilde{\sigma}$: On input pk, M, $\sigma = (Z, Y, \hat{Y})$, key converter $\rho \in \mathbb{Z}_p^*$, sample $\psi \leftarrow \mathbb{Z}_p^*$. Output $\tilde{\sigma} = (Z^{\psi\rho}, Y^{\frac{1}{\psi}}, \hat{Y}^{\frac{1}{\psi}})$.

$\mathsf{ChangeRep}(\mathsf{pk}, M, \sigma, \mu) \to (M', \sigma')$: On input pk, M, $\sigma = (Z, Y, \hat{Y})$, $\mu \in \mathbb{Z}_p^*$, sample $\psi \leftarrow \mathbb{Z}_p^*$. Compute $M' = M^\mu$, $\sigma' = (Z^{\psi\mu}, Y^{\frac{1}{\psi}}, \hat{Y}^{\frac{1}{\psi}})$. Output (M', σ').

B Zero-Knowledge Proofs

Let us now address which zero-knowledge proof of knowledge (ZKPoK) protocol ought to be used in the signing protocol (Section 4.1) . There is a rich literature on ZKPoK protocols for discrete logarithm-based relations that are both practical and provably secure. For our purposes, a ZKPoK protocol needs to be secure under the appropriate notion of composition: our unforgeability game allows the adversary to issue many signing queries, so the challenger must be able to respond to many queries. The best security for our purposes would be UC security [11], but it may come at an efficiency cost. For efficient and UC-secure Σ-protocols [16], Dodis, Shoup, and Walfish [17] offer a solution, but it relies on verifiable encryption [10] or similar, which adds complexity and setup assumptions. In the random oracle model, Fischlin [19] as well as Bernhard, Fischlin, and Warinschi [3] show how to get an extractor that does not need to rewind, thereby allowing composition. If all we want is sequential composition, then we can rely on the fact that proofs of knowledge compose under sequential composition, but that means that in our unforgeability game, the signer can only respond to one signature query at a time.

At the heart of all of these approaches is an efficient Σ-protocol [16] that is then compiled (using the techniques cited above) into a ZKPoK. Depending on which flavor of ZKPoK is needed, the compiler may be very efficient (e.g., if a Fiat-Shamir proof is good enough) or relatively more involved (e.g., if we want UC security with the Fischlin compiler). Below, we give the Σ-protocols that are needed and refer the reader to the cited literature for the details of how to compile them to obtain a ZKPoK.

The signing protocol features two ZKPoKs. In Step 1, the Receiver of the signature on a message $m = (\hat{g}, u_1, \dots, u_n)$ gives a ZKPoK that, for all $1 \le i \le n$, he knows m_i such that $u_i = \hat{g}^{m_i}$. This can be instantiated using a standard transformation from a Σ-protocol for proving knowledge of a discrete logarithm (Figure 2) [26].

In Step 3 of the signing protocol, the Signer gives a ZKPoK that he has computed the glue element $\tilde{h}$ correctly. If verification of the glue and signature passes, the Receiver outputs the message $\tilde{m} = (\tilde{g}, \tilde{u}_1, \dots, \tilde{u}_n)$ and signature $(\tilde{h}, \sigma)$. The Σ-protocol for this ZKPoK can be viewed as a combination of the following Σ-protocols: (1) a proof of knowledge of a discrete logarithm; (2) a proof of knowledge of the opening of a commitment (Figure 2); (3) a proof of equality of two committed values; and (4) a proof that a committed value is the product of two other committed values (Figure 3) [24]. This last proof can be used to show that a committed value is the square of another committed value and, furthermore, that a committed value is the n^{th} power of another.

Common inputs:	**Common inputs:**
(p, g)	(p, g, h)
$X = g^x$	**public key of a commitment scheme**
	$X = g^x h^y$
Prover's inputs: x	**Prover's inputs:** x, y
Prove $\mapsto$ Verify	**Prove $\mapsto$ Verify**
$r \leftarrow \mathbb{Z}_p^*$	$r_1, r_2 \leftarrow \mathbb{Z}_p^*$
send $R \leftarrow g^r$	**send** $R \leftarrow g^{r_1} h^{r_2}$
Verify $\mapsto$ Prove	**Verify $\mapsto$ Prove**
send $c \leftarrow \mathbb{Z}_p^*$	**send** $c \leftarrow \mathbb{Z}_p^*$
Prove $\mapsto$ Verify	**Prove $\mapsto$ Verify**
send $s \leftarrow r + cx$	$s_1 \leftarrow r_1 + cx$
	$s_2 \leftarrow r_2 + cy$
	send s_1, s_2
Verify	**Verify**
return $(RX^c = g^z)$	**return** $(RX^c = g^{s_1} h^{s_2})$

Fig. 2. Σ-protocols for proving knowledge of a discrete logarithm (left-hand side) and knowledge of the opening of a commitment (right-hand side).

Common inputs:	**Common inputs:**
(p, g_1, h_1, g_2, h_2)	(p, g, h)
public key of a commitment scheme	**public key of a commitment scheme**
$X = g_1^x h_1^y$	$C_x = g^x h^{r_x}$
$Y = g_2^x h_2^z$	$C_y = g^y h^{r_y}$
	$C_z = g^{xy} h^{r_z}$
Prover's inputs: x, y, z	**Prover's inputs:** x, y, r_x, r_y, r_z
Prove $\mapsto$ Verify	**Prove $\leftrightarrow$ Verify**
$r_1, r_2, r_3 \leftarrow \mathbb{Z}_p^*$	**Two steps:**
$R_1 \leftarrow g_1^{r_1} h_1^{r_2}$	**1.** $PK\{(\alpha, \rho_x) : C_x = g^\alpha h^{\rho_x}\}$
$R_2 \leftarrow g_2^{r_2} h_2^{r_3}$	**2.** $PK\{(\beta, \rho_y, \rho') : C_y = g^\beta h^{\rho_y} \wedge C_z = C_x^\beta h^{\rho'}\}$
send R_1, R_2	
Verify $\mapsto$ Prove	
send $c \leftarrow \mathbb{Z}_p^*$	
Prove $\mapsto$ Verify	
$s_1 \leftarrow r_1 + cx$	
$s_2 \leftarrow r_2 + cy$	
$s_3 \leftarrow r_3 + cz$	
send s_1, s_2, s_3	
Verify	
return $(R_1 X^c = g_1^{s_1} h_1^{s_2}) \wedge (R_2 Y^c = g_2^{s_1} h_2^{s_3})$	

Fig. 3. Σ-protocols for proving the equality of committed values (left-hand side) and that a committed value is the product of two other committed values (right-hand side).

The honest verifier zero-knowledge property holds for these Σ-protocols information-theoretically. The knowledge extraction holds under the discrete logarithm assumption: the extractor algorithm either outputs the desired values or solves the instance of the discrete logarithm problem defined by the parameters of the system. Note that for the Σ-protocol proving the equality of committed values, the proof of security goes through as long as *one* of the commitment keys was chosen uniformly at random during setup; the other may be chosen arbitrarily. This was observed by Camenisch and Michels [9].

We can construct a Σ-protocol for a ZKPoK of the glue element $\tilde{h}$ as follows. The Signer (Prover) engages in following protocols with the Receiver (Verifier):

1. Prove knowledge of the discrete logarithm of $\hat{X}_6 = \hat{P}^{x_6}$ and $\hat{X}_7 \hat{X}_6^{-1} = \hat{P}^{\hat{x}}$. Let $\hat{X} = \hat{P}^{\hat{x}}$ and form a commitment $\hat{C}_{\hat{x}} = \hat{P}^{\hat{x}} \hat{H}^{\hat{r}_{\hat{x}}}$. Prove knowledge of the discrete logarithm of $\hat{C}_{\hat{x}} \hat{X}^{-1} = \hat{H}^{\hat{r}_{\hat{x}}}$. Repeat for $\hat{P}^{x_8}, \hat{P}^{x_8 \cdot y_1}, \hat{P}^{x_8 \cdot y_2}$ and $\hat{C}_{y_1} = \hat{P}^{y_1} \hat{H}^{\hat{r}_{y_1}}, \hat{C}_{y_2} = \hat{P}^{y_2} \hat{H}^{\hat{r}_{y_2}}$.
2. Form the commitments $C_{\hat{x}} = \hat{g}^{\hat{x}} H^{r_{\hat{x}}}, C_{y_1} = \hat{g}^{y_1} H^{r_{y_1}}, C_{y_2} = \hat{g}^{y_2} H^{r_{y_2}}, C_w = \hat{g}^w H^{r_w}$. Prove knowledge of the discrete logarithm of $\tilde{g} = \hat{g}^w$ and $C_w \tilde{g}^{-1} = H^{r_w}$. Prove the equality of the committed values in $\hat{C}_{\hat{x}} = \hat{P}^{\hat{x}} \hat{H}^{\hat{r}_{\hat{x}}}$ and $C_{\hat{x}} = \hat{g}^{\hat{x}} H^{r_{\hat{x}}}$.
3. Form the following commitments:

$$\begin{aligned} C_1 &= u_1^{w \cdot y_1 \cdot y_2} H^{r_1} \\ C_2 &= u_2^{w \cdot y_1 \cdot y_2 \cdot \hat{x}} H^{r_2} \\ &\vdots \\ C_{n-1} &= u_{n-1}^{w \cdot y_1 \cdot y_2 \cdot \hat{x}^{n-2}} H^{r_{n-1}} \end{aligned}$$

and prove that they are products of the contents of the commitments $C_w, C_{y_1}, C_{y_2}, C_{\hat{x}}$. (Note that the C_i's may be computed using u_i's as bases because their counterparts in the proofs of equality contain bases chosen from the public parameters.) Now compute $C_n = u_n^{w \cdot y_1 \cdot y_2 \cdot \hat{x}^{n-1}} H^{r_n}$, where $r_n = -\sum_{i=1}^{n-1} r_i$. Then:

$$\prod_{i=1}^{n} C_i = \Big(\prod_{i=1}^{n} \tilde{u}_i^{\hat{x}^{i-1}} \Big)^y = \tilde{h}$$

Lemma 2. *Under the discrete logarithm assumption, the protocol described for computing the glue element $\tilde{h}$ is a Σ-protocol zero-knowledge proof of knowledge.*

Proof. Since all of the Σ-protocols used are proofs of knowledge, the appropriate values can be extracted. As for the zero-knowledge property, form each commitment $C_1, \ldots, C_{n-1}$ at random. Then set

$$C_n = \frac{\tilde{h}}{\prod_{i=1}^{n-1} C_i}.$$

Next, invoke the zero-knowledge simulator of all of the constituent Σ-protocols. ☐

C Public Key Class-Hiding Proof

Proof. We now provide descriptions of the games and proofs of the claims in Section 4.4.

Game 0. In this real signing game, the public keys $\mathsf{pk}_{\mathsf{X},1}, \mathsf{pk}_{\mathsf{X},2}$ are independent, and the glue element is $\tilde{h} = \tilde{g}^{y^{(\delta)} \cdot q_\delta}$, where $q_\delta = p(\hat{x}_\delta)$ for $\delta \in \{1, 2\}$. There is no extraction or zero-knowledge simulation.

The challenger $\mathcal{C}$ computes the public parameters $PP = \mathsf{BG} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, P, \hat{P}, e)$ and two sets of keys for a mercurial signature scheme MS_f on messages of length $\ell = 5$: $(\mathsf{sk}_1, \mathsf{pk}_1) = ((x_{1,1}, x_{1,2}, x_{1,3}, x_{1,4}, x_{1,5}), (\hat{X}_{1,1}, \hat{X}_{1,2}, \hat{X}_{1,3}, \hat{X}_{1,4}, \hat{X}_{1,5}))$, $(\mathsf{sk}_2, \mathsf{pk}_2) = ((x_{2,1}, x_{2,2}, x_{2,3}, x_{2,4}, x_{2,5}), (\hat{X}_{2,1}, \hat{X}_{2,2}, \hat{X}_{2,3}, \hat{X}_{2,4}, \hat{X}_{2,5}))$, where $\hat{X}_{i,j} = \hat{P}^{x_{i,j}}$. $\mathcal{C}$ chooses uniformly at random secret points $\hat{x}_1, \hat{x}_2 \leftarrow \mathbb{Z}_p^*$ and secret seeds $y_1^{(1)}, y_1^{(2)}, y_2^{(1)}, y_2^{(2)} \leftarrow \mathbb{Z}_p^*$. He also picks $x_{1,6}, x_{2,6}, x_{1,8}, x_{2,8} \leftarrow \mathbb{Z}_p^*$ and sets: $x_{1,7} = x_{1,6} \cdot \hat{x}_1, x_{1,9} = x_{1,8} \cdot y_1^{(1)}, x_{1,10} = x_{1,8} \cdot y_2^{(1)}$, $x_{2,7} = x_{2,6} \cdot \hat{x}_2, x_{2,9} = x_{2,8} \cdot y_1^{(2)}, x_{2,10} = x_{2,8} \cdot y_2^{(2)}$. $\mathcal{C}$ then sets: $\mathsf{pk}_{\mathsf{X},1} = (\mathsf{pk}_1, \hat{X}_{1,6}, \hat{X}_{1,7}, \hat{X}_{1,8}, \hat{X}_{1,9}, \hat{X}_{1,10})$, $\mathsf{pk}_{\mathsf{X},2} = (\mathsf{pk}_2, \hat{X}_{2,6}, \hat{X}_{2,7}, \hat{X}_{2,8}, \hat{X}_{2,9}, \hat{X}_{2,10})$, where $\hat{X}_{i,j} = \hat{P}^{x_{i,j}}$. $\mathcal{C}$ forwards $PP_{\mathsf{X}} = PP$ and $\mathsf{pk}_{\mathsf{X},1}, \mathsf{pk}_{\mathsf{X},2}$ to $\mathcal{A}$.

$\mathcal{A}$ proceeds to make signature queries on messages of the form $m = (\hat{g}, u_1, \ldots, u_n) \in (\mathbb{G}_1^*)^{n+1}$, where $\hat{g}$ is a generator of $\mathbb{G}_1$. For each signature query, $\mathcal{A}$ selects whether he would like m to be signed under $\mathsf{sk}_{\mathsf{X},1}$ or $\mathsf{sk}_{\mathsf{X},2}$. $\mathcal{C}$ acts as the verifier while $\mathcal{A}$ gives a ZKPoK that, for all $1 \le i \le n$, he knows m_i such that $u_i = \hat{g}^{m_i}$. If the verification fails, $\mathcal{C}$ denies $\mathcal{A}$ the signature; otherwise, $\mathcal{C}$ computes $y^{(1)} = y_1^{(1)} \cdot y_2^{(1)}$ and $y^{(2)} = y_1^{(2)} \cdot y_2^{(2)}$ and: $\hat{h} = \left(\prod_{i=1}^n u_i^{\hat{x}_\delta^{i-1}}\right)^{y^{(\delta)}}$, where $\delta \in \{1, 2\}$ corresponds to the secret key $\mathsf{sk}_{\mathsf{X},\delta}$ $\mathcal{A}$ selected. $\mathcal{C}$ picks uniformly at random $w \leftarrow \mathbb{Z}_p^*$ and computes $\tilde{g} = \hat{g}^w, \tilde{h} = \hat{h}^w$, and $\tilde{u}_i = u_i^w \ \forall i$. He also computes $\tilde{g}^2, \ldots, \tilde{g}^n$. He then signs n messages of the form $M_i = (\tilde{g}, \tilde{g}^i, \tilde{g}^n, \tilde{h}, \tilde{u}_i)$ using his secret key sk_δ for MS_f and sends $\tilde{m} = (\tilde{g}, \tilde{u}_1, \ldots, \tilde{u}_n)$ and $(\tilde{h}, \sigma = \{\sigma_1, \ldots, \sigma_n\})$ to $\mathcal{A}$, along with a ZKPoK that $\tilde{h}$ was computed correctly. $\mathcal{A}$ issues queries for signatures on messages a polynomial number of times.

Game 1. In this game, the public keys $\mathsf{pk}_{\mathsf{X},1}, \mathsf{pk}_{\mathsf{X},2}$ are again independent, and the glue element is again $\tilde{h} = \tilde{g}^{y^{(\delta)} \cdot q_\delta}$, where $q_\delta = p(\hat{x}_\delta)$ for $\delta \in \{1, 2\}$; however, now there is simulation.

Game 1 is the same as Game 0, except the challenger $\mathcal{C}$ simulates the ZKPoK that $\tilde{h}$ was computed correctly.

Claim 1. *A PPT adversary cannot distinguish Game 0 from Game 1, except with negligible probability.*

The only difference between the two games is zero-knowledge simulation. In Game 1, the challenger simulates the ZKPoK that the glue $\tilde{h}$ was computed correctly, whereas in Game 0, the challenger gives a real ZKPoK. If an adversary could distinguish the two games, it would break the zero-knowledge property. This is the same as Claim 1 in the proof of unforgeability.

Game 2. In this game, the public keys $\mathsf{pk}_{\mathsf{X},1}, \mathsf{pk}_{\mathsf{X},2}$ are again independent, and the glue element is again $\tilde{h} = \tilde{g}^{y^{(\delta)} \cdot q_\delta}$, where $q_\delta = p(\hat{x}_\delta)$ for $\delta \in \{1, 2\}$; however, now there is extraction and simulation.

Game 2 is the same as Game 1, except for each signature query, the challenger $\mathcal{C}$ acts as the extractor while $\mathcal{A}$ gives a ZKPoK that, for all $1 \le i \le n$, he knows m_i such that $u_i = \hat{g}^{m_i}$. $\mathcal{C}$ extracts the m_i's, or if the extraction fails, $\mathcal{C}$ denies $\mathcal{A}$ the signature. $\mathcal{C}$ computes $\tilde{h}$ as in Game 1, signs n messages $M_i = (\tilde{g}, \tilde{g}^i, \tilde{g}^n, \tilde{h}, \tilde{u}_i)$ using his secret key sk_δ for MS_f, and sends $\tilde{m} = (\tilde{g}, \tilde{u}_1, \ldots, \tilde{u}_n)$ and $(\tilde{h}, \sigma = \{\sigma_1, \ldots, \sigma_n\})$ to $\mathcal{A}$, along with a simulated ZKPoK that $\tilde{h}$ was computed correctly.

Claim 2. *A PPT adversary cannot distinguish Game 1 from Game 2, except with negligible probability.*

The glue elements $\tilde{h}$ in both games are identical. The only difference between the two games is extraction. In Game 2, the challenger extracts the m_i's to compute the glue $\tilde{h}$, whereas in Game 1, the challenger computes the correct $\tilde{h}$ directly from the u_i's, without extracting the m_i's. If an adversary could distinguish the two games, it would break the knowledge extractor property. This is the same as Claim 2 in the proof of unforgeability.

Game 3. In this game, the public keys $\mathsf{pk}_{\mathsf{X},1}, \mathsf{pk}_{\mathsf{X},2}$ are now half in the same equivalence class and half independent, but the glue element remains $\tilde{h} = \tilde{g}^{y^{(\delta)} \cdot q_\delta}$, where $q_\delta = p(\hat{x}_\delta)$ for $\delta \in \{1, 2\}$.

Game 3 is the same as Game 2, except the challenger $\mathcal{C}$ computes pk_2 as pk_1^β for a uniformly random $\beta \leftarrow \mathbb{Z}_p^*$. $\mathcal{C}$ computes $\tilde{h}$ as in Game 2, signs n messages $M_i = (\tilde{g}, \tilde{g}^i, \tilde{g}^n, \tilde{h}, \tilde{u}_i)$ using his secret key sk_δ for MS_f, and sends $\tilde{m} = (\tilde{g}, \tilde{u}_1, \ldots, \tilde{u}_n)$ and $(\tilde{h}, \sigma = \{\sigma_1, \ldots, \sigma_n\})$ to $\mathcal{A}$, along with a simulated ZKPoK that $\tilde{h}$ was computed correctly.

Claim 3. *If a PPT adversary can distinguish Game 2 from Game 3 with non-negligible probability, then public key class-hiding of MS_f doesn't hold.*

Suppose a PPT adversary $\mathcal{A}$ can distinguish Game 2 from Game 3 for MS_X on messages of length n^*. Then, we construct a PPT reduction $\mathcal{B}$ for breaking public key-class hiding of MS_f as follows. $\mathcal{B}$ receives as input PP and two fixed public keys $\mathsf{pk}_1, \mathsf{pk}_2^b$ for a mercurial signature scheme MS_f on messages of length $\ell = 5$. His goal is to determine if $\mathsf{pk}_2^b \in [\mathsf{pk}_1]_{\mathcal{R}_{\mathsf{pk}}}$ or not. He constructs public keys $\mathsf{pk}_{\mathsf{X},1}, \mathsf{pk}_{\mathsf{X},2}$ as follows: $\mathsf{pk}_{\mathsf{X},1} = (\mathsf{pk}_1, \hat{X}_{1,6}, \hat{X}_{1,7}, \hat{X}_{1,8}, \hat{X}_{1,9}, \hat{X}_{1,10})$, $\mathsf{pk}_{\mathsf{X},2}^b = (\mathsf{pk}_2^b, \hat{X}_{2,6}, \hat{X}_{2,7}, \hat{X}_{2,8}, \hat{X}_{2,9}, \hat{X}_{2,10})$, where the $\hat{X}_{i,j}$'s are computed independently. $\mathcal{B}$ then forwards $PP_\mathsf{X} = PP$ and $\mathsf{pk}_{\mathsf{X},1}, \mathsf{pk}_{\mathsf{X},2}^b$ to $\mathcal{A}$.

For each signature query, $\mathcal{A}$ selects whether he would like the message m to be signed under $\mathsf{sk}_{\mathsf{X},1}$ or $\mathsf{sk}_{\mathsf{X},2}^b$. $\mathcal{B}$ extracts the m_i's, or if the extraction fails, $\mathcal{B}$ denies $\mathcal{A}$ the signature. $\mathcal{B}$ computes $\tilde{h}$ as in Game 2/Game 3, forwards n^* messages $M_i = (\tilde{g}, \tilde{g}^i, \tilde{g}^{n^*}, \tilde{h}, \tilde{u}_i)$ to the appropriate signing oracle, either $\mathsf{Sign}_f(\mathsf{sk}_1, \cdot)$ or $\mathsf{Sign}_f(\mathsf{sk}_2^b, \cdot)$, and forwards the signature $\tilde{m} = (\tilde{g}, \tilde{u}_1, \ldots, \tilde{u}_{n^*})$ and $(\tilde{h}, \sigma = \{\sigma_1, \ldots, \sigma_{n^*}\})$ to $\mathcal{A}$, along with a simulated ZKPoK that $\tilde{h}$ was computed correctly. It is clear that $\mathsf{pk}_{\mathsf{X},2}^b$ is half in the same equivalence class as $\mathsf{pk}_{\mathsf{X},1}$ and half independent (Game 3) if and only if $\mathsf{pk}_2^b \in [\mathsf{pk}_1]_{\mathcal{R}_{\mathsf{pk}}}$, so $\mathcal{A}$'s success in distinguishing Game 2 from Game 3 translates directly into $\mathcal{B}$'s success in breaking public key class-hiding of MS_f.

Game 4. In this game, the public keys $\mathsf{pk}_{\mathsf{X},1}, \mathsf{pk}_{\mathsf{X},2}$ are again half in the same equivalence class and half independent, but the glue element is $\tilde{h} = \tilde{g}^{R_\delta(q_\delta)}$, where $q_\delta = p(\hat{x}_\delta)$ and $R_\delta : \mathbb{Z}_p^* \to \mathbb{Z}_p^*$ is a random function for $\delta \in \{1, 2\}$.

The challenger $\mathcal{C}$ computes the public keys as: $\mathsf{pk}_{\mathsf{X},1} = (\mathsf{pk}_1, \hat{X}_{1,6}, \hat{X}_{1,7}, \hat{X}_{1,8}, \hat{X}_{1,9}, \hat{X}_{1,10})$, $\mathsf{pk}_{\mathsf{X},2} = (\mathsf{pk}_1^\beta, \hat{X}_{2,6}, \hat{X}_{2,7}, \hat{X}_{2,8}, \hat{X}_{2,9}, \hat{X}_{1,10})$, where the $\hat{X}_{i,j}$'s are computed independently. $\mathcal{C}$ chooses two random functions $R_1, R_2 : \mathbb{Z}_p^* \to \mathbb{Z}_p^*$ and computes $\tilde{h} = \tilde{g}^{R_\delta(q_\delta)}$ according to the secret key $\mathsf{sk}_{\mathsf{X},\delta}$ $\mathcal{A}$ selected. He then signs n messages $M_i = (\tilde{g}, \tilde{g}^i, \tilde{g}^n, \tilde{h}, \tilde{u}_i)$ using his secret key sk_δ for MS_f, and sends $\tilde{m} = (\tilde{g}, \tilde{u}_1, \ldots, \tilde{u}_n)$ and $(\tilde{h}, \sigma = \{\sigma_1, \ldots, \sigma_n\})$ to $\mathcal{A}$, along with a simulated ZKPoK that $\tilde{h}$ was computed correctly.

Claim 4. *A PPT adversary cannot distinguish Game 3 from Game 4 under the ABDDH$^+$ assumption in $\mathbb{G}_1$.*

This is very similar to Claim 3 (ABDDH$^+$) in the proof of unforgeability.

Game 5. In this game, the public keys $\mathsf{pk}_{\mathsf{X},1}, \mathsf{pk}_{\mathsf{X},2}$ are again half in the same equivalence class and half independent, but the glue element is $\tilde{h} = \tilde{g}^{R_\delta(\dot{q}_\delta)}$, where $\dot{q}_\delta = p(\alpha_\delta)$, α_δ is a "fake" secret point, and $R_\delta : \mathbb{Z}_p^* \to \mathbb{Z}_p^*$ is a random function for $\delta \in \{1, 2\}$.

Game 5 is the same as Game 4, except the challenger $\mathcal{C}$ computes the glue element as $\tilde{h} = \tilde{g}^{R_\delta(\dot{q}_\delta)}$, where $\delta \in \{1, 2\}$ corresponds to the secret key $\mathsf{sk}_{\mathsf{X},\delta}$ $\mathcal{A}$ selected.

Claim 5. *A PPT adversary $\mathcal{A}$ cannot distinguish Game 4 from Game 5 under the DL assumption in $\mathbb{G}_2$.*

The only difference between the two games is that in Game 5, the polynomials $p_j(x)$ are evaluated at a "fake" secret point α_δ, which is independent of the true secret point $\hat{x}_\delta$. If $q_{\delta,i} = q_{\delta,j}$ for some $p_i(x) \neq p_j(x)$ and some $\delta \in \{1, 2\}$, then $R_\delta(q_{\delta,i}) = R_\delta(q_{\delta,j})$, so $\mathcal{A}$ learns that $p_i(\hat{x}_\delta) = p_j(\hat{x}_\delta)$. We showed in Claim 4 of the proof of unforgeability that a collision $p_i(\alpha_\delta) = p_j(\alpha_\delta)$ occurs with negligible probability, so $\mathcal{A}$ can distinguish Game 4 from Game 5 only if a collision $p_i(\hat{x}_\delta) = p_j(\hat{x}_\delta)$ occurs in Game 4 with non-negligible probability. We showed in Claim 5 of the proof of unforgeability that such a collision occurs with negligible probability, or the DL assumption doesn't hold.

Game 6. In this game, the public keys $\mathsf{pk}_{\mathsf{X},1}, \mathsf{pk}_{\mathsf{X},2}$ are again half in the same equivalence class and half independent, but the glue element is $\tilde{h} = \tilde{g}^{R_\delta(m_1, \ldots, m_n)}$, where $R_\delta : (\mathbb{Z}_p^*)^n \to \mathbb{Z}_p^*$ is a random function for $\delta \in \{1, 2\}$.

Game 6 is the same as Game 5, except the challenger $\mathcal{C}$ computes the glue element as $\tilde{h} = \tilde{g}^{R_\delta(m_1, \ldots, m_n)}$, where $\delta \in \{1, 2\}$ corresponds to the secret key $\mathsf{sk}_{\mathsf{X},\delta}$ $\mathcal{A}$ selected.

Claim 6. *An adversary's view in Game 5 is the same as it is in Game 6, except with negligible probability.*

If $\dot{q}_{\delta,i} = \dot{q}_{\delta,j}$ for some $p_i(x) \neq p_j(x)$ and some $\delta \in \{1,2\}$, then $R_\delta(\dot{q}_{\delta,i}) = R_\delta(\dot{q}_{\delta,j})$, so $\mathcal{A}$ learns that $p_i(\alpha_\delta) = p_j(\alpha_\delta)$. The value α_δ is independent of the adversary's view unless such a collision occurs. We showed in Claim 4 of the proof of unforgeability that a collision $p_i(\alpha_\delta) = p_j(\alpha_\delta)$ occurs with negligible probability. If there are no such collisions, $\mathcal{A}$'s view is identical in both games because he receives random values.

Game 7. In this game, the public keys $\mathsf{pk}_{\mathsf{X},1}, \mathsf{pk}_{\mathsf{X},2}$ are again half in the same equivalence class and half independent, but the glue element is $\tilde{h} = \tilde{g}^{R(m_1,\ldots,m_n)}$, where $R : (\mathbb{Z}_p^*)^n \to \mathbb{Z}_p^*$ is a random function.

Game 7 is the same as Game 6, except the challenger $\mathcal{C}$ computes the glue element as $\tilde{h} = \tilde{g}^{R(m_1,\ldots,m_n)}$. Note that the same function R is used regardless of which secret key $\mathsf{sk}_{\mathsf{X},\delta}$ $\mathcal{A}$ selected.

Claim 7. *A PPT adversary cannot distinguish Game 6 from Game 7 under the DDH assumption in $\mathbb{G}_1$.*

We prove this via a hybrid argument. Suppose a PPT adversary $\mathcal{A}$ can distinguish hybrids $\mathcal{H}_i$ from $\mathcal{H}_{i+1}$ (described below) for some i with non-negligible probability (bounded by the best advantage in breaking DDH). Then, we construct a PPT reduction $\mathcal{B}$ for breaking the DDH assumption as follows. $\mathcal{B}$ receives as input (g_0, A, B, C), where g_0 is a generator of $\mathbb{G}_1$ and implicitly $A = g_0^a, B = g_0^b$, and $C = g_0^{ab}$ or g_0^r for some uniformly random $a, b, r \in \mathbb{Z}_p^*$. He computes public parameters PP and public keys $\mathsf{pk}_{\mathsf{X},1}, \mathsf{pk}_{\mathsf{X},2}$ as follows: $\mathsf{pk}_{\mathsf{X},1} = (\mathsf{pk}_1, \hat{X}_{1,6}, \hat{X}_{1,7}, \hat{X}_{1,8}, \hat{X}_{1,9}, \hat{X}_{1,10}), \mathsf{pk}_{\mathsf{X},2} = (\mathsf{pk}_1^\beta, \hat{X}_{2,6}, \hat{X}_{2,7}, \hat{X}_{2,8}, \hat{X}_{2,9}, \hat{X}_{2,10})$, where the $\hat{X}_{i,j}$'s are computed independently. $\mathcal{B}$ chooses random functions $R_1, R_2 : (\mathbb{Z}_p^*)^n \to \mathbb{Z}_p^*$ and forwards $PP_{\mathsf{X}} = PP$ and $\mathsf{pk}_{\mathsf{X},1}, \mathsf{pk}_{\mathsf{X},2}$ to $\mathcal{A}$.

Let $\mathcal{A}$'s j^{th} signature query be on message $m_j = (\hat{g}_j, u_{j,1}, \ldots, u_{j,n})$. $\mathcal{B}$ acts as the extractor while $\mathcal{A}$ gives a ZKPoK that, for all $1 \le i \le n$, he knows $m_{j,i}$ such that $u_{j,i} = \hat{g}_j^{m_{j,i}}$. $\mathcal{B}$ extracts the $m_{j,i}$'s, or if the extraction fails, $\mathcal{B}$ denies $\mathcal{A}$ the signature. Otherwise,

(1) If $j \le i$, $\mathcal{B}$ computes: $\tilde{h}_j^{(1)} = (\tilde{g}_j^{(1)})^{R_1(m_{j,1},\ldots,m_{j,n})}, \tilde{h}_j^{(2)} = (\tilde{g}_j^{(2)})^{R_2(m_{j,1},\ldots,m_{j,n})}$. He signs n messages $M_{j,i}^{(1)} = (\tilde{g}_j^{(1)}, (\tilde{g}_j^{(1)})^i, (\tilde{g}_j^{(1)})^n, \tilde{h}_j^{(1)}, \tilde{u}_{j,i}^{(1)})$ using the secret key sk_1 for MS_f and also signs n messages $M_{j,i}^{(2)} = (\tilde{g}_j^{(2)}, (\tilde{g}_j^{(2)})^i, (\tilde{g}_j^{(2)})^n, \tilde{h}_j^{(2)}, \tilde{u}_{j,i}^{(2)})$ using the secret key sk_2 for MS_f. $\mathcal{B}$ sends $\tilde{m}_j^{(1)} = (\tilde{g}_j^{(1)}, \tilde{u}_{j,1}^{(1)}, \ldots, \tilde{u}_{j,n}^{(1)})$, $(\tilde{h}_j^{(1)}, \sigma_j^{(1)} = \{\sigma_{j,1}^{(1)}, \ldots, \sigma_{j,n}^{(1)}\})$, $\tilde{m}_j^{(2)} = (\tilde{g}_j^{(2)}, \tilde{u}_{j,1}^{(2)}, \ldots, \tilde{u}_{j,n}^{(2)})$, and $(\tilde{h}_j^{(2)}, \sigma_j^{(2)} = \{\sigma_{j,1}^{(2)}, \ldots, \sigma_{j,n}^{(2)}\})$ to $\mathcal{A}$, along with simulated ZKPoKs that $\tilde{h}_j^{(1)}$ and $\tilde{h}_j^{(2)}$ are computed correctly.

(2) If $j = i + 1$, $\mathcal{B}$ computes: $\tilde{g}_j^{(1)} = g_0, \tilde{h}_j^{(1)} = B, \tilde{u}_{j,i}^{(1)} = g_0^{m_{j,i}}$ $\forall i, \tilde{g}_j^{(2)} = A, \tilde{h}_j^{(2)} = C, \tilde{u}_{j,i}^{(2)} = A^{m_{j,i}}$ $\forall i$. He then signs n messages $M_{j,i}^{(1)} = (g_0, g_0^i, g_0^n, B, g_0^{m_{j,i}})$ using sk_1 and n messages $M_{j,i}^{(2)} = (A, A^i, A^n, C, A^{m_{j,i}})$ using sk_2 and sends the signatures and simulated proofs to $\mathcal{A}$.

(3) If $j > i + 1$, $\mathcal{B}$ computes: $\tilde{h}_j^{(1)} = (\tilde{g}_j^{(1)})^{R_1(m_{j,1},\ldots,m_{j,n})}$, $\tilde{h}_j^{(2)} = (\tilde{g}_j^{(2)})^{R_1(m_{j,1},\ldots,m_{j,n})}$. He signs the messages $M_{j,i}^{(1)}, M_{j,i}^{(2)}$ and forwards the signatures and simulated proofs to $\mathcal{A}$.

Let $\Gamma(k)$ be the number of queries $\mathcal{A}$ makes. Hybrid $\mathcal{H}_0$ corresponds to the game in which all glue elements are formed as $\tilde{h}_j = \tilde{g}_j^{R_1(m_1,\ldots,m_n)}$ (Game 7), while $\mathcal{H}_{\Gamma(k)}$ corresponds to the game in which all glue elements are formed as $\tilde{h}_j = \tilde{g}_j^{R_\delta(m_1,\ldots m_n)}$ for $\delta \in \{1,2\}$ (Game 6). $C = g_0^{ab}$ corresponds to hybrid $\mathcal{H}_i$ and $C = g_0^r$ corresponds to hybrid $\mathcal{H}_{i+1}$. Thus, if $\mathcal{A}$ is able to distinguish $\mathcal{H}_i$ from $\mathcal{H}_{i+1}$ for some i with non-negligible probability, then $\mathcal{B}$ breaks the DDH assumption.

Game 8. In this game, now $\mathsf{pk}_{\mathsf{X},2} \in [\mathsf{pk}_{\mathsf{X},1}]_{\mathcal{R}_{\mathsf{pk}}}$, but the glue element remains $\tilde{h} = \tilde{g}^{R(m_1,\ldots,m_n)}$,where $R : (\mathbb{Z}_p^*)^n \to \mathbb{Z}_p^*$ is a random function.

Game 8 is the same as Game 7, except the challenger $\mathcal{C}$ computes the public keys as $\mathsf{pk}_{\mathsf{X},2} = \mathsf{pk}_{\mathsf{X},1}^\beta$ for a uniformly random $\beta \leftarrow \mathbb{Z}_p^*$.

Claim 8. *A PPT adversary cannot distinguish Game 7 from Game 8 under the DDH assumption in $\mathbb{G}_2$.*

Consider the following set of games. In each game, $\tilde{h} = \tilde{g}^{R(m_1,\ldots,m_n)}$, and the reduction $\mathcal{B}$ receives as input $(\hat{g}_0, \hat{A}, \hat{B}, \hat{C})$, where $\hat{g}_0$ is a generator of $\mathbb{G}_2$ and implicitly $\hat{A} = \hat{g}_0^a, \hat{B} = \hat{g}_0^b$, and $\hat{C} = \hat{g}_0^{ab}$ or $\hat{g}_0^r$ for some uniformly random $a, b, r \in \mathbb{Z}_p^*$.

Game 7. Recall that $\mathsf{pk}_{\mathsf{X},1}$ and $\mathsf{pk}_{\mathsf{X},2}$ are of the form: $\mathsf{pk}_{\mathsf{X},1} = (\mathsf{pk}_1, \hat{X}_{1,6}, \hat{X}_{1,6}^{\hat{x}_1}, \hat{X}_{1,8}, \hat{X}_{1,8}^{y_1^{(1)}}, \hat{X}_{1,8}^{y_2^{(1)}}), \mathsf{pk}_{\mathsf{X},2} = (\mathsf{pk}_1^\beta, \hat{X}_{1,6}^\gamma, (\hat{X}_{1,6}^\gamma)^{\hat{x}_2}, \hat{X}_{1,8}^\lambda, (\hat{X}_{1,8}^\lambda)^{y_1^{(2)}}, (\hat{X}_{1,8}^\lambda)^{y_2^{(2)}})$, where $\beta, \gamma, \lambda, \hat{x}_1, \hat{x}_2, y_1^{(1)}, y_1^{(2)}, y_2^{(1)}, y_2^{(2)} \leftarrow \mathbb{Z}_p^*$ are all uniformly random.

Intermediate Game 1. Consider $\mathsf{pk}_{\mathsf{X},1}$ and $\mathsf{pk}_{\mathsf{X},2}$ of the form: $\mathsf{pk}_{\mathsf{X},1} = (\mathsf{pk}_1, \hat{X}_{1,6}, \hat{X}_{1,6}^{\hat{x}}, \hat{X}_{1,8}, \hat{X}_{1,8}^{y_1^{(1)}}, \hat{X}_{1,8}^{y_2^{(1)}})$, $\mathsf{pk}_{\mathsf{X},2} = (\mathsf{pk}_1^\beta, \hat{X}_{1,6}^\gamma, (\hat{X}_{1,6}^\gamma)^{\hat{x}}, \hat{X}_{1,8}^\lambda, (\hat{X}_{1,8}^\lambda)^{y_1^{(2)}}, (\hat{X}_{1,8}^\lambda)^{y_2^{(2)}})$, where $\beta, \gamma, \lambda, y_1^{(1)}, y_1^{(2)}, y_2^{(1)}, y_2^{(2)} \leftarrow \mathbb{Z}_p^*$ are all uniformly random.

The reduction plugs in the DDH challenge $(\hat{g}_0, \hat{A}, \hat{B}, \hat{C})$ as follows: $\mathsf{pk}_{\mathsf{X},1} = (\mathsf{pk}_1, \hat{g}_0, \hat{A}, \hat{X}_{1,8}, \hat{X}_{1,8}^{y_1^{(1)}}, \hat{X}_{1,8}^{y_2^{(1)}})$, $\mathsf{pk}_{\mathsf{X},2} = (\mathsf{pk}_1^{\beta}, \hat{B}, \hat{C}, \hat{X}_{1,8}^{\lambda}, (\hat{X}_{1,8}^{\lambda})^{y_1^{(2)}}, (\hat{X}_{1,8}^{\lambda})^{y_2^{(2)}})$, where $\beta, \lambda, y_1^{(1)}, y_1^{(2)}, y_2^{(1)}, y_2^{(2)} \leftarrow \mathbb{Z}_p^*$ are all uniformly random. Thus, we have that $\hat{x}_1 = a$ and $\gamma = b$. If $\hat{C} = \hat{g}_0^{ab}$, then $\hat{x}_2 = a = \hat{x}_1$ (Int. Game 1). If $\hat{C} = \hat{g}_0^{r}$, then $\hat{x}_1$ and $\hat{x}_2$ are independent (Game 7).

Intermediate Game 2. Consider $\mathsf{pk}_{\mathsf{X},1}$ and $\mathsf{pk}_{\mathsf{X},2}$ of the form: $\mathsf{pk}_{\mathsf{X},1} = (\mathsf{pk}_1, \hat{X}_{1,6}, \hat{X}_{1,6}^{\hat{x}}, \hat{X}_{1,8}, \hat{X}_{1,8}^{y_1}, \hat{X}_{1,8}^{y_2^{(1)}})$, $\mathsf{pk}_{\mathsf{X},2} = (\mathsf{pk}_1^{\beta}, \hat{X}_{1,6}^{\gamma}, (\hat{X}_{1,6}^{\gamma})^{\hat{x}}, \hat{X}_{1,8}^{\lambda}, (\hat{X}_{1,8}^{\lambda})^{y_1}, (\hat{X}_{1,8}^{\lambda})^{y_2^{(2)}})$, where $\beta, \gamma, \lambda, y_2^{(1)}, y_2^{(2)} \leftarrow \mathbb{Z}_p^*$ are all uniformly random.

The reduction plugs in the DDH challenge $(\hat{g}_0, \hat{A}, \hat{B}, \hat{C})$ as follows: $\mathsf{pk}_{\mathsf{X},1} = (\mathsf{pk}_1, \hat{X}_{1,6}, \hat{X}_{1,6}^{\hat{x}}, \hat{g}_0, \hat{A}, \hat{g}_0^{y_2^{(1)}})$, $\mathsf{pk}_{\mathsf{X},2} = (\mathsf{pk}_1^{\beta}, \hat{X}_{1,6}^{\gamma}, (\hat{X}_{1,6}^{\gamma})^{\hat{x}}, \hat{B}, \hat{C}, \hat{B}^{y_2^{(2)}})$, where $\beta, \gamma, y_2^{(1)}, y_2^{(2)} \leftarrow \mathbb{Z}_p^*$ are all uniformly random. Thus, we have that $y_1^{(1)} = a$ and $\lambda = b$. If $\hat{C} = \hat{g}_0^{ab}$, then $y_1^{(2)} = y_1^{(1)}$ (Int. Game 2). If $\hat{C} = \hat{g}_0^{r}$, then $y_1^{(1)}$ and $y_1^{(2)}$ are independent (Int. Game 1).

Intermediate Game 3. Consider $\mathsf{pk}_{\mathsf{X},1}$ and $\mathsf{pk}_{\mathsf{X},2}$ of the form: $\mathsf{pk}_{\mathsf{X},1} = (\mathsf{pk}_1, \hat{X}_{1,6}, \hat{X}_{1,6}^{\hat{x}}, \hat{X}_{1,8}, \hat{X}_{1,8}^{y_1}, \hat{X}_{1,8}^{y_2})$, $\mathsf{pk}_{\mathsf{X},2} = (\mathsf{pk}_1^{\beta}, \hat{X}_{1,6}^{\gamma}, (\hat{X}_{1,6}^{\gamma})^{\hat{x}}, \hat{X}_{1,8}^{\lambda}, (\hat{X}_{1,8}^{\lambda})^{y_1}, (\hat{X}_{1,8}^{\lambda})^{y_2})$, where $\beta, \gamma, \lambda \leftarrow \mathbb{Z}_p^*$ are all uniformly random.

The reduction plugs in the DDH challenge $(\hat{g}_0, \hat{A}, \hat{B}, \hat{C})$ as follows: $\mathsf{pk}_{\mathsf{X},1} = (\mathsf{pk}_1, \hat{X}_{1,6}, \hat{X}_{1,6}^{\hat{x}}, \hat{g}_0, \hat{g}_0^{y_1}, \hat{A})$, $\mathsf{pk}_{\mathsf{X},2} = (\mathsf{pk}_1^{\beta}, \hat{X}_{1,6}^{\gamma}, (\hat{X}_{1,6}^{\gamma})^{\hat{x}}, \hat{B}, \hat{B}^{y_1}, \hat{C})$, where $\beta, \gamma \leftarrow \mathbb{Z}_p^*$ are uniformly random. Thus, we have that $y_2^{(1)} = a$ and $\lambda = b$. If $\hat{C} = \hat{g}_0^{ab}$, then $y_2^{(2)} = y_2^{(1)}$ (Int. Game 3). If $\hat{C} = \hat{g}_0^{r}$, then $y_1^{(1)}$ and $y_1^{(2)}$ are independent (Int. Game 2).

Intermediate Game 4. Consider $\mathsf{pk}_{\mathsf{X},1}$ and $\mathsf{pk}_{\mathsf{X},2}$ of the form: $\mathsf{pk}_{\mathsf{X},1} = (\mathsf{pk}_1, \hat{X}_{1,6}, \hat{X}_{1,6}^{\hat{x}}, \hat{X}_{1,8}, \hat{X}_{1,8}^{y_1}, \hat{X}_{1,8}^{y_2})$, $\mathsf{pk}_{\mathsf{X},2} = (\mathsf{pk}_1^{\beta}, \hat{X}_{1,6}^{\gamma}, (\hat{X}_{1,6}^{\gamma})^{\hat{x}}, \hat{X}_{1,8}^{\gamma}, (\hat{X}_{1,8}^{\gamma})^{y_1}, (\hat{X}_{1,8}^{\gamma})^{y_2})$, where $\beta, \gamma \leftarrow \mathbb{Z}_p^*$ are uniformly random.

The reduction plugs in the DDH challenge $(\hat{g}_0, \hat{A}, \hat{B}, \hat{C})$ as follows: $\mathsf{pk}_{\mathsf{X},1} = (\mathsf{pk}_1, \hat{g}_0, \hat{g}_0^{\hat{x}}, \hat{B}, \hat{B}^{y_1}, \hat{B}^{y_2})$, $\mathsf{pk}_{\mathsf{X},2} = (\mathsf{pk}_1^{\beta}, \hat{A}, \hat{A}^{\hat{x}}, \hat{C}, \hat{C}^{y_1}, \hat{C}^{y_2})$. Thus, we have that $\gamma = a$. If $\hat{C} = \hat{g}_0^{ab}$, then $\lambda = a = \gamma$ (Int. Game 4). If $\hat{C} = \hat{g}_0^{r}$, then $\hat{C}$ is distributed the same as $\hat{B}^{\lambda}$ for λ independent from γ (Int. Game 3).

Game 8. Recall that $\mathsf{pk}_{\mathsf{X},1}$ and $\mathsf{pk}_{\mathsf{X},2}$ are of the form: $\mathsf{pk}_{\mathsf{X},1} = (\mathsf{pk}_1, \hat{X}_{1,6}, \hat{X}_{1,6}^{\hat{x}}, \hat{X}_{1,8}, \hat{X}_{1,8}^{y_1}, \hat{X}_{1,8}^{y_2})$, $\mathsf{pk}_{\mathsf{X},2} = (\mathsf{pk}_1^{\beta}, \hat{X}_{1,6}^{\beta}, (\hat{X}_{1,6}^{\beta})^{\hat{x}}, \hat{X}_{1,8}^{\beta}, (\hat{X}_{1,8}^{\beta})^{y_1}, (\hat{X}_{1,8}^{\beta})^{y_2})$, where $\beta \leftarrow \mathbb{Z}_p^*$ is uniformly random.

The reduction plugs in the DDH challenge $(\hat{g}_0, \hat{A}, \hat{B}, \hat{C})$ as follows: $\mathsf{pk}_{\mathsf{X},1} = (\hat{g}_0, \hat{g}_0^{x_{1,2}}, \hat{g}_0^{x_{1,3}}, \hat{g}_0^{x_{1,4}}, \hat{g}_0^{x_{1,5}}, \hat{A}, \hat{A}^{\hat{x}}, \hat{A}^{\omega}, (\hat{A}^{\omega})^{y_1}, (\hat{A}^{\omega})^{y_2})$, $\mathsf{pk}_{\mathsf{X},2} = (\hat{B}, \hat{B}^{x_{1,2}}, \hat{B}^{x_{1,3}}, \hat{B}^{x_{1,4}}, \hat{B}^{x_{1,5}}, \hat{C}, \hat{C}^{\hat{x}}, \hat{C}^{\omega}, (\hat{C}^{\omega})^{y_1}, (\hat{C}^{\omega})^{y_2})$, where the $x_{i,j}$'s and ω are uniformly random. If $\hat{C} = \hat{g}_0^{ab}$, then $\mathsf{pk}_{\mathsf{X},2} = (\mathsf{pk}_{\mathsf{X},1})^{b}$, so $\mathsf{pk}_{\mathsf{X},2} \in [\mathsf{pk}_{\mathsf{X},1}]_{\mathcal{R}_{\mathsf{pk}}}$ (Game 8). If $\hat{C} = \hat{g}_0^{r}$, then $\hat{C}$ is distributed the same as $\hat{A}^{\gamma}$ for some γ independent from b (Int. Game 4).

Game 8. $\mathsf{pk}_{\mathsf{X},2} \in [\mathsf{pk}_{\mathsf{X},1}]_{\mathcal{R}_{\mathsf{pk}}}$, $\tilde{h} = \tilde{g}^{R(m_1,\ldots,m_n)}$.

$\updownarrow$ Claim 9: polynomial collision argument, same as unforgeability Claim 6

Game 9. $\mathsf{pk}_{\mathsf{X},2} \in [\mathsf{pk}_{\mathsf{X},1}]_{\mathcal{R}_{\mathsf{pk}}}$, $\tilde{h} = \tilde{g}^{R(\dot{q})}$.

$\updownarrow$ Claim 10: polynomial collision argument and DL assumption in $\mathbb{G}_2$, similar to unforgeability Claims 4/5

Game 10. $\mathsf{pk}_{\mathsf{X},2} \in [\mathsf{pk}_{\mathsf{X},1}]_{\mathcal{R}_{\mathsf{pk}}}$, $\tilde{h} = \tilde{g}^{R(q)}$.

$\updownarrow$ Claim 11: ABDDH^{+}, same as unforgeability Claim 3

Game 11. $\mathsf{pk}_{\mathsf{X},2} \in [\mathsf{pk}_{\mathsf{X},1}]_{\mathcal{R}_{\mathsf{pk}}}$, $\tilde{h} = \tilde{g}^{y \cdot q}$.

$\updownarrow$ Claim 12: knowledge extractor property, same as unforgeability Claim 2

Game 12. $\mathsf{pk}_{\mathsf{X},2} \in [\mathsf{pk}_{\mathsf{X},1}]_{\mathcal{R}_{\mathsf{pk}}}$, $\tilde{h} = \tilde{g}^{y \cdot q}$. No extraction.

$\updownarrow$ Claim 13: ZK property, same as unforge Claim 1

Game 13. $\mathsf{pk}_{\mathsf{X},2} \in [\mathsf{pk}_{\mathsf{X},1}]_{\mathcal{R}_{\mathsf{pk}}}$, $\tilde{h} = \tilde{g}^{y \cdot q}$. No extraction or ZK simulation. This is the real signing game.

□