Mercurial Signatures for Variable-Length Messages*

Elizabeth C. Crites[1] and Anna Lysyanskaya[2]

[1] University College London, UK
e.crites@ucl.ac.uk
[2] Brown University, USA
anna@cs.brown.edu

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 n, the signer's public key must also be of length n. In this paper, we eliminate this restriction and provide a signing protocol that admits messages of any length. This significantly improves the applicability of mercurial signatures to chains of anonymous credentials.

Keywords: Signature schemes, anonymous credentials.

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}, attributes $attr_{Alice}$, and a signature on both from the CA: $\sigma_{CA \rightarrow Alice}$. Suppose Bob obtains a certificate from Alice, rather than directly from the CA. As a result, Bob's certificate consists of Alice's public key pk_{Alice}, attributes $attr_{Alice}$, and certificate from the CA: $\sigma_{CA \rightarrow Alice}$ as well as his own public key pk_{Bob}, attributes $attr_{Bob}$, and certificate from Alice: $\sigma_{Alice \rightarrow Bob}$.

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

* Supported by EPSRC Grant EP/N028104/1[1] and NSF Grant 1422361[1,2].

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: the public key under which the signature verifies can 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, which allows the signed message to be transformed. Using the public-key-blinding feature, which allows the public key to be transformed, 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. Also, with care in how attributes are incorporated, a user may have the option of disclosing only a subset of her attributes and those higher up in her certification chain.

Mercurial signatures were introduced in a recent paper by Crites and Lysyanskaya [CL19]. A construction was provided in which messages and public keys are fixed-length vectors of group elements. Specifically, messages and public keys are of the form $M = (M_1, \ldots, M_\ell)$ and $\mathsf{pk} = (\hat{X}_1, \ldots, \hat{X}_\ell)$ for a fixed length ℓ, where M and pk are defined over bilinear groups $\mathbb{G}_1$ and $\mathbb{G}_2$. Mercurial signatures allow a message M to be transformed into an equivalent message $M' = (M_1^\mu, \ldots, M_\ell^\mu)$ for a scalar μ, and public keys may be transformed similarly.

Our new construction of mercurial signatures 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 attributes. 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. 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', the attributes remain the same, and thus her certificate still authorizes access to the same buildings at the same times. Should she then issue a certificate to Bob, she may grant him access to the same buildings or a subset of the buildings to which she has access and potentially limit the hours during which Bob is authorized. (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.)

The construction in [CL19] worked in a limited way: the length of a user's public key was an upper bound on the length of the message that user could sign. For example, if Alices's public key is of length ℓ and her attributes are of length k, as above, 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 credential chain).

1.1 Related Work

Our motivating application is anonymous credentials [Cha86,LRSW99,CL01,Lys02] [CL04,CKL+14,CDHK15]. 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) and government policy (such as the NSTIC document released by the Obama administration).

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. Furthermore, mercurial signatures may be used as a building block for even more interesting applications, such as *delegatable* anonymous credentials [CL19]. 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.

Mercurial signatures were inspired by Fuchsbauer, Hanser and Slamanig's work on structure-preserving signatures on equivalence classes (SPS-EQ) [FHS19], which introduces the idea of transforming a signature σ on a message m into a signature σ' on an equivalent but unlinkable message m'. A related concept, signatures with flexible public key [BHKS18], allows blinding of the public key, but not the message.

1.2 Our Contribution

The only previously known construction of mercurial signatures [CL19] 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.

We fell short of our goal as far as unforgeability is concerned: instead of constructing a mercurial signature scheme that is existentially unforgeable under adaptive chosen message attack (EUF-CMA), we construct a scheme that is unforgeable in a more limited sense. Namely, instead of the adversary having access to the regular signing oracle that, on input a message m, responds with its signature σ, it obtains signatures via a *signing protocol* in which it is required to prove knowledge of discrete logarithms of components of m. This variant of unforgeability was defined by Fuchsbauer and Gay [FG18] as existential unforgeability under chosen *open* message attack (EUF-CoMA).

Unfortunately, we also fell short of our goal 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; the *origin-hiding* property of mercurial signatures guarantees that the resulting signature σ' is distributed identically to what Bob would have gotten by having m' signed anew. Our construction can only guarantee origin-hiding if the signer follows the signing algorithm; a malicious signer can issue improperly formed signatures that will allow it to tell whether σ' was adapted from σ or issued anew. Luckily, the signer Alice can convince signature recipient Bob that σ' is properly formed via an efficient zero-knowledge proof protocol.

Even though we failed to meet the full-blown unforgeability and origin-hiding requirements of mercurial signatures, for the purpose of anonymous credentials, our results still constitute a success. This is because the protocol for issuing anonymous credentials typically requires that the recipient prove knowledge of his or her secret key anyway, so relaxing unforgeability to EUF-CoMA comes for free. Relaxing origin-hiding so it only holds when signatures were issued properly adds an additional step to the signing protocol — namely, a step where the signer convinces the recipient that the signature is properly formed — but this can be executed efficiently so is also a reasonable relaxation.

Our construction of a variable-length mercurial signature scheme uses the fixed-length mercurial signature scheme of [CL19] as a building block and is proven secure (under the variants of unforgeability and origin-hiding just outlined) assuming (1) the security of the underlying mercurial signature scheme and (2) the ABDDH$^+$ assumption, introduced in prior work [FHKS16] and reminiscent of the decisional Diffie-Hellman assumption for Type III bilinear pairings.

1.3 Technical Considerations

Origin-hiding of mercurial signatures. In addition to the $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ algorithms for a conventional signature scheme, a mercurial signature scheme includes two other algorithms: $\mathsf{ChangeRep}$ and $\mathsf{ConvertSig}$. $\mathsf{ChangeRep}$ takes in a public key pk, a message m, and a signature σ that verifies under pk and transforms them into an equivalent message m' and a signature σ' that verifies under the same public key pk. $\mathsf{ConvertSig}$ takes in (pk, m, σ) and outputs an equivalent m' and a signature σ' on m' that verifies under an equivalent pk'. These transformations are *origin-hiding*: given two equivalent messages, m_0 and m_1, and their signatures, σ_0 and σ_1, the distribution of (m_0', σ_0') produced by $\mathsf{ChangeRep}$ for $(\mathsf{pk}, m_0, \sigma_0)$ is statistically close to the distribution (m_1', σ_1') produced by

$\mathsf{ChangeRep}$ for $(\mathsf{pk}, m_1, \sigma_1)$. The same holds for $\mathsf{ConvertSig}$ for equivalent public keys pk_0 and pk_1.

It is important that the distributions be statistically close rather than simply indistinguishable. Recall that a mercurial signature is a building block for privacy-preserving applications, such as anonymous credentials and delegatable anonymous credentials. What makes it a good building block? It easy to prove both unforgeability and privacy of an anonymous credential.

Suppose that Alice is associated with an equivalence class of identifiers, of which m_A is a representative. A credential from the CA is a mercurial signature on one of the messages from her equivalence class. The unforgeability property of the mercurial signature scheme ensures that Alice cannot claim to possess a credential that in fact she does not.

The origin-hiding property is what protects Alice's privacy. It is important that the CA, who knows Alice by the identifier m_A and issued her the signature σ_A, cannot link her to (m'_A, σ'_A), where m'_A is another one of Alice's identifiers and σ'_A is the output of $\mathsf{ConvertSig}$. Let us see how origin-hiding prevents linkability.

The proof strategy is as follows. Imagine that we have some number of honest users in the system. In a real system, they would each use a different class of identifiers. However, due to the class-hiding property of the system, we can show via a hybrid argument that the adversary cannot distinguish the real system from one in which all honest users' identifiers come from the same equivalence class. At this point, if origin-hiding holds unconditionally, we have arrived at an experiment in which, given (m', σ'), even a computationally unbounded adversary cannot tell Alice from another user. However, if it holds only computationally, then an unbounded adversary could link σ' to (m_A, σ_A), and this proof strategy would be insufficient.

Towards constructing variable-length mercurial signatures. A naive approach to extending mercurial signatures to allow for messages of any length would be to hash the messages down to the correct fixed length and use the fixed-length mercurial signature scheme. 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 relationship between messages, we instead break a message $m = (\hat{g}, u_1, \ldots, u_n)$, where $\hat{g}$ is the base group element and, for all $1 \leq i \leq n$, $u_i = \hat{g}^{m_i}$ for some $m_i \in \mathbb{Z}_p^*$, into its n constituent group elements u_i. We then sign each one individually, together with powers of a base group element indicating its index i, using the fixed-length mercurial signature scheme. This does not suffice, though, because an adversary may be able to mix and match elements of the n new messages M_i being signed under the fixed-length scheme. In order to mitigate this, an additional group element, which we call a "glue" element, is included in each of the n messages M_i to link them together and to the original message m in an unforgeable way. Specifically, we represent the message m to be signed as a sequence of n fixed-length 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 a change of message representative from m to $m' = m^\mu$, for any $\mu \in \mathbb{Z}_p^*$, by simply changing each M_i to $M_i' = M_i^\mu$ and invoking the $\mathsf{ChangeRep}$ algorithm of the underlying fixed-length scheme. The problem with this approach, however, is that different signatures will receive different glue values, so origin-hiding will not hold in a statistical sense. It is important for the origin-hiding property that the glue element $\tilde{g}^s$ for a message m be a function of the discrete logarithms m_i so that if an equivalent 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' is equivalent to M_i for all $1 \leq i \leq n$). Computing $\tilde{g}^s$ 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 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.

Interactive signing protocol. Unfortunately, we are not able to achieve a signing *algorithm* that, on input a message vector, computes a group element that can act as the glue. Instead, we provide an interactive signing protocol that the signer and the signature recipient run together to compute the glue element. In this protocol, for a message $m = (\hat{g}, u_1, \ldots, u_n)$, the recipient of the signature (but not the signer!) must know the discrete logarithms $m_i \in \mathbb{Z}_p^*$, where $u_i = \hat{g}^{m_i}$ for $1 \leq i \leq n$. This, in turn, leads to a relaxation of the notion of unforgeability [FG18], whereby the adversary's query to the signing oracle on a message m must be paired with its discrete logarithms. Even though interactivity is a shortcoming of our result, is sufficient for the motivating application: when a user certifies another user's public key, the credential recipient must prove, via a ZKPoK, that she knows her corresponding secret key. Thus, interaction in signing is commensurate with the ZKPoK already required for anonymous credentials.

2 Definition of Mercurial Signatures

We propose a definition of mercurial signatures that is nearly identical to the definition stated in [CL19], but allows for messages of unbounded length. As we shall see, the original construction of mercurial signatures satisfies this revised definition for a fixed-length message space. Though mostly a restatement, we provide the definition here to highlight the ways in which it differs from the original. Unlike the prior scheme [CL19], the message space $\mathcal{M}$ for our new mercurial signature scheme consists of vectors of any length, and we denote by $\mathcal{M}_n$ the message space consisting of all message vectors of length n. Additionally, the KeyGen algorithm no longer takes a length parameter as input, and $\mathsf{ConvertSig}$ now takes a message converter μ as input in order to convert (m, σ) into $(\tilde{m}, \tilde{\sigma})$.

Definition 1 (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 otherwise stated:

$\mathsf{PPGen}(1^k) \to PP$: 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 2 (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 $PP \in \mathsf{PPGen}(1^k)$, 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 [CL19]. 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 new signature $\tilde{\sigma}$ is valid on the new, equivalent message m' under the new public key $\tilde{\mathsf{pk}}$.

Definition 3 (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:

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

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

This unforgeability property is similar to existential unforgeability under chosen open message attack (EUF-CoMA), defined by Fuchsbauer and Gay [FG18]. EUF-CoMA differs from EUF-CMA in that the adversary must provide the discrete logarithms $\bar{m}$ of the message m to be signed, which has the advantage that the adversary's success is efficiently verifiable. Our notion of unforgeability is similar to EUF-CoMA, except the adversary's winning condition is different. As in the EUF-CoMA game, the adversary is given the public key pk and is allowed to issue signature queries to the 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 choose 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, making the adversary's task harder. EUF-CoMA restricts the forged message m^* in this way, but does not allow a forgery under an equivalent public key. The definition in [CL19] allows a forgery under an equivalent public key, but does not require the adversary to provide the discrete logarithms of the message to be signed.

Definition 4 (Class- and origin-hiding). 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 *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:

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

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

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

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 $[\mathsf{pk}^*]_{\mathcal{R}_{\mathsf{pk}}}$.

Note that this definition of origin-hiding is a relaxation of the prior definition [CL19] 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 3.1.

3 Construction of Mercurial Signatures for Variable-Length Messages

Let $\mathbb{G}_1, \mathbb{G}_2$, and $\mathbb{G}_T$ be multiplicative groups of prime order p with a Type III bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ (see Appendix A). Similar to the prior mercurial signature scheme [CL19], 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 $1 \leq i \leq n$, $u_i = \hat{g}^{m_i}$ 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 of interest to us are as follows:

$$\mathcal{R}_m = \{(m, m') \in (\mathbb{G}_1^*)^{n+1} \times (\mathbb{G}_1^*)^{n+1} \mid \exists\, \mu \in \mathbb{Z}_p^* \text{ such that } m' = m^{\mu}\}$$
$$\mathcal{R}_{\mathsf{sk}} = \{(\mathsf{sk}_{\mathsf{X}}, \tilde{\mathsf{sk}}_{\mathsf{X}}) \in (\mathbb{Z}_p^*)^{10} \times (\mathbb{Z}_p^*)^{10} \mid \exists\, \rho \in \mathbb{Z}_p^* \text{ such that } \tilde{\mathsf{sk}} = \rho \cdot \mathsf{sk}\}$$
$$\mathcal{R}_{\mathsf{pk}} = \{(\mathsf{pk}_{\mathsf{X}}, \tilde{\mathsf{pk}}_{\mathsf{X}}) \in (\mathbb{G}_2^*)^{10} \times (\mathbb{G}_2^*)^{10} \mid \exists\, \rho \in \mathbb{Z}_p^* \text{ such that } \tilde{\mathsf{pk}} = \mathsf{pk}^{\rho}\}$$

Our variable-length mercurial signature scheme, denoted MS_{X}, is an extension of the prior fixed-length scheme, denoted MS_f [CL19], which can be found in Appendix B. The subscript X, for extension, is used to denote all keys and algorithms associated with the variable-length scheme MS_{X}.

Let us discuss the security properties of the fixed-length scheme MS_f. It satisfies the definition of security in Section 2, 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 2 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. [CL19]. *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.*

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 $1 \leq i \leq n$). 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: $p_m(x) = m_1 + m_2x + m_3x^2 \cdots + m_nx^{n-1}$. 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:

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

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 3.1) involves an interactive signing protocol carried out between the signer and the recipient of the signature.

Construction. The following algorithms from the fixed-length scheme MS_f are invoked: $\mathsf{ChangeRep}_f(\mathsf{pk}, m, \sigma, \mu) \to (m', \sigma')$, 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 $\mathsf{ConvertSig}_f(\mathsf{pk}, m, \sigma, \rho) \to \tilde{\sigma}$, 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 $\rho \in \mathbb{Z}_p^*$.

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

$\mathsf{KeyGen}_\mathsf{X}(PP_\mathsf{X}) \to (\mathsf{pk}_\mathsf{X}, \mathsf{sk}_\mathsf{X})$: 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 $\mathsf{sk}_\mathsf{X} = (\mathsf{sk}, x_6, x_7, x_8, x_9, x_{10})$ and $\mathsf{pk}_\mathsf{X} = (\mathsf{pk}, \hat{X}_6, \hat{X}_7, \hat{X}_8, \hat{X}_9, \hat{X}_{10})$, where $\hat{X}_i = \hat{P}^{x_i}$, and output $(\mathsf{pk}_\mathsf{X}, \mathsf{sk}_\mathsf{X})$.

$\mathsf{Sign}_\mathsf{X}(\mathsf{sk}_\mathsf{X}, m) \rightarrow (\hat{h}, \sigma)$: On input $\mathsf{sk}_\mathsf{X} = (\mathsf{sk}, x_6, x_7, x_8, x_9, x_{10})$ 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:

$$\hat{h} = \Big(\prod_{i=1}^{n} u_i^{\hat{x}^{i-1}}\Big)^y$$

Compute $\hat{g}^2, \ldots, \hat{g}^n$. For all $1 \leq i \leq n$, 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 $(\hat{h}, \sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_n\})$.

$\mathsf{Verify}_\mathsf{X}(\mathsf{pk}_\mathsf{X}, m, (\hat{h}, \sigma)) \rightarrow 0/1$: On input $\mathsf{pk}_\mathsf{X} = (\mathsf{pk}, \hat{X}_6, \hat{X}_7, \hat{X}_8, \hat{X}_9, \hat{X}_{10})$, $m = (\hat{g}, u_1, \ldots, u_n)$, and a signature $(\hat{h}, \sigma = \{\sigma_1, \ldots, \sigma_n\})$, compute $\hat{g}^2, \ldots, \hat{g}^n$. For all $1 \leq i \leq n$, 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.

$\mathsf{ConvertSK}_\mathsf{X}(\mathsf{sk}_\mathsf{X}, \rho) \rightarrow \tilde{\mathsf{sk}}_\mathsf{X}$: On input $\mathsf{sk}_\mathsf{X} = (\mathsf{sk}, x_6, x_7, x_8, x_9, x_{10})$ 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 $6 \leq i \leq 10$, and output the new secret key $\tilde{\mathsf{sk}}_\mathsf{X} = (\tilde{\mathsf{sk}}, \tilde{x}_6, \tilde{x}_7, \tilde{x}_8, \tilde{x}_9, \tilde{x}_{10})$.

$\mathsf{ConvertPK}_\mathsf{X}(\mathsf{pk}_\mathsf{X}, \rho) \rightarrow \tilde{\mathsf{pk}}_\mathsf{X}$: On input $\mathsf{pk}_\mathsf{X} = (\mathsf{pk}, \hat{X}_6, \hat{X}_7, \hat{X}_8, \hat{X}_9, \hat{X}_{10})$ 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 $6 \leq i \leq 10$, and output the new public key $\tilde{\mathsf{pk}}_\mathsf{X} = (\tilde{\mathsf{pk}}, \tilde{X}_6, \tilde{X}_7, \tilde{X}_8, \tilde{X}_9, \tilde{X}_{10})$.

$\mathsf{ChangeRep}_\mathsf{X}(\mathsf{pk}_\mathsf{X}, m, (\hat{h}, \sigma), \mu) \rightarrow (m', (\hat{h}', \sigma'))$: On input $\mathsf{pk}_\mathsf{X} = (\mathsf{pk}, \hat{X}_6, \hat{X}_7, \hat{X}_8, \hat{X}_9, \hat{X}_{10})$, $m = (\hat{g}, u_1, \ldots, u_n)$, signature $(\hat{h}, \sigma = \{\sigma_1, \ldots, \sigma_n\})$, and $\mu \in \mathbb{Z}_p^*$, compute $\hat{g}^2, \ldots, \hat{g}^n$. For all $1 \leq i \leq n$, 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 $(m', (\hat{h}', \sigma' = \{\sigma_1', \ldots, \sigma_n'\}))$.

$\mathsf{ConvertSig}_\mathsf{X}(\mathsf{pk}_\mathsf{X}, m, (\hat{h}, \sigma), \rho, \mu) \rightarrow (m', (\hat{h}', \tilde{\sigma}))$: On input pk_X, m, signature $(\hat{h}, \sigma)$, and $\rho, \mu \in \mathbb{Z}_p^*$, run $(m', (\hat{h}', \sigma')) \leftarrow \mathsf{ChangeRep}_\mathsf{X}(\mathsf{pk}_\mathsf{X}, m, (\hat{h}, \sigma), \mu)$, 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 $1 \leq i \leq n$, 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 $(m', (\hat{h}', \tilde{\sigma} = \{\tilde{\sigma}_1, \ldots, \tilde{\sigma}_n\}))$.

3.1 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 $1 \leq i \leq n$, 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 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 3.

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 being 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 via a ZKPoK that the glue element was indeed computed correctly, so origin-hiding holds for all signers, not just honest ones.

Let us now address which ZKPoK protocol ought to be used. 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 [Can01], but it may come at an efficiency cost. For efficient and UC-secure Sigma protocols [Dam02], Dodis, Shoup, and Walfish [DSW08] offer a solution, but it relies on verifiable encryption [CS03] or similar, which adds complexity and setup assumptions. In the random oracle model, Fischlin [Fis05] as well as Bernhard, Fischlin, and Warinschi [BFW15] 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.

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.

$[\mathsf{Sign}_\mathsf{X}(\mathsf{sk}_\mathsf{X}, m) \leftrightarrow \mathsf{Receive}_\mathsf{X}(\mathsf{pk}_\mathsf{X}, m, (m_1, \ldots, m_n))] \rightarrow (\tilde{m}, (\tilde{h}, \sigma))$: The Signer takes as input his signing key $\mathsf{sk}_\mathsf{X} = (\mathsf{sk}, x_6, x_7, x_8, x_9, x_{10})$ and a message $m = (\hat{g}, u_1, \ldots, u_n)$. The Receiver takes as input the corresponding public key $\mathsf{pk}_\mathsf{X} = (\mathsf{pk}, \hat{X}_6, \hat{X}_7, \hat{X}_8, \hat{X}_9, \hat{X}_{10})$, 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 $1 \leq i \leq n$.
1. The Signer acts as the verifier while the Receiver 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, 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 $\tilde{m} = (\tilde{g}, \tilde{u}_1, \dots, \tilde{u}_n) = (\hat{g}^w, u_i^w, \dots, u_n^w)$. He also computes $\tilde{g}^2, \dots, \tilde{g}^n$. For all $1 \leq i \leq n$, 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 $(\tilde{h}, \sigma = \{\sigma_1, \dots, \sigma_n\})$ 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 $\tilde{m} = (\tilde{g}, \tilde{u}_1, \dots, \tilde{u}_n)$:

$\mathsf{Verify_X}(\mathsf{pk_X}, \tilde{m}, (\tilde{h}, \sigma)) \rightarrow 0/1$: 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 $1 \leq i \leq n$.

$\mathsf{ChangeRep_X}(\mathsf{pk_X}, \tilde{m}, (\tilde{h}, \sigma), \mu) \rightarrow (\tilde{m}', (\tilde{h}', \sigma'))$: 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 $1 \leq i \leq n$. Output $(\tilde{m}' = (\tilde{g}^\mu, \tilde{u}_1^\mu, \dots, \tilde{u}_n^\mu), (\tilde{h}' = \tilde{h}^\mu, \sigma' = \{\sigma_1', \dots, \sigma_n'\}))$.

$\mathsf{ConvertSig_X}(\mathsf{pk_X}, \tilde{m}, (\tilde{h}, \sigma), \rho, \mu) \rightarrow (\tilde{m}', (\tilde{h}', \tilde{\sigma}))$: Run $(\tilde{m}', (\tilde{h}', \sigma' = \{\sigma_1', \dots, \sigma_n'\})) \leftarrow \mathsf{ChangeRep_X}(\mathsf{pk_X}, \tilde{m}, (\tilde{h}, \sigma), \mu)$. 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 $1 \leq i \leq n$. Output $(\tilde{m}', (\tilde{h}', \tilde{\sigma} = \{\tilde{\sigma}_1, \dots, \tilde{\sigma}_n\}))$.

Remark. While the elements $\hat{X}_6, \hat{X}_7, \hat{X}_8, \hat{X}_9, \hat{X}_{10}$ of the public key $\mathsf{pk_X}$ do not participate in signature verification, they do participate in Step 3 of the signing protocol.

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 $(\tilde{m}, (\tilde{h}, \sigma = \{\sigma_1, \dots, \sigma_n\}))$ 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 $1 \leq i \leq n$, $\mathsf{ChangeRep}_f(\mathsf{pk}, M_i, \sigma_i, \mu)$ 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 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 $1 \leq i \leq n$, which implies that $\mathsf{Verify_X}(\mathsf{pk_X}, \tilde{m}', (\tilde{h}', \sigma')) = 1$, where $\tilde{m}' = (\tilde{g}^\mu, \tilde{u}_1^\mu, \dots, \tilde{u}_n^\mu) \in [\tilde{m}]_{\mathcal{R}_m}$.

3.2 Origin-hiding

Theorem 3 (Origin-hiding). *Let MS_f be a mercurial signature scheme on message space $(\mathbb{G}_1^*)^5$ as in Theorem 1, and let 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 3.1, where the proof system used in Step 3 is sound under sequential (or concurrent) composition. Then, MS_X is origin-hiding under sequential (or concurrent) composition.*

Origin-hiding of $\mathsf{ChangeRep}_\mathsf{X}$: Let $\mathsf{pk}_\mathsf{X}^*, \tilde{m}, (\tilde{h}, \sigma = \{\sigma_1, \dots, \sigma_n\})$ be such that $\mathsf{Verify}_\mathsf{X}(\mathsf{pk}_\mathsf{X}^*, \tilde{m}, (\tilde{h}, \sigma)) = 1$, where pk_X^* is possibly adversarially generated. $\mathsf{ChangeRep}_\mathsf{X}(\mathsf{pk}_\mathsf{X}^*, \tilde{m}, (\tilde{h}, \sigma), \mu)$ outputs

$$(\tilde{m}', (\tilde{h}', \sigma')) = (\tilde{m}^\mu, (\tilde{h}^\mu, \{\sigma_1', \dots, \sigma_n'\}))$$

where $\tilde{m}^\mu$ is shorthand for $\tilde{m}^\mu = (\tilde{g}^\mu, \tilde{u}_1^\mu, \dots, \tilde{u}_n^\mu)$. 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 $M_i = (\tilde{g}, \tilde{g}^i, \tilde{g}^n, \tilde{h}, \tilde{u}_i)$. $\mathsf{ChangeRep}_\mathsf{X}$ 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 origin-hiding of $\mathsf{ChangeRep}_f$ (Theorem 1), σ_i' is distributed the same as a fresh signature on M_i' for all $1 \le i \le n$. 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 $(\tilde{h}^\mu, (\sigma_1', \dots, \sigma_n'))$ is a uniformly random element in the space of signatures $(\bar{h}, \bar{\sigma})$ satisfying $\mathsf{Verify}_\mathsf{X}(\mathsf{pk}_\mathsf{X}^*, \tilde{m}^\mu, (\bar{h}, \bar{\sigma})) = 1$ with overwhelming probability.

Origin-hiding of $\mathsf{ConvertSig}_\mathsf{X}$: Let $\mathsf{pk}_\mathsf{X}^*, \tilde{m}, (\tilde{h}, \sigma = \{\sigma_1, \dots, \sigma_n\})$ be such that $\mathsf{Verify}_\mathsf{X}(\mathsf{pk}_\mathsf{X}^*, \tilde{m}, (\tilde{h}, \sigma)) = 1$, where pk_X^* is possibly adversarially generated. $\mathsf{ConvertSig}_\mathsf{X}(\mathsf{pk}_\mathsf{X}^*, \tilde{m}, (\tilde{h}, \sigma), \rho, \mu)$ outputs

$$(\tilde{m}', (\tilde{h}', \tilde{\sigma})) = (\tilde{m}^\mu, (\tilde{h}^\mu, (\tilde{\sigma}_1, \dots, \tilde{\sigma}_n)))$$

where $\tilde{m}^\mu$ is shorthand for $\tilde{m}^\mu = (\tilde{g}^\mu, \tilde{u}_1^\mu, \dots, \tilde{u}_n^\mu)$. 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 $\mathsf{ConvertSig}_\mathsf{X}$ is computed in two steps. First, $\mathsf{ChangeRep}_\mathsf{X}(\mathsf{pk}_\mathsf{X}^*, \tilde{m}, (\tilde{h}, \sigma), \mu)$ outputs $(\tilde{m}', (\tilde{h}', \sigma')) = (\tilde{m}^\mu, (\tilde{h}^\mu, (\sigma_1', \dots, \sigma_n')))$. Then, $\mathsf{ConvertSig}_f(\mathsf{pk}^*, M_i', \sigma_i', \rho)$ outputs $\tilde{\sigma}_i$ for all $1 \le i \le n$. $\mathsf{ChangeRep}_\mathsf{X}$ 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 $(\tilde{h}^\mu, (\tilde{\sigma}_1, \dots, \tilde{\sigma}_n))$ is a uniformly random element in the space of signatures $(\bar{h}, \bar{\sigma})$ satisfying $\mathsf{Verify}_\mathsf{X}(\mathsf{ConvertPK}_\mathsf{X}(\mathsf{pk}_\mathsf{X}^*, \rho), \tilde{m}^\mu, (\bar{h}, \bar{\sigma})) = 1$ with overwhelming probability (where $\mathsf{ConvertPK}_\mathsf{X}(\mathsf{pk}_\mathsf{X}^*, \rho) = (\mathsf{pk}_\mathsf{X}^*)^\rho$ is a uniformly random element of $[\mathsf{pk}_\mathsf{X}^*]_{\mathcal{R}_\mathsf{pk}}$).

3.3 Unforgeability

Unforgeability of $\mathsf{MS_X}$ holds under a variant of the decisional Diffie-Hellman assumption for Type III bilinear pairings, called the ABDDH$^+$ assumption. It was introduced by Fuchsbauer, Hanser, Kamath, and Slamanig [FHKS16] and proven to hold in generic groups.

Definition 5 (ABDDH$^+$ assumption). [FHKS16] 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:

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

Proposition 1. [FHKS16] *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 3.1, 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 $\mathsf{pk_X} = (\mathsf{pk}, \hat{X}_6, \hat{X}_7, \hat{X}_8, \hat{X}_9, \hat{X}_{10})$, where $\hat{X}_i = \hat{P}^{x_i}$. $\mathcal{B}_{\mathsf{MS}_f}$ forwards $PP_{\mathsf{X}} = PP$ and $\mathsf{pk_X}$ 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 $1 \leq i \leq n$, 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:

$$\hat{h} = \Big(\prod_{i=1}^{n} u_i^{\hat{x}^{i-1}}\Big)^y$$

$\mathcal{B}_{\mathsf{MS}_f}$ 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 $1 \leq i \leq n$. 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:

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

$\mathcal{B}_{\mathsf{MS}_f}$ 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 queries m that $\mathcal{A}$ has issued 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 $\mathcal{A}$: either (1) 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, or (2) 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.

(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. The full proof can be found in Appendix C.

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

$\updownarrow$ Claim 3: ABDDH$^+$ assumption in $\mathbb{G}_1$ (hybrid argument)

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

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

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

$\updownarrow$ Claim 6: polynomial collision argument

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

Thus, $\mathcal{A}$ cannot mix and match when real glue elements are used, except with negligible probability. With overwhelming probability, the Good Case occurs and $\mathcal{A}$'s non-negligible success in producing a forgery for MS_X becomes $\mathcal{B}_{\mathsf{MS}_f}$'s non-negligible success in producing a forgery for MS_f. $\square$

3.4 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 $\mathsf{pk}_{\mathsf{X},2} \in [\mathsf{pk}_{\mathsf{X},1}]_{\mathcal{R}_{\mathsf{pk}}}$. The proof of message class-hiding is a straightforward hybrid argument under the decisional Diffie-Hellman assumption (DDH) [FHS19]. The bulk of this section is devoted to proving public key class-hiding.

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 MS_X be the variable-length mercurial signature scheme on message space $(\mathbb{G}_1^*)^{n+1}$ constructed above. Then, message class-hiding of 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.*

Proof. This is analogous to the proof of message class-hiding for MS_f, which was inherited from the work of Fuchsbauer et al. [FHS19]. $\square$

Theorem 6 (Public key class-hiding). *Let MS_f be a mercurial signature scheme on message space $(\mathbb{G}_1^*)^5$, and let 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 3.1, where the proof system used in Step 1 is extractable under sequential (or concurrent) composition. Then, public key class-hiding of 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.*

Proof. Consider two public keys for the mercurial signature scheme 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 sample 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). Our goal is to 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 each of these three types as well as glue elements that are computed in the various ways specified in the proof of unforgeability. The overall proof structure is as follows. Arrows indicate why consecutive games are indistinguishable. The full proof can be found in Appendix D.

Game 0. $\mathsf{pk}_{\mathsf{X},1}, \mathsf{pk}_{\mathsf{X},2}$ are independent, $\tilde{h} = \tilde{g}^{y^{(\delta)}\cdot q_\delta}$ for $\delta \in \{1,2\}$. No extraction or ZK simulation. This is the real signing game.

$\updownarrow$ Claim 1: zero-knowledge property, same as unforgeability Claim 1

Game 1. $\mathsf{pk}_{\mathsf{X},1}, \mathsf{pk}_{\mathsf{X},2}$ are independent, $\tilde{h} = \tilde{g}^{y^{(\delta)}\cdot q_\delta}$ for $\delta \in \{1,2\}$. No extraction, but simulation of ZKPoK of glue $\tilde{h}$.

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

Game 2. $\mathsf{pk}_{\mathsf{X},1}, \mathsf{pk}_{\mathsf{X},2}$ are independent, $\tilde{h} = \tilde{g}^{y^{(\delta)} \cdot q_\delta}$ for $\delta \in \{1, 2\}$. Extraction of the m_i's and simulation of ZKPoK of glue $\tilde{h}$ in this and subsequent games.

$\updownarrow$ Claim 3: reduction to public key class-hiding of MS_f

Game 3. $\mathsf{pk}_{\mathsf{X},1}, \mathsf{pk}_{\mathsf{X},2}$ are 1/2 independent 1/2 equivalent, $\tilde{h} = \tilde{g}^{y^{(\delta)} \cdot q_\delta}$.

$\updownarrow$ Claim 4: ABDDH$^+$ assumption in $\mathbb{G}_1$, similar to unforgeability Claim 3

Game 4. $\mathsf{pk}_{\mathsf{X},1}, \mathsf{pk}_{\mathsf{X},2}$ are 1/2 independent 1/2 equivalent, $\tilde{h} = \tilde{g}^{R_\delta(q_\delta)}$, where $R_\delta : \mathbb{Z}_p^* \rightarrow \mathbb{Z}_p^*$ is a random function for $\delta \in \{1, 2\}$.

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

Game 5. $\mathsf{pk}_{\mathsf{X},1}, \mathsf{pk}_{\mathsf{X},2}$ are 1/2 independent 1/2 equivalent, $\tilde{h} = \tilde{g}^{R_\delta(\dot{q}_\delta)}$, where $\dot{q}_\delta = p(\alpha_\delta)$ for a "fake" secret point α_δ, $\delta \in \{1, 2\}$.

$\updownarrow$ Claim 6: polynomial collision argument, similar to unforgeability Claim 6

Game 6. $\mathsf{pk}_{\mathsf{X},1}, \mathsf{pk}_{\mathsf{X},2}$ are 1/2 independent 1/2 equivalent, $\tilde{h} = \tilde{g}^{R_\delta(m_1,\ldots,m_n)}$, where $R_\delta : (\mathbb{Z}_p^*)^n \rightarrow \mathbb{Z}_p^*$ is a random function for $\delta \in \{1, 2\}$.

$\updownarrow$ Claim 7: DDH assumption in $\mathbb{G}_1$

Game 7. $\mathsf{pk}_{\mathsf{X},1}, \mathsf{pk}_{\mathsf{X},2}$ are 1/2 independent 1/2 equivalent, $\tilde{h} = \tilde{g}^{R(m_1,\ldots,m_n)}$, where $R : (\mathbb{Z}_p^*)^n \rightarrow \mathbb{Z}_p^*$ is a random function.

$\updownarrow$ **Intermediate Games**: Claim 8: 5 $\times$ DDH assumption in $\mathbb{G}_2$

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 Claim 4 and Claim 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$^+$ assumption, 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: zero-knowlege property, same as unforgeability 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.

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), MS_X satisfies public key class-hiding. $\square$

References

[BFW15] David Bernhard, Marc Fischlin, and Bogdan Warinschi. Adaptive proofs of knowledge in the random oracle model. In Jonathan Katz, editor, *PKC 2015*, volume 9020 of *LNCS*, pages 629–649. Springer, Heidelberg, March / April 2015.

[BHKS18] Michael Backes, Lucjan Hanzlik, Kamil Kluczniak, and Jonas Schneider. Signatures with flexible public key: A unified approach to privacy-preserving signatures (full version). Cryptology ePrint Archive, Report 2018/191, 2018. `https://eprint.iacr.org/2018/191`.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, October 2001.

[CDHK15] Jan Camenisch, Maria Dubovitskaya, Kristiyan Haralambiev, and Markulf Kohlweiss. Composable and modular anonymous credentials: Definitions and practical constructions. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015, Part II*, volume 9453 of *LNCS*, pages 262–288. Springer, Heidelberg, November / December 2015.

[Cha86] David Chaum. Showing credentials without identification: Signatures transferred between unconditionally unlinkable pseudonyms. In Franz Pichler, editor, *EUROCRYPT'85*, volume 219 of *LNCS*, pages 241–244. Springer, Heidelberg, April 1986.

[CKL$^+$14] Jan Camenisch, Stephan Krenn, Anja Lehmann, Gert Læssøe Mikkelsen, Gregory Neven, and Michael Østergaard Pedersen. Formal treatment of privacy-enhancing credential systems. Cryptology ePrint Archive, Report 2014/708, 2014. `http://eprint.iacr.org/2014/708`.

[CL01] Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 93–118. Springer, Heidelberg, May 2001.

[CL04] Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 56–72. Springer, Heidelberg, August 2004.

[CL19] Elizabeth C. Crites and Anna Lysyanskaya. Delegatable anonymous credentials from mercurial signatures. In Mitsuru Matsui, editor, *CT-RSA 2019*, volume 11405 of *LNCS*, pages 535–555. Springer, Heidelberg, March 2019.

[CS03] Jan Camenisch and Victor Shoup. Practical verifiable encryption and decryption of discrete logarithms. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 126–144. Springer, Heidelberg, August 2003.

[Dam02] Ivan Damgård. On σ-protocols. Available at `http://www.daimi.au.dk/~ivan/Sigma.ps`, 2002.

[DSW08] Yevgeniy Dodis, Victor Shoup, and Shabsi Walfish. Efficient constructions of composable commitments and zero-knowledge proofs. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 515–535. Springer, Heidelberg, August 2008.

[FG18] Georg Fuchsbauer and Romain Gay. Weakly secure equivalence-class signatures from standard assumptions. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018, Part II*, volume 10770 of *LNCS*, pages 153–183. Springer, Heidelberg, March 2018.

[FHKS16] Georg Fuchsbauer, Christian Hanser, Chethan Kamath, and Daniel Slamanig. Practical round-optimal blind signatures in the standard model from weaker assumptions. In Vassilis Zikas and Roberto De Prisco, editors, *SCN 16*, volume 9841 of *LNCS*, pages 391–408. Springer, Heidelberg, August / September 2016.

[FHS19] Georg Fuchsbauer, Christian Hanser, and Daniel Slamanig. Structure-preserving signatures on equivalence classes and constant-size anonymous credentials. *Journal of Cryptology*, 32(2):498–546, April 2019.

[Fis05] Marc Fischlin. Communication-efficient non-interactive proofs of knowledge with online extractors. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 152–168. Springer, Heidelberg, August 2005.

[LRSW99] Anna Lysyanskaya, Ronald L. Rivest, Amit Sahai, and Stefan Wolf. Pseudonym systems. In Howard M. Heys and Carlisle M. Adams, editors, *SAC 1999*, volume 1758 of *LNCS*, pages 184–199. Springer, Heidelberg, August 1999.

[Lys02] Anna Lysyanskaya. *Signature schemes and applications to cryptographic protocol design.* PhD thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, September 2002.

A Preliminaries

Definition 6 (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:

Bilinearity: $e(P^a, \hat{P}^b) = e(P, \hat{P})^{ab} = e(P^b, \hat{P}^a) \ \forall\ a, b \in \mathbb{Z}_p$.
Non-degeneracy: $e(P, \hat{P}) \neq 1_{\mathbb{G}_T}$ (i.e., $e(P, \hat{P})$ generates $\mathbb{G}_T$).
Computability: There exists an efficient algorithm to compute e.

Bilinear pairings can be classified into three types:

Type I (symmetric): $\mathbb{G}_1 = \mathbb{G}_2$.
Type II (asymmetric): $\mathbb{G}_1 \neq \mathbb{G}_2$, but there exists an efficiently computable homomorphism $\phi : \mathbb{G}_2 \to \mathbb{G}_1$ (none in the reverse direction).
Type III (asymmetric): $\mathbb{G}_1 \neq \mathbb{G}_2$, but there exists no efficiently computable homomorphism in either direction.

Definition 7 (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 $\mathsf{BG} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, P, \hat{P}, e)$ consisting of groups $\mathbb{G}_1 = \langle P \rangle$, $\mathbb{G}_2 = \langle \hat{P} \rangle$, and $\mathbb{G}_T$ of prime order p with $\log_2 p = \lceil k \rceil$ and a Type III pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$.

Definition 8 (Discrete logarithm assumption (DL)). Let BGGen be a bilinear group generator that outputs $\mathsf{BG} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, P_1 = P, P_2 = \hat{P}, e)$. 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 9 (Decisional Diffie-Hellman assumption (DDH)). Let BGGen be a bilinear group generator that outputs $\mathsf{BG} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, P_1 = P, P_2 = \hat{P}, e)$. 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:

$$\Pr[b \leftarrow \{0,1\}, \mathsf{BG} \leftarrow \mathsf{BGGen}(1^k), s, t, r \leftarrow \mathbb{Z}_p,$$
$$b^* \leftarrow \mathcal{A}(\mathsf{BG}, P_i^s, P_i^t, P_i^{(1-b)\cdot r + b \cdot st}) : b^* = b] - \frac{1}{2} \leq \nu(k)$$

B Prior Construction of Mercurial Signatures [CL19]

This is the only previously known construction of mercurial signatures [CL19]. 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:

$$\mathcal{R}_M = \{(M, M') \in (\mathbb{G}_1^*)^\ell \times (\mathbb{G}_1^*)^\ell \ \mid \ \exists\ \mu \in \mathbb{Z}_p^* \text{ such that } M' = M^\mu\}$$
$$\mathcal{R}_{\mathsf{sk}} = \{(\mathsf{sk}, \tilde{\mathsf{sk}}) \in (\mathbb{Z}_p^*)^\ell \times (\mathbb{Z}_p^*)^\ell \ \mid \ \exists\ \rho \in \mathbb{Z}_p^* \text{ such that } \tilde{\mathsf{sk}} = \rho \cdot \mathsf{sk}\}$$
$$\mathcal{R}_{\mathsf{pk}} = \{(\mathsf{pk}, \tilde{\mathsf{pk}}) \in (\mathbb{G}_2^*)^\ell \times (\mathbb{G}_2^*)^\ell \ \mid \ \exists\ \rho \in \mathbb{Z}_p^* \text{ such that } \tilde{\mathsf{pk}} = \mathsf{pk}^\rho\}$$

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:

$\mathsf{PPGen}(1^k) \rightarrow PP$: Compute $\mathsf{BG} \leftarrow \mathsf{BGGen}(1^k)$. Output $PP = \mathsf{BG} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, P, \hat{P}, e)$. Now that BG is well-defined, the relations $\mathcal{R}_M$, $\mathcal{R}_{\mathsf{pk}}$, $\mathcal{R}_{\mathsf{sk}}$ are also well-defined. sample_ρ and sample_μ are the same algorithm, namely the one that samples a random element of $\mathbb{Z}_p^*$.

$\mathsf{KeyGen}(PP, \ell) \rightarrow (\mathsf{pk}, \mathsf{sk})$: For $1 \leq i \leq \ell$, pick $x_i \leftarrow \mathbb{Z}_p^*$ and set secret key $\mathsf{sk} = (x_1, \ldots, x_\ell)$. Compute public key $\mathsf{pk} = (\hat{X}_1, \ldots, \hat{X}_\ell)$, where $\hat{X}_i = \hat{P}^{x_i}$ for $1 \leq i \leq \ell$. Output $(\mathsf{pk}, \mathsf{sk})$.

$\mathsf{Sign}(\mathsf{sk}, M) \rightarrow \sigma$: On input $\mathsf{sk} = (x_1, \ldots, x_\ell)$ and $M = (M_1, \ldots, M_\ell) \in (\mathbb{G}_1^*)^\ell$, pick a random $y \leftarrow \mathbb{Z}_p^*$ and 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 $\hat{Y} = \hat{P}^{\frac{1}{y}}$.

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

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

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

$\mathsf{ConvertSig}(\mathsf{pk}, M, \sigma, \rho) \to \tilde{\sigma}$: On input pk, message M, signature $\sigma = (Z, Y, \hat{Y})$, and 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').

C Proof of Unforgeability

Proof. (of Theorem 4.) As in Section 3.3, 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. $\tilde{h} = \tilde{g}^{y \cdot q}$, where $q = p(\hat{x})$. Extraction and simulation henceforth.

$\updownarrow$ Claim 3: ABDDH$^+$ assumption in $\mathbb{G}_1$ (hybrid argument)

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

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

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

$\updownarrow$ Claim 6: polynomial collision argument

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

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:

$$\hat{h} = \Big(\prod_{i=1}^{n} u_i^{\hat{x}^{i-1}}\Big)^y$$

$\mathcal{C}$ 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:

$$\tilde{h} = \Big(\prod_{i=1}^{n} (u_i^w)^{\hat{x}^{i-1}}\Big)^y$$

for a uniformly random $w \leftarrow \mathbb{Z}_p^*$. But note that:

$$\tilde{h} = \Big(\prod_{i=1}^{n} (\hat{g}^{m_i \cdot w})^{\hat{x}^{i-1}}\Big)^y = \Big(\prod_{i=1}^{n} \tilde{g}^{y \cdot m_i \cdot \hat{x}^{i-1}}\Big) = \tilde{g}^{y \cdot q}$$

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 (Asymmetric bilinear decisional Diffie-Hellman† problem (ABDDH†)). 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 *asymmetric bilinear decisional Diffie-Hellman† problem* in $\mathbb{G}_1$ is to distinguish between the distributions D_0 and D_1 defined by:

$$\begin{aligned} D_0 = \{&\mathsf{BG} \leftarrow \mathsf{BGGen}(1^k);\ \alpha, \beta, u, v, \omega \leftarrow \mathbb{Z}_p^*; \\ &(\mathsf{BG}, \hat{P}^{\alpha}, \hat{P}^{\alpha u}, \hat{P}^{\alpha v}, P^{\beta}, P^{\beta u v}, P^{\omega}, P^{\omega u v})\} \qquad (2) \\ D_1 = \{&\mathsf{BG} \leftarrow \mathsf{BGGen}(1^k);\ \alpha, \beta, u, v, \omega, r \leftarrow \mathbb{Z}_p^*; \\ &(\mathsf{BG}, \hat{P}^{\alpha}, \hat{P}^{\alpha u}, \hat{P}^{\alpha v}, P^{\beta}, P^{\beta u v}, P^{\omega}, P^{r})\} \qquad (3) \end{aligned}$$

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 $0 \le i \le \Gamma(k)$, 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 $1 \le i \le n$, 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 $j \le i$, $\mathcal{C}$ 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 defined by Equations (2) and (3).

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, WLOG 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 each 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 $R : \mathbb{Z}_p^* \rightarrow \mathbb{Z}_p^*$ 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.

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 MS_X (Theorem 4). $\square$

D Proof of Public Key Class-Hiding

Proof. (of Theorem 6.) As in Section 3.4, the overall proof structure is as follows. Arrows indicate why consecutive games are indistinguishable.

Game 0. $\mathsf{pk}_{\mathsf{X},1}, \mathsf{pk}_{\mathsf{X},2}$ are independent, $\tilde{h} = \tilde{g}^{y^{(\delta)} \cdot q_\delta}$ for $\delta \in \{1, 2\}$. No extraction or ZK simulation. This is the real signing game.

$\updownarrow$ Claim 1: zero-knowledge property, same as unforgeability Claim 1

Game 1. $\mathsf{pk}_{\mathsf{X},1}, \mathsf{pk}_{\mathsf{X},2}$ are independent, $\tilde{h} = \tilde{g}^{y^{(\delta)} \cdot q_\delta}$ for $\delta \in \{1, 2\}$. No extraction, but simulation of ZKPoK of glue $\tilde{h}$.

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

Game 2. $\mathsf{pk}_{\mathsf{X},1}, \mathsf{pk}_{\mathsf{X},2}$ are independent, $\tilde{h} = \tilde{g}^{y^{(\delta)} \cdot q_\delta}$ for $\delta \in \{1, 2\}$. Extraction of the m_i's and simulation of ZKPoK of glue $\tilde{h}$ in this and subsequent games.

$\updownarrow$ Claim 3: reduction to public key class-hiding of MS_f

Game 3. $\mathsf{pk}_{\mathsf{X},1}, \mathsf{pk}_{\mathsf{X},2}$ are 1/2 independent 1/2 equivalent, $\tilde{h} = \tilde{g}^{y^{(\delta)} \cdot q_\delta}$.

$\updownarrow$ Claim 4: ABDDH$^+$ assumption in $\mathbb{G}_1$, similar to unforgeability Claim 3

Game 4. $\mathsf{pk}_{\mathsf{X},1}, \mathsf{pk}_{\mathsf{X},2}$ are 1/2 independent 1/2 equivalent, $\tilde{h} = \tilde{g}^{R_\delta(q_\delta)}$, where $R_\delta : \mathbb{Z}_p^* \to \mathbb{Z}_p^*$ is a random function for $\delta \in \{1, 2\}$.

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

Game 5. $\mathsf{pk}_{\mathsf{X},1}, \mathsf{pk}_{\mathsf{X},2}$ are 1/2 independent 1/2 equivalent, $\tilde{h} = \tilde{g}^{R_\delta(\dot{q}_\delta)}$, where $\dot{q}_\delta = p(\alpha_\delta)$ for a "fake" secret point α_δ, $\delta \in \{1, 2\}$.

$\updownarrow$ Claim 6: polynomial collision argument, similar to unforgeability Claim 6

Game 6. $\mathsf{pk}_{\mathsf{X},1}, \mathsf{pk}_{\mathsf{X},2}$ are 1/2 independent 1/2 equivalent, $\tilde{h} = \tilde{g}^{R_\delta(m_1,\ldots,m_n)}$, where $R_\delta : (\mathbb{Z}_p^*)^n \to \mathbb{Z}_p^*$ is a random function for $\delta \in \{1, 2\}$.

$\updownarrow$ Claim 7: DDH assumption in $\mathbb{G}_1$

Game 7. $\mathsf{pk}_{\mathsf{X},1}, \mathsf{pk}_{\mathsf{X},2}$ are 1/2 independent 1/2 equivalent, $\tilde{h} = \tilde{g}^{R(m_1,\ldots,m_n)}$, where $R : (\mathbb{Z}_p^*)^n \to \mathbb{Z}_p^*$ is a random function.

$\updownarrow$ **Intermediate Games**: Claim 8: 5 $\times$ DDH assumption in $\mathbb{G}_2$

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 Claim 4 and Claim 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$^+$ assumption, 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: zero-knowlege property, same as unforgeability 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.

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

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:

$$x_{1,7} = x_{1,6} \cdot \hat{x}_1, \quad x_{1,9} = x_{1,8} \cdot y_1^{(1)} \quad x_{1,10} = x_{1,8} \cdot y_2^{(1)}$$
$$x_{2,7} = x_{2,6} \cdot \hat{x}_2, \quad x_{2,9} = x_{2,8} \cdot y_1^{(2)} \quad x_{2,10} = x_{2,8} \cdot y_2^{(2)}$$

$\mathcal{C}$ then sets:

$$\mathsf{pk}_{\mathsf{X},1} = (\mathsf{pk}_1, \hat{X}_{1,6}, \hat{X}_{1,7}, \hat{X}_{1,8}, \hat{X}_{1,9}, \hat{X}_{1,10}) \tag{4}$$
$$\mathsf{pk}_{\mathsf{X},2} = (\mathsf{pk}_2, \hat{X}_{2,6}, \hat{X}_{2,7}, \hat{X}_{2,8}, \hat{X}_{2,9}, \hat{X}_{2,10}) \tag{5}$$

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 $1 \leq i \leq n$, 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 $y^{(1)} = y_1^{(1)} \cdot y_2^{(1)}$ and $y^{(2)} = y_1^{(2)} \cdot y_2^{(2)}$ and:

$$\hat{h} = \Big(\prod_{i=1}^{n} u_i^{\hat{x}_\delta^{i-1}}\Big)^{y^{(\delta)}}$$

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 $1 \leq i \leq n$, 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. $\mathcal{C}$ 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:

$$\mathsf{pk}_{\mathsf{X},1} = (\mathsf{pk}_1, \hat{X}_{1,6}, \hat{X}_{1,7}, \hat{X}_{1,8}, \hat{X}_{1,9}, \hat{X}_{1,10})$$
$$\mathsf{pk}^b_{\mathsf{X},2} = (\mathsf{pk}^b_2, \hat{X}_{2,6}, \hat{X}_{2,7}, \hat{X}_{2,8}, \hat{X}_{2,9}, \hat{X}_{2,10})$$

where the $\hat{X}_{i,j}$'s are computed independently, as in Equations (4) and (5). $\mathcal{B}$ then forwards $PP_{\mathsf{X}} = PP$ and $\mathsf{pk}_{\mathsf{X},1}, \mathsf{pk}^b_{\mathsf{X},2}$ 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 $\mathsf{sk}^b_{\mathsf{X},2}$. $\mathcal{B}$ 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 $\mathsf{Sign}_f(\mathsf{sk}^b_2, \cdot)$, 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 $\mathsf{pk}^b_{\mathsf{X},2}$ is half in the same equivalence class as $\mathsf{pk}_{\mathsf{X},1}$ and half independent (Game 3) if and only if $\mathsf{pk}^b_2 \in [\mathsf{pk}_1]_{\mathcal{R}_{\mathsf{pk}}}$, 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:

$$\mathsf{pk}_{\mathsf{X},1} = (\mathsf{pk}_1, \hat{X}_{1,6}, \hat{X}_{1,7}, \hat{X}_{1,8}, \hat{X}_{1,9}, \hat{X}_{1,10})$$
$$\mathsf{pk}_{\mathsf{X},2} = (\mathsf{pk}^\beta_1, \hat{X}_{2,6}, \hat{X}_{2,7}, \hat{X}_{2,8}, \hat{X}_{2,9}, \hat{X}_{1,10})$$

where the $\hat{X}_{i,j}$'s are computed independently, as in Equations (4) and (5), and $\beta \leftarrow \mathbb{Z}_p^*$. $\mathcal{C}$ 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, as in Equations (4) and (5), and $\beta \leftarrow \mathbb{Z}_p^*$. $\mathcal{B}$ 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:

$$\tilde{g}_j^{(1)} = g_0 \qquad \tilde{h}_j^{(1)} = B \qquad \tilde{u}_{j,i}^{(1)} = g_0^{m_{j,i}} \ \ \forall i$$
$$\tilde{g}_j^{(2)} = A \qquad \tilde{h}_j^{(2)} = C \qquad \tilde{u}_{j,i}^{(2)} = A^{m_{j,i}} \ \ \forall i$$

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 $\hat{x}_2 = a = \hat{x}_1$ (Intermediate 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 $y_1^{(2)} = y_1^{(1)}$ (Intermediate Game 2). If $\hat{C} = \hat{g}_0^{r}$, then $y_1^{(1)}$ and $y_1^{(2)}$ are independent (Intermediate 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 $y_2^{(2)} = y_2^{(1)}$ (Intermediate Game 3). If $\hat{C} = \hat{g}_0^{r}$, then $y_1^{(1)}$ and $y_1^{(2)}$ are independent (Intermediate 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 $\lambda = a = \gamma$ (Intermediate 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 (Intermediate 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 (Intermediate Game 4).

Game 9. In this game, again $\mathsf{pk}_{\mathsf{X},2} \in [\mathsf{pk}_{\mathsf{X},1}]_{\mathcal{R}_{\mathsf{pk}}}$, but the glue element is $\tilde{h} = \tilde{g}^{R(\dot{q})}$, where $\dot{q} = p(\alpha)$.

Claim 9. *An adversary's view in Game 8 is the same as it is in Game 9, except with negligible probability.*

The is the same as Claim 6 (collision argument) in the proof of unforgeability.

Game 10. In this game, again $\mathsf{pk}_{\mathsf{X},2} \in [\mathsf{pk}_{\mathsf{X},1}]_{\mathcal{R}_{\mathsf{pk}}}$, but the glue element is $\tilde{h} = \tilde{g}^{R(q)}$, where $q = p(\hat{x})$.

Claim 10. *A PPT adversary cannot distinguish Game 9 from Game 10 under the DL assumption in* $\mathbb{G}_2$.

This is the same as Claim 4 (collision argument) and Claim 5 (DL) in the proof of unforgeability.

Game 11. In this game, again $\mathsf{pk}_{\mathsf{X},2} \in [\mathsf{pk}_{\mathsf{X},1}]_{\mathcal{R}_{\mathsf{pk}}}$, but the glue element is $\tilde{h} = \tilde{g}^{y \cdot q}$.

Claim 11. *A PPT adversary cannot distinguish Game 10 from Game 11 under the ABDDH$^+$ assumption in* $\mathbb{G}_1$.

This is the same as Claim 3 (ABDDH$^+$) in the proof of unforgeability.

Game 12. In this game, again $\mathsf{pk}_{\mathsf{X},2} \in [\mathsf{pk}_{\mathsf{X},1}]_{\mathcal{R}_{\mathsf{pk}}}$, but the glue element is $\tilde{h} = \tilde{g}^{y \cdot q}$. There is extraction, but no zero-knowledge simulation.

Claim 12. *A PPT adversary cannot distinguish Game 11 from Game 12, except with negligible probability.*

This is the same as Claim 2 (extraction) in the proof of unforgeability.

Game 13. In this real signing game, again $\mathsf{pk}_{\mathsf{X},2} \in [\mathsf{pk}_{\mathsf{X},1}]_{\mathcal{R}_{\mathsf{pk}}}$, but the glue element is again $\tilde{h} = \tilde{g}^{y \cdot q}$. There is no extraction or zero-knowledge simulation.

Claim 13. *A PPT adversary cannot distinguish Game 12 from Game 13, except with negligible probability.*

This is the same as Claim 1 (simulation) in the proof of unforgeability.

This completes the proof of public key class-hiding for MS_{X}. (Theorem 6). $\square$