Selective Delegation of Attributes in Mercurial Signature Credentials*

C. Putman and K. M. Martin
Royal Holloway, University of London
Egham, Surrey, TW200EX, UK
Colin.Putman.2017@live.rhul.ac.uk,
Keith.Martin@rhul.ac.uk

Abstract

Anonymous credential schemes enable service providers to verify information that a credential holder willingly discloses, without needing any further personal data to corroborate that information, and without allowing the user to be tracked from one interaction to the next. Mercurial signatures are a novel class of anonymous credentials which show good promise as a simple and efficient construction without heavy reliance on zero-knowledge proofs. However, they still require significant development in order to achieve the functionality that most existing anonymous credential schemes provide. Encoding multiple attributes of the credential holder in such a way that they can be disclosed selectively with each use of the credential is often seen as a vital feature of anonymous credentials, and is one that mercurial signatures have not yet implemented. In this paper, we show a simple way to encode attributes in a mercurial signature credential and to regulate which attributes a credential holder can issue when delegating their credential to another user. We also extend the security model associated with mercurial signatures to account for the inclusion of attributes, and prove the security of our extension with respect to the original mercurial signature construction.

Keywords: Privacy · Anonymous credentials · Delegatable credentials · Mercurial signatures · Selective disclosure.

*First published in Cryptography and Coding, IMACC 2023 [11]. Reproduced with permission from Springer Nature.

1 Introduction

Privacy in the digital world is an increasingly serious concern, with many efforts being made to minimise the use, storage, and disclosure of personal information. However, this goal is difficult to reconcile with the interests of service providers who often have a need to know that their clients are engaging in good faith and are permitted to use their services. These reasonable checks usually involve some form of identification, which makes it easy for providers to build profiles on their users.

One powerful solution to these conflicting interests is the use of anonymous credentials, which are designed to allow service providers to verify information that the credential holder willingly discloses, without needing any further personal data to corroborate that information, and without allowing the user to be tracked from one interaction to the next. Anonymous credentials have been fully realised since 2001 [2], but their adoption has been extremely slow, due in large part to the cumbersome nature of the zero-knowledge proofs they rely on.

A recently developed type of anonymous credential, known as mercurial signatures [6], shows promise in overcoming this hurdle, as its malleable nature replaces much of the need for traditional zero-knowledge proofs. Mercurial signatures also allow for credential holders to delegate their credentials anonymously to other users, forming a chain of trust and enabling more private and versatile use of their systems.

However, the novelty of mercurial signatures means that they still lack much of the functionality that more established types of anonymous credential offer. Most notably, they do not yet offer the ability to encode detailed information about the credential holder's attributes and disclose only a subset of that information, a feature known as selective disclosure which is often viewed as necessary for an anonymous credential system.

Our contribution. This paper proposes an elegant extension to mercurial signatures which allows the selective disclosure of attributes, in a way which also remains compatible with the delegation of credentials. We also extend the security game associated with the original mercurial signatures to take account of the additional requirements that accompany credential attributes, and we prove the security of our extension with respect to the original CL19 mercurial signature construction. Although we use the CL19 construction for simplicity in our demonstration, the extension is also compatible with other credentials based on FHS-type structure-preserving signatures and set commitments, including the CL21, CLP22, and MSBM23 constructions [4, 7, 10].

Section 2 introduces the basic concepts that underpin this work. Section 3 gives an overview of the previous development of mercurial signatures. In Section 4, we detail our extension and give an overview of the associated proofs. We conclude in Section 5 with a brief look at the other areas in which this field can be progressed.

2 Preliminaries

A function $\epsilon : \mathbb{N} \to \mathbb{R}^+$ is called **negligible** if for all $c > 0$ there is a k_0 such that $\epsilon(k) < 1/k^c$ for all $k > k_0$. We use $a \leftarrow^R S$ to denote that a is chosen uniformly at random from a set S. Given a probabilistic algorithm $\mathtt{A}(a_1, ..., a_n)$, we use $\mathtt{A}(a_1, ..., a_n; r)$ to make the randomness r used by the algorithm explicit, and $[\mathtt{A}(a_1, ..., a_n)]$ to denote the set of points with positive probability of being output by A. We write groups

multiplicatively throughout this paper, and given a group $\mathbb{G}$ we use $\mathbb{G}^*$ to denote $\mathbb{G}\backslash\{1_{\mathbb{G}}\}$.

2.1 Bilinear maps

Given three cyclic groups $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$, all of prime order p, a **bilinear map** or **pairing** is an efficiently computable function $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ such that, given generators P and $\hat{P}$ of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively, $e(P^a, \hat{P}^b) = e(P, \hat{P})^{ab}$. The pairing is called **non-degenerate** if $e(P, \hat{P}) \neq 1_{\mathbb{G}_T}$, in which case $e(P, \hat{P})$ generates $\mathbb{G}_T$.

All of the bilinear maps used in this paper will be non-degenerate and based on cyclic groups of the same prime order. We define a **bilinear-group generator** to be a polynomial-time algorithm that takes as input a security parameter 1^κ and outputs a tuple $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, P, \hat{P})$ such that the groups $\mathbb{G}_1 = \langle P \rangle$, $\mathbb{G}_2 = \langle \hat{P} \rangle$, and $\mathbb{G}_T$ are cyclic groups of prime order p with $\lceil \log_2 p \rceil = \kappa$, and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a non-degenerate bilinear map.

2.2 Zero-knowledge proofs of knowledge

Let $L_{\mathcal{R}} = \{x | \exists w : (x, w) \in \mathcal{R}\} \subseteq \{0,1\}^*$ be a formal language with a binary, polynomial-time witness relation $\mathcal{R} \subseteq \{0,1\}^* \times \{0,1\}^*$, so that the membership of $x \in L_{\mathcal{R}}$ can be decided in polynomial time when given a witness w of length polynomial in $|x|$ certifying $(x, w) \in \mathcal{R}$. Consider an interactive protocol $(\mathcal{P}, \mathcal{V})$ between a potentially unbounded prover $\mathcal{P}$ and a PPT verifier $\mathcal{V}$ with outcome $(\cdot, b) \leftarrow (\mathcal{P}(\cdot,\cdot), \mathcal{V}(\cdot))$ where $b = 0$ indicates that $\mathcal{V}$ rejects and $b = 1$ indicates that $\mathcal{V}$ accepts the conversation with $\mathcal{P}$. Such a protocol is a **zero-knowledge proof of knowledge (ZKPoK)** if it satisfies the following three properties:

- **Completeness:** We call such a protocol $(\mathcal{P}, \mathcal{V})$ complete if, for all $x \in L_{\mathcal{R}}$ and w such that $(x, w) \in \mathcal{R}$ we have that $(\cdot, 1) \leftarrow (\mathcal{P}(x, w), \mathcal{V}(x))$ with probability 1.
- **Zero knowledge:** We say that the protocol $(\mathcal{P}, \mathcal{V})$ is zero-knowledge if for all PPT algorithms $\mathcal{V}^*$ there exists a PPT simulator $\mathcal{S}$ such that:

  $$\{\mathcal{S}^{\mathcal{V}^*}(x)\}_{x \in L_{\mathcal{R}}} \approx \{\langle \mathcal{P}(x, w), \mathcal{V}^*(x) \rangle\}_{(x,w) \in \mathcal{R}}$$

  where $\langle \mathcal{P}(\cdot,\cdot), \mathcal{V}^*(\cdot) \rangle$ denotes the transcript of the interaction between $\mathcal{P}$ and $\mathcal{V}$, and "$\approx$" denotes perfect indistinguishability.
- **Knowledge soundness:** We say that $(\mathcal{P}, \mathcal{V})$ is a proof of knowledge (PoK) relative to an NP relation $\mathcal{R}$ if, for any (possibly unbounded) malicious prover $\mathcal{P}^*$ such that $(\cdot, 1) \leftarrow (\mathcal{P}^*(x), \mathcal{V}(x))$ with non-negligible probability, there exists a PPT knowledge extractor $\mathcal{K}$ with rewinding black-box access to $\mathcal{P}^*$ such that $\mathcal{K}^{\mathcal{P}^*}(x)$ returns a value w satisfying $(x, w) \in \mathcal{R}$.

Zero-knowledge proofs of knowledge are used by provers to convince verifiers that they know a secret value w satisfying a specific statement x. For this work we are particularly interested in protocols to prove knowledge of a discrete logarithm, which can be efficiently instantiated using Σ-protocols as in Cramer et al. [5]. When using zero-knowledge proofs of knowledge, we denote the composite of proofs of witnesses $w_1, \ldots, w_n$ satisfying statements $x_1, \ldots, x_n$ by $\text{PoK}\{(w_1, \ldots, w_n) \mid x_1 \wedge \cdots \wedge x_n\}$.

2.3 Anonymous credentials

Anonymous credentials are a privacy-preserving system which allows a prover to obtain, from a trusted issuer, a credential on one or more **attributes** representing access rights or pieces of identifying information, which can then be used to prove their possession of these attributes to a verifier in zero knowledge. An anonymous credential system consists of the following PPT algorithms:

$\mathsf{IssuerKeyGen}(1^\kappa, 1^t)$: A probabilistic algorithm which takes as input a security parameter κ and an upper bound t on the size of attribute sets, and outputs a key pair (osk, opk) for an issuer.

$\mathsf{ProverKeyGen}(\mathsf{opk})$: A probabilistic algorithm which takes as input an issuer's public key opk and outputs a key pair (usk, upk) for a prover.

($\mathsf{Obtain}(\mathsf{usk}, \mathsf{opk}, \mathsf{A})$, $\mathsf{Issue}(\mathsf{upk}, \mathsf{osk}, \mathsf{A})$): A pair of probabilistic algorithms run by a prover and an issuer, respectively, which interact during execution. Obtain takes as input the prover's secret key usk, the issuer's public key opk, and a non-empty attribute set A of size $|\mathsf{A}| \leq t$, and Issue takes as input the prover's public key upk, the issuer's secret key osk, and a non-empty attribute set A of size $|\mathsf{A}| \leq t$. At the end of the protocol, Obtain outputs a credential cred for the user on the attribute set A, or $\perp$ if the execution failed.

($\mathsf{Show}(\mathsf{opk}, \mathsf{A}, \mathsf{D}, \mathsf{cred})$, $\mathsf{Verify}(\mathsf{opk}, \mathsf{D})$): A pair of algorithms run by a prover and a verifier, respectively, which interact during execution. Show is a probabilistic algorithm which takes as input an issuer's public key opk, an attribute set A of size $|\mathsf{A}| \leq t$, a non-empty set $\mathsf{D} \subseteq \mathsf{A}$, and a credential cred. Verify is a deterministic algorithm which takes as input an issuer's public key opk and a set D. At the end of the protocol, Verify outputs either 1 or 0, indicating whether it accepts the credential showing or not.

Informally, the security properties that an anonymous credential must satisfy are as follows:

- **Correctness:** A showing of a credential cred with respect to a non-empty set of attributes D always verifies if cred was issued honestly for some attribute set A such that $\mathsf{D} \subseteq \mathsf{A}$.
- **Unforgeability:** A prover cannot perform a valid showing of attributes for which they do not possess a credential, and no coalition of provers can combine their credentials to perform a valid showing of attributes for which no single prover in the coalition has a credential. This must hold even after seeing arbitrary showings of valid credentials by honest users.
- **Anonymity:** During a showing, no verifier, issuer, or coalition of multiple verifiers and/or issuers can learn anything about the prover except that they possess a valid credential on an attribute set that includes D.

We say that an anonymous credential system offers **selective disclosure** if it supports attribute sets of size greater than 1 (and therefore allows D to be a proper subset of A). Such credentials are often called **attribute-based credentials**.

We say that a credential is **multi-show** if it can be used in multiple runs of the Show algorithm and the verifier(s) to which it has been shown cannot distinguish them from protocol runs using different credentials. We can identify two distinct families of

multi-show credentials: **zero-knowledge credentials**, which are not revealed to the verifier during the Show algorithm, instead using zero-knowledge proofs to convince the verifier of the valid credential's existence, and **self-blindable credentials**, which are partially or fully shown to the verifier during the Show algorithm, and subsequently altered such that they remain valid but are unrecognisable in later showings.

2.4 Delegatable credentials

Delegatable credentials, first proposed in 2006 by Chase and Lysyanskaya [3], extend the anonymous credential model by allowing the holder of a credential to (anonymously) issue new credentials to other users. This process is called delegation, and allows credentials to form a chain of trust; each credential identifies the root authority which issued the original credential, and how many steps removed from that original credential it is, but does not uniquely identify the other delegators along the chain.

Crites and Lysyanskaya [6] provide a model for delegatable credentials, which differs from the basic anonymous credential model in a few key ways. First, they add the following PPT algorithm:

NymGen(sk, $L(\mathrm{pk}_0)$): A probabilistic algorithm that takes as input a participant's secret key sk and a **delegation level** (i.e. number of steps removed from the root issuer) $L(\mathrm{pk}_0)$ under the root issuer whose public key is pk_0, and generates a pseudonym and auxiliary information for that participant at that level.

The pseudonym and auxiliary information function as a fresh public key and secret key, respectively, that other participants cannot link to the underlying long-term key pair. The Issue/Obtain and Prove/Verify algorithms then become:

(Obtain($L_I(\mathrm{pk}_0)$, pk_0, sk_R, nym_R, aux_R, nym_I), Issue($L_I(\mathrm{pk}_0)$, pk_0, sk_I, nym_I, aux_I, cred_I, nym_R)): Obtain takes as input the issuer's delegation level, the root authority's public key, the secret key and a pseudonym and auxiliary information belonging to the recipient, and the issuer's pseudonym (not the issuer's long-term public key), and Issue takes as input the issuer's delegation level, the root authority's public key, the secret key, pseudonym, auxiliary information, and credential belonging to the issuer, and the recipient's pseudonym. The protocol outputs a new credential for the recipient, which has a (potentially equal) subset of the attributes in cred_I and delegation level $L_I(\mathrm{pk}_0)$ + 1. By convention, a root authority runs Issue with $L_I(\mathrm{pk}_0) = 0$, $\mathrm{nym}_I = \mathrm{pk}_0$, and $\mathrm{aux}_I = \mathrm{cred}_I = \perp$.

(Prove($L_P(\mathrm{pk}_0)$, pk_0, sk_P, nym_P, aux_P, cred_P), Verify(pp, $L_P(\mathrm{pk}_0)$, pk_0, nym_P)): Prove takes as input the prover's delegation level, the root authority's public key, the prover's secret key, the pseudonym by which the verifier knows the prover (which should differ from the pseudonym to which the credential was issued) and its auxiliary information, and the prover's credential. Verify takes as input the system parameters pp (which were assumed in the basic model to be part of the issuer's public key), the prover's delegation level, the root authority's public key, and the prover's pseudonym. The output is unchanged from the basic anonymous credential model.

Delegatable credentials also extend the basic security goals of anonymous credentials. In particular, the Correctness and Unforgeability properties are extended such that every credential along a chain must be generated correctly and honestly in or-

der for the credential shown at the end to be considered correct and honest. The Anonymity property is also extended to apply to delegators; this can be to protect the delegators' privacy, but it is also necessary for providing anonymity to the prover, since two credentials with the same root authority but different delegators would otherwise become distinguishable, which would reduce the prover's anonymity set.

Though there is no clear consensus on how delegatable credentials should use attributes, Blömer and Bobolz [1] propose the requirement that a delegated credential must encode a subset of the attributes in the delegator's credential, conceptually enforcing that not only the credential but the attributes themselves are delegated from one level to the next. We will refer to this as **selective delegation** (of attributes). This model is ideal for scenarios in which the attributes of delegated credentials are expected to represent strictly equal or lesser privileges than those of the credentials higher up the chain. This includes the case in which the issuing process itself is delegated, as the selective delegation model allows the root issuer to provide each sub-issuer with a credential containing all of the attributes that sub-issuer is empowered to sign.

3 Previous work

3.1 SPS-EQ credentials

The precursor to mercurial signatures, SPS-EQ credentials are a form of self-blindable, attribute-based credentials first proposed by Hanser and Slamanig in 2014 [9] and refined by Fuchsbauer et al. in 2019 [8]. They are based on a novel primitive called structure-preserving signatures on equivalence classes, usually abbreviated as SPS-EQ.

A structure-preserving signature scheme is one in which the message and the signature are both made up of group elements in the same bilinear pairing, as is the public key. SPS-EQ schemes further define an equivalence relation on the message and signature spaces, and allow both messages and signatures to be randomised within the resulting equivalence classes.

In addition to the usual Sign and Verify functions, SPS-EQ schemes include a function $\mathsf{ChgRep}_{\mathcal{R}}(M, \sigma, \mu, \mathsf{pk})$, parametrised by equivalence relation $\mathcal{R}$, which takes as input a message M, a signature σ, a randomising factor μ, and a public key pk, and outputs a message M' in the same equivalence class as M and a signature σ' such that:

$$\mathsf{Verify}(M', \sigma', \mathsf{pk}) = \mathsf{Verify}(M, \sigma, \mathsf{pk}).$$

The security requirements of SPS-EQ schemes include a class-hiding property, which states that an adversary given a pair of messages should not be able to tell whether they are in the same equivalence class, and an origin-hiding property, which states that the output of the $\mathsf{ChgRep}_{\mathcal{R}}$ function should be indistinguishable from a fresh message-signature pair.

Hanser and Slamanig's construction defines its message space by generating a bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ with DDH-hard groups, and creating a vector space of elements in $(\mathbb{G}_1^*)^l$ with l greater than 1. The equivalence relation is then defined such that two messages M and M' are equivalent if and only if M' is a scalar power of M.

The secret key is a vector $(x_i)_{i \in l}$ in $(\mathbb{Z}_p^*)^l$ and the public key is the corresponding vector $(\hat{X}_i)_{i \in l} = (\hat{P}^{x_i})_{i \in l}$ where $\hat{P}$ is a generator of $\mathbb{G}_2$ included in the public pa-

rameters; the public key is therefore a vector in $(\mathbb{G}_2^*)^l$. A signature σ on a message $M \in (\mathbb{G}_1^*)^l$ is a tuple $(Z, Y, \hat{Y})$ with $Z, Y \in \mathbb{G}_1$ and $\hat{Y} \in \mathbb{G}_2$ such that:

$$Z = \prod_{i \in l} M_i^{x_i y},$$

$$Y = P^{\frac{1}{y}},$$

$$\hat{Y} = \hat{P}^{\frac{1}{y}},$$

where P and $\hat{P}$ are generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively, included in the public parameters, and y is chosen randomly from $\mathbb{Z}_p^*$ at the time of signing.

SPS-EQ schemes were specifically designed for use in anonymous credentials, using their randomisation as the credential blinding mechanism, with SPS-EQ's class-hiding and origin-hiding properties providing unlinkability for the resulting credentials. In order to encode attributes, they had to be combined with a commitment scheme that could be randomised in a manner consistent with the SPS-EQ construction.

3.2 Randomisable set commitments

In order to support attributes with selective disclosure, Hanser and Slamanig also constructed a set commitment scheme which can opened securely to a chosen subset of the committed set, and can be randomised in a similar manner to the signature scheme, allowing a set commitment to the desired attribute set A to be used as the message in the signature scheme.

The scheme works by committing to a polynomial $f_S(X)$ whose roots are the members of the committed set S; that is, $f_S(X) = \prod_{s \in S} (X - s)$. S must be a subset of $\mathbb{Z}_p$, but could encode other types of data using a hash function.

The commitment scheme is instantiated by a manager who selects the security parameter 1^κ and a maximum set cardinality 1^t and generates a bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ which is published along with generators P and $\hat{P}$ for $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively. The manager then chooses a random trapdoor $a \in \mathbb{Z}_p$ and publishes $(P^{a^i}, \hat{P}^{a^i})_{i \in [t]}$. This ensures that the trapdoor is not needed to compute $P^{f_S(a)} = \prod_{i=0}^{|S|} P^{f_i a^i}$, which is needed to generate and verify commitments, or $\hat{P}^{f_S(a)}$, which is needed to verify subsets.

In order to commit to a set, the prover chooses a random $\rho \in \mathbb{Z}_p^*$ and computes the commitment $C = P^{\rho f_S(a)}$ which is stored along with the opening information $O = (0, \rho)$. To open this commitment to the full set S, the prover sends S, C, and O to the verifier, who is able to compute C from S and O and confirm the match.

If the prover wishes to open a subset T of the committed set S, they first generate a witness $W = P^{\rho f_{S \setminus T}(a)}$. This can be verified without revealing the full set S by using the bilinear map to check whether $e(W, \hat{P}^{f_T(a)}) = e(C, \hat{P})$.

If a commitment C is randomised using a blinding factor μ after a witness W has been generated as above, applying the same blinding factor to W produces a new witness W' which is consistent with the new commitment. This allows the set commitments to be used as messages for SPS-EQ signatures, such that it is possible to encode attributes in an SPS-EQ credential and selectively disclose them to a verifier even after the credential has been randomised.

Subsequent work by Connolly et al. [4] extended this commitment scheme with a function for opening on disjoint sets, allowing the commitment's owner to prove that

certain values are not included in the committed set. They also add an optional proof of exponentiation (PoE) technique to shift computation work from the verifier to the prover during openings. These functions require no changes to the structure of the commitment, and hence incur no additional cost, while significantly expanding the expressiveness of the scheme.

3.3 Mercurial signatures

Mercurial signatures are an extension to SPS-EQ proposed by Crites and Lysyanskaya [6] with the intention of supporting delegatable credentials. To do this, they add a set of functions to allow a key pair (pk, sk) to be randomised such that the resulting (pk', sk') is still a valid key pair, and a signature under pk to be randomised to produce a valid signature on the same message under pk'. Randomising the public key allows delegators to be anonymised, and prevents the credential chain of a delegated credential from becoming identifiable.

Given parameterised equivalence relations $\mathcal{R}_M$, $\mathcal{R}_{\mathsf{pk}}$, and $\mathcal{R}_{\mathsf{sk}}$, Crites and Lysyanskaya define mercurial signatures generally as consisting of the following PPT algorithms:

$\mathsf{PPGen}(1^k) \to PP$: A probabilistic algorithm which takes as input the security parameter 1^k and outputs the public parameters PP, including parameters for $\mathcal{R}_M$, $\mathcal{R}_{\mathsf{pk}}$, and $\mathcal{R}_{\mathsf{sk}}$, and parameters for algorithms sample_ρ and sample_μ, which are used to generate converters for keys and for messages, respectively.

$\mathsf{KeyGen}(PP, l) \to (\mathsf{pk}, \mathsf{sk})$: A probabilistic algorithm which takes as input the public parameters PP and a length parameter l and outputs a key pair (pk, sk). Following the authors' example, we also write $(\mathsf{pk}, \mathsf{sk}) \in \mathsf{KeyGen}(PP, l)$ to denote that there exists a set of random choices KeyGen could make on input (PP, l) that would result in (pk, sk) as the output. It is also noted that the message space $\mathcal{M}$ is well-defined from PP and l.

$\mathsf{Sign}(\mathsf{sk}, M) \to \sigma$: A probabilistic algorithm which takes as input a signing key sk and a message $M \in \mathcal{M}$ and outputs a signature σ.

$\mathsf{Verify}(\mathsf{pk}, M, \sigma)$: $\to 0/1$: A deterministic algorithm which takes as input a public key pk, a message $M \in \mathcal{M}$, and a purported signature σ, and outputs 0 or 1.

$\mathsf{ConvertSK}(\mathsf{sk},\rho) \to \bar{\mathsf{sk}}$: A deterministic algorithm which takes as input a signing key sk and a key converter $\rho \in \mathsf{sample}_\rho$ and outputs a new signing key $\bar{\mathsf{sk}} \in [\mathsf{sk}]_{\mathcal{R}_{\mathsf{sk}}}$.

$\mathsf{ConvertPK}(\mathsf{pk},\rho) \to \bar{\mathsf{pk}}$: A deterministic algorithm which takes as input a public key pk and a key converter $\rho \in \mathsf{sample}_\rho$ and outputs a new public key $\bar{\mathsf{pk}} \in [\mathsf{pk}]_{\mathcal{R}_{\mathsf{pk}}}$.

$\mathsf{ConvertSig}(\mathsf{pk}, M, \sigma, \rho) \to \bar{\sigma}$: A probabilistic algorithm which takes as input a public key pk, a message $M \in \mathcal{M}$, a signature σ, and a key converter $\rho \in \mathsf{sample}_\rho$, and outputs a new signature $\bar{\sigma}$.

$\mathsf{ChangeRep}(\mathsf{pk}, M, \sigma, \mu) \to (M', \sigma')$: A probabilistic algorithm which takes as input a public key pk, a messages $M \in \mathcal{M}$, a signature σ, and a message converter $\mu \in \mathsf{sample}_\mu$, computes a new message $M' \in [M]_{\mathcal{R}_M}$ and a new signature σ', and outputs (M', σ').

In order to define a construction for mercurial signatures, Crites and Lysyanskaya made a simple extension from Hanser and Slamanig's structure-preserving signature construction. Recalling that the secret key is a vector $(x_i)_{i\in l}$ in $(\mathbb{Z}_p^*)^l$ and the public key is the corresponding vector $(\hat{X}_i)_{i\in l} = (\hat{P}^{x_i})_{i\in l}$ in $(\mathbb{G}_2^*)^l$, Crites and Lysyanskaya's construction randomises the keys by taking an input $\rho \in \mathbb{Z}_p^*$ and setting $\mathrm{sk}' = (\rho x_i)_{i\in l}$ and $\mathrm{pk}' = (\hat{X}_i^{\rho})_{i\in l}$. A signature $\sigma = (Z, Y, \hat{Y})$ can then be randomised by choosing a random $\psi \in \mathbb{Z}_p^*$ and setting $\sigma' = (Z^{\psi\rho}, Y^{\frac{1}{\psi}}, \hat{Y}^{\frac{1}{\psi}})$. This is identical to the signature randomisation in SPS-EQ, and ensures that if σ is a valid signature on a message M under pk, σ' is a valid signature on M under pk'.

One major limitation of the CL19 mercurial signature construction (and, indeed, of the SPS-EQ construction before it) is that the length of the signer's key serves as an upper bound on the length of the message to be signed. This is especially problematic in the context of delegatable credentials, where a typical message consists of the prover's public key plus a representation of at least one attribute. If the issuer has a key of length l and signs a credential with k group elements representing its attributes, the prover's key length can only be at most $l - k$. Furthermore, if that prover then wishes to delegate the credential with all of its attributes, the recipient's key length can only be up to $l - 2k$, and so on.

In a subsequent paper, Crites and Lysyanskaya proposed a method to overcome this problem [7]. The message is assumed to be of the form $(P, M_1, ..., M_n)$, where P is a generator of $\mathbb{G}_1$ and $M_i = P^{m_i}$ for all $1 \leq i \leq n$, with the values m_i being encoded information such as private key elements and attributes. The approach taken by Crites and Lysyanskaya is to transform each element of the message into its own fixed-length message which can be signed separately using the original scheme.

Later work by Connolly et al. [4] used similar techniques to obtain issuer-hiding SPS-EQ credentials which could easily be converted to mercurial signatures by adding a protocol for delegation. They also made use of the FHS randomisable set commitment scheme in Section 3.2 to encode attributes with selective disclosure; however, their work does not achieve selective delegation, as we will see in the next section.

The only relative of mercurial signatures to directly combine delegation with selective disclosure of attributes is the recent work by Mir et al. [10], which replaces the SPS-EQ primitive with SPS-EQ on Updatable Commitments in order to achieve an efficient construction with several desirable properties, including the ability to restrict the number of times a credential can be delegated and to prevent a delegatee from showing attributes from a higher delegation level. They also introduce a method to batch subset openings of multiple commitments for efficient verification, called cross-set commitment aggregation. However, their approach also does not consider selective delegation of attributes; in contrast to Blömer and Bobolz's model, there is no enforced relationship between the attributes on different delegation levels. A relationship could still be shown to a verifier, but only if every delegator provides the delegatee with the opening of their commitment so that the delegatee can show the relevant attributes on higher levels. Even if the delegator uses a subset witness to keep some attributes hidden, this would sacrifice the information-theoretic privacy of the delegation protocol.

4 Providing selective disclosure

The CL19 credential construction does not include a way to encode any information other than the holder's public key, meaning it cannot be used to prove anything other

than that the holder is the genuine owner of the credential. While the mere fact of possessing a credential can be taken to certify a single, binary attribute, most existing credential schemes are built to be able to certify a multitude of attributes, along with providing a way to disclose only those that are relevant during a particular transaction.

One approach to encoding these attributes is to use the same set commitment scheme as SPS-EQ signatures. Since mercurial signatures directly extend SPS-EQ and the credential schemes are derived in a similar way, when a mercurial signature is first issued the set commitment can work identically to the commitments in SPS-EQ credentials. This also has the benefit of allowing a credential's size to be constant, rather than linear in the number of attributes.

The CLP22 scheme [4] takes this approach, but their work does not consider credential delegation. The MSBM23 scheme [10] is similar but does allow credential delegation; however, under their model the attributes in a delegated credential bear no relation to the delegator's attributes. In this paper, we are concerned with the more restrictive model of selective delegation introduced by Blömer and Bobolz [1], which adds a further challenge to overcome.

4.1 Selective delegation

The difficulty here arises because when one user, Alice, delegates a credential to another user, Bob, Bob's attribute commitment $\hat{C}_B$ has not been signed by the root issuer, and Alice's identity is intentionally hidden from any party Bob discloses the credential to, meaning that her signature cannot be trusted in the same way.

In order to ensure that no user has issued a credential more permissive than their own, the verifier of Bob's credential must have some way of confirming that Alice was authorised to issue the attributes in Bob's attribute set B. This means that the verifier must be able to check whether or not B is a subset of Alice's attribute set A. In order to achieve this, we will design a special subset witness that can be included on the delegation chain to connect Alice and Bob's credentials, leading to a chain in which each credential except for the first has an associated witness value linking it to the one before. The difficulty in this approach lies in designing a witness value that can be computed by Alice and Bob at the time of delegation and does not leak any information about either attribute set.

In the case that $\mathsf{A} = \mathsf{B}$, the verifier can already confirm the relation with the pairing equation $e(X_1^{\rho_1\rho_2}, \hat{C}_B^{\rho_3}) = e(C_A^{\rho_1\rho_2}, \hat{Y}_1^{\rho_3})$ which can be verified using only the elements $X_1^{\rho_1\rho_2}, C_A^{\rho_1\rho_2}, \hat{Y}_1^{\rho_3}, \hat{C}_B^{\rho_3}$ within the blinded cred_B. Here X_1 is the first element of the public key in Alice's credential, $\hat{Y}_1$ is the first element of the public key in Bob's credential, C_A and $\hat{C}_B$ are Alice and Bob's attribute commitments, respectively, as formulated in the FHS19 credential scheme [8], and ρ_1, ρ_2, and ρ_3 are the blinding factors used by Alice and Bob to randomise their credentials, with Alice applying ρ_1 to her credential, and Bob applying ρ_2 to Alice's credential and ρ_3 to his own credential. Note that the assignment of $\mathbb{G}_1$ and $\mathbb{G}_2$ in this example is arbitrary and can be reversed as needed depending on the delegation level.

To support the case where B is a proper subset of A, the verifier will need to replace $X_1^{\rho_1\rho_2}$ with $X_1^{\rho_1\rho_2 f_{A\setminus B}(a)}$, a subset witness that can only be generated by Alice. Alice could in theory supply this witness to Bob during delegation, but if Bob subsequently delegates the credential further, that witness will need to be passed along again. Effectively, this is no different to storing the witness value as part of the credential chain.

However, formulating the witness in this way leaks whether or not $\mathsf{A} = \mathsf{B}$, since in that case $f_{A\setminus B}(a) = 1$, and so $X_1^{\rho_1\rho_2 f_{A\setminus B}(a)} = X_1^{\rho_1\rho_2}$. Indeed, this means anyone could attempt to guess the full attribute set of a credential they have seen and check their guess by creating a commitment $\hat{C}_Q$ and testing whether $e(X_1^{\rho_1\rho_2}, \hat{C}_Q) = e(C_A^{\rho_1\rho_2}, \hat{Y}_1^{\rho_3})$. This can be prevented by decoupling the commitment's opening factor from the owner's key pair, making it instead a fresh, random exponent, ψ. The pairing equation then correspondingly becomes:

$$e(P^{\rho_1\rho_2\psi_A f_{A\setminus B}(a)}, \hat{C}_B^{\rho_3}) = e(C_A^{\rho_1\rho_2}, \hat{P}^{\psi_B\rho_3}).$$

Because the element $P^{\psi_A f_{A\setminus B}(a)}$ is in $\mathbb{G}_1$, it cannot be signed by Alice, and it cannot be generated in advance to gain the issuer's signature, since it requires knowledge of Bob's attribute set. With two unsigned elements in the equation, the values could be adjusted to a trivial solution, and so the result cannot be trusted by a verifier. However, Bob can include the exponent $\rho_3^{-1}\psi_B^{-1}$ and rearrange the pairing equation to:

$$e(P^{\rho_1\rho_2\psi_A\rho_3^{-1}\psi_B^{-1} f_{A\setminus B}(a)}, \hat{C}_B^{\rho_3}) = e(C_A^{\rho_1\rho_2}, \hat{P}).$$

Because $P^{\rho_1\rho_2\psi_A\rho_3^{-1}\psi_B^{-1} f_{A\setminus B}(a)}$ is a unique solution to this equation and is being compared with two signed elements and a public parameter, an adversary cannot modify it without rendering the credential useless, so it can safely be placed on the credential chain as an unsigned witness tag connecting party A's credential to party B's credential. Using this tag, any verifier can easily check that $\mathsf{B} \subseteq \mathsf{A}$, while the random factors mask any further information.

4.2 Construction of mercurial signature credentials with set commitments

Applying these modifications to CL19 credentials [6], we arrive at the following construction.

Let λ be the maximum delegation level that should be permitted on a credential. This has to be specified because the keys must get shorter at each delegation level to allow signing attributes.

Define $\mathsf{MS}_i = (\mathsf{PPGen}_i, \mathsf{KeyGen}_i, \mathsf{Sign}_i, \mathsf{Verify}_i, \mathsf{ConvertSK}_i, \mathsf{ConvertPK}_i, \mathsf{ConvertSig}_i, \mathsf{ChangeRep}_i)$ for all $0 \leq i \leq \lambda$ as instantiations of mercurial signatures as constructed by Crites and Lysyanskaya [6], such that for all $i, j \in [\lambda]$, $\mathsf{PPGen}_i = \mathsf{PPGen}_j$, MS_i is parameterised with key and message length $l_i = \lambda + 1 - i$, and the roles of $\mathbb{G}_1$ and $\mathbb{G}_2$ are reversed in all other algorithms of MS_i if i is odd, so that for $0 \leq i < \lambda$, $(\mathcal{R}_M)_i = (\mathcal{R}_{\mathrm{pk}})_{i+1}$. Let $\mathsf{Com}_0 = (\mathsf{Setup}_0, \mathsf{Commit}_0, \mathsf{Open}_0, \mathsf{OpenSubset}_0, \mathsf{VerifySubset}_0)$ and $\mathsf{Com}_1 = (\mathsf{Setup}_1, \mathsf{Commit}_1, \mathsf{Open}_1, \mathsf{OpenSubset}_1, \mathsf{VerifySubset}_1)$ be two instantiations of FHS randomisable set commitments [8] with the roles of $\mathbb{G}_1$ and $\mathbb{G}_2$ in Com_1 reversed.

The construction consists of the following algorithms and protocols. For simplicity, the protocols are written as if the credential chain length L is even; if L is odd, the roles of $\mathbb{G}_1$ and $\mathbb{G}_2$ must be reversed.

$\mathsf{Setup}(1^k, 1^t, 1^\lambda) \to (params)$: Given $k, t, \lambda > 0$, compute $PP \leftarrow \mathsf{PPGen}_0(1^k)$; extract $(P, \hat{P})$ from PP, choose $a \leftarrow^R \mathbb{Z}_p$, and compute $(P^{a^i}, \hat{P}^{a^i})_{i\in[t]}$; output $params = (PP, p, t, \lambda, (P^{a^i}, \hat{P}^{a^i})_{i\in[t]})$.

$\mathsf{KeyGen}(params) \rightarrow (\mathsf{pk}, \mathsf{sk})$: There are two cases. For the root authority, compute $(\mathsf{pk}_0, \mathsf{sk}_0) \leftarrow \mathsf{KeyGen}_0(PP, l_0)$ and output it. For others, compute $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}_i(PP, l_i)$ for all $i \in [\lambda]$ and output all of the key pairs $(\mathsf{pk}_i, \mathsf{sk}_i)_{i \in [\lambda]}$.

$\mathsf{Issue}(params, L, \mathsf{pk}_0, \mathsf{sk}_I, \mathsf{pk}_I, O_I, \mathsf{cred}_I, \mathsf{A}_I, \mathsf{A}_R) \leftrightarrow \mathsf{Receive}(params, L, \mathsf{pk}_0, \mathsf{sk}_{L+1}, \mathsf{pk}_{L+1}, \mathsf{A}_R) \rightarrow (\mathsf{cred}_R, \rho)$

- If $L = 0$, define $\mathsf{cred}_I = \perp$ and $\mathsf{A}_I = \mathbb{Z}_p$.
- If $L \geq \lambda$, return $\perp$.
- Receiver calculates $(C_R, O_R) \leftarrow \mathsf{Commit}(params, \mathsf{A}_R)$, extracts ρ_R from $O_R = (b, \rho_R)$, and calculates P^{ρ_R}.
- Receiver sends $C_R, P^{\rho_R}, \mathsf{pk}_{L+1}$.
- Receiver proves $\mathsf{PoK}\{\alpha_1, ..., \alpha_l, \beta | (P^{\alpha_1}, ..., P^{\alpha_l}) = \mathsf{pk}_{L+1} \wedge P^{\beta} = P^{\rho_R}\}$.
- If $\mathsf{A}_R \not\subseteq \mathsf{A}_I$ or $|\mathsf{A}_R| > t$ or the PoK fails, Issuer returns $\perp$.
- If $e(C_R, \hat{P}) \neq e(P^{\rho_R}, \hat{P}^{f_{A_R}(a)})$ and $\forall a' \in \mathsf{A}_R : P^{a'} \neq P^a$, Issuer returns $\perp$.
- If $L = 0$, Issuer computes $\sigma_1 \leftarrow \mathsf{Sign}_0(\mathsf{sk}_I, (\mathsf{pk}_{L+1}, C_R))$ and sends $\mathsf{cred}_R = (\mathsf{pk}_{L+1}, C_R, \sigma_1)$.
- If $L > 0$, Issuer computes $(\mathsf{cred}'_I, \mathsf{sk}'_I, \psi) \leftarrow \mathsf{RandCred}(\mathsf{cred}_I, \mathsf{sk}_I, \mathsf{pk}_0, L)$ and $\sigma_{L+1} \leftarrow \mathsf{Sign}_L(\mathsf{sk}'_I, (\mathsf{pk}_{L+1}, C_R))$.
- If $L > 0$ and $\forall a' \in \mathsf{A}_I : P^{a'} \neq P^a$, Issuer extracts ρ_I from O_I and $\hat{C}'_I$ from cred'_I, computes $\bar{W}_R \leftarrow \mathsf{OpenSubset}(params, \hat{C}'_I, \mathsf{A}_I, (0, \rho_I \psi), \mathsf{A}_R)$, and sends $\sigma_{L+1}, \bar{W}_R$, and cred'_I.
- If $L > 0$ and $\exists a' \in \mathsf{A}_I \backslash \mathsf{A}_R : P^{a'} = P^a$, Issuer calculates $f_{A_R}(a')^{-1}$, sets $\bar{W}_R \leftarrow (C'_I)^{f_{A_R}(a')^{-1}}$, and sends $\sigma_{L+1}, \bar{W}_R$, and cred'_I.
- If $L > 0$ and $\exists a' \in \mathsf{A}_R : P^{a'} = P^a$, Issuer sets $\bar{C}_R = C'_I$, $\bar{W}_R = P^{\rho_R}$, and $\sigma_{L+1} \leftarrow \mathsf{Sign}_L(\mathsf{sk}'_I, (\mathsf{pk}_{L+1}, \bar{C}_R))$, and sends $\sigma_{L+1}, \bar{C}_R, \bar{W}_R$, and cred'_I.
- If $\exists a' \in \mathsf{A}_R : P^{a'} = P^a$, Receiver sets $C_R = \bar{C}_R$.
- If $\forall 2 \leq i \leq L : \mathsf{Verify}_{i-1}(\mathsf{pk}'_{i-1}, \mathsf{nym}'_i, \sigma'_i) = 1 \wedge e(C'_i, W'_i) = e(P, C'_{i-1})$ and $\mathsf{Verify}_0(\mathsf{pk}_0, \mathsf{nym}'_1, \sigma'_1) = 1$ and $\mathsf{Verify}_L(\mathsf{pk}'_I, \mathsf{nym}_R, \sigma_{L+1}) = 1$ and $e(C_R, \bar{W}_R) = e(P^{\rho_R}, C'_I)$, Receiver calculates $W_R = \bar{W}_R^{\rho_R^{-1}}$, appends pk_{L+1}, C_R, σ_{L+1}, W_R to cred'_I to form cred_R, and stores $\mathsf{cred}_R, O_R, \mathsf{sk}_R = \mathsf{sk}_{L+1}, \mathsf{pk}_R = \mathsf{pk}_{L+1}$.

$\mathsf{RandCred}(\mathsf{cred}, \mathsf{sk}, \mathsf{pk}_0, L) \rightarrow (\mathsf{cred}', \mathsf{sk}', \rho)$: If $L > \lambda$, return $\perp$; otherwise, given cred of the form $(\mathsf{nym}_1, ..., \mathsf{nym}_L, \sigma_1, ..., \sigma_L, W_2, ..., W_L)$, where $\mathsf{nym}_i = (\mathsf{pk}_i, C_i)$, choose random $(\rho_1, ..., \rho_L) \leftarrow (\mathbb{Z}_p^*)^L$; define $\mathsf{nym}'_0 = \mathsf{pk}_0, \bar{\sigma}_1 = \sigma_1$; if $L \geq 2$, for $2 \leq i \leq L$, set $\bar{\sigma}_i = \mathsf{ConvertSig}_{i-1}(\mathsf{pk}_{i-1}, \mathsf{nym}_i, \sigma_i, \rho_{i-1})$ and $W'_i = W_i^{\rho_{i-1} \rho_i^{-1}}$; for $1 \leq i \leq L$, set $(\mathsf{nym}'_i, \sigma'_i) = \mathsf{ChangeRep}_i(\mathsf{nym}'_{i-1}, \mathsf{nym}_i, \bar{\sigma}_i, \rho_i)$; set $\mathsf{cred}' = (\mathsf{nym}'_1, ..., \mathsf{nym}'_L, \sigma'_1, ..., \sigma'_L, W'_2, ..., W'_L)$ and $\mathsf{sk}' = \rho_L(\mathsf{sk})$; output $(\mathsf{cred}', \mathsf{sk}', \rho_L)$.

$\mathsf{CredProve}(params, L_P, \mathsf{pk}_0, \mathsf{sk}_P, \mathsf{pk}_P, O_P, \mathsf{cred}_P, \mathsf{A}_P, \mathsf{S}, \mathsf{D}) \leftrightarrow \mathsf{CredVerify}(params, \mathsf{pk}_0) \rightarrow \{0, 1\}$

- Prover extracts C_{L_P} from cred_P and computes $W_{L_P} \leftarrow \mathsf{OpenSubset}(params, C_{L_P}, \mathsf{A}_P, O_P, \mathsf{S})$, $\hat{W}_{L_P} \leftarrow \mathsf{OpenDisjoint}(params, C_{L_P}.\mathsf{A}_P, O_P, \mathsf{D})$.

- If $L_P \leq \lambda$, Prover computes $(\mathsf{cred}'_P, \mathsf{sk}'_P, \psi) \leftarrow \mathsf{RandCred}(\mathsf{cred}_P, \mathsf{sk}_P, \mathsf{pk}_0, L_P)$, $W'_{L_P} = W^{\psi}_{L_P}$, and $\hat{W}'_{L_P} = \hat{W}^{\psi}_{L_P}$.
- Prover sends $\mathsf{cred}'_P, W'_{L_P}, \hat{W}'_{L_P}, \mathsf{S}, \mathsf{D}$.
- Prover proves $\mathsf{PoK}\{\alpha_1, ..., \alpha_l | (P^{\alpha_1}, ..., P^{\alpha_l}) = \mathsf{pk}'_{L_P}\}$.
- Verifier extracts $\mathsf{nym}'_{L_P} = (\mathsf{pk}'_{L_P}, C'_{L_P})$ from cred_P and infers L_P from the length of cred'_P.
- If $L_P > \lambda$ or the PoK fails, Verifier returns 0.
- If $\forall 2 \leq i \leq L_P : \mathsf{Verify}_{i-1}(\mathsf{pk}'_{i-1}, \mathsf{nym}'_i, \sigma'_i) = 1 \wedge e(C'_i, W'_i) = e(P, C'_{i-1})$, $\mathsf{Verify}_0(\mathsf{pk}_0, \mathsf{nym}'_1, \sigma'_1) = 1$, $\mathsf{VerifySubset}(params, C'_{L_P}, \mathsf{S}, W'_{L_P}) = 1$, and $\mathsf{VerifyDisjoint}(params, C'_{L_P}, \mathsf{D}, \hat{W}'_{L_P}) = 1$, Verifier outputs 1; otherwise, Verifier outputs 0.

Similar adjustments can be made to add selective delegation to the variable-length CL21 scheme [7] and to the issuer-hiding CLP22 scheme [4], since all three constructions randomise message elements in the same way and are therefore compatible with the same set commitments and witness values. The witness values in Section 4.1 are also compatible with the MSBM23 credential scheme [10], allowing its delegation mechanism to be used with the more restrictive Blömer and Bobolz model.

In the next subsection, we give an overview of the security proofs for the above construction, which can be found in full in the appendices. Similar proofs should be obtainable for the variable-length Crites-Lysyanskaya scheme and the Connolly et al. scheme with minimal changes to the proof strategy.

4.3 Security analysis

In order to assess the security of this scheme, we must first establish the security goals of a delegatable, attribute-based credential (DABC): namely correctness, unforgeability, and anonymity. In the context of this scheme, these can loosely be defined as follows:

- **Correctness:** The scheme is correct if, whenever Setup and KeyGen are run correctly and the Issue-Receive protocol is executed correctly on correctly generated inputs including an attribute set $\mathcal{X}$, the receiver outputs a certification chain that, when used as input to the prover in an honest execution of the CredProve-CredVerify protocol with input sets $\mathcal{S}$ and $\mathcal{D}$ such that $\mathcal{S} \subseteq \mathcal{X}$ and $\mathcal{D} \cap \mathcal{X} = \emptyset$, is accepted by the verifier with probability 1.
- **Unforgeability:** The scheme is unforgeable if:
  - a (non-root) user without a correctly-issued credential cannot perform a showing or issue a credential that would be accepted by a verifier with non-negligible probability
  - a user in possession of a credential for an attribute set $\mathcal{X}$ cannot with non-negligible probability perform a valid showing for sets $\mathcal{S}$ and $\mathcal{D}$ such that $\mathcal{S} \not\subseteq \mathcal{X}$ and $\mathcal{D} \cap \mathcal{X} \neq \emptyset$, even if colluding with other users.
  - a user in possession of a credential for an attribute set $\mathcal{X}$ cannot with non-negligible probability issue a valid delegated credential for a set $\mathcal{Y} \not\subseteq \mathcal{X}$, even if colluding with other users.

- **Anonymity:** The scheme is anonymous if, during a showing of a credential, no verifier, issuer, or coalition of verifiers and issuers can identify the user, identify past showings of the same credential, or learn anything about the user other than that they possess a valid credential for the attributes being shown.

Note that the CL19 and CL21 schemes [6, 7] cannot satisfy anonymity in cases where the credential chain includes a public key (other than the root issuer) whose secret key is known by the adversary. Such cases are therefore eliminated in the security game that provides the formal criteria.

In order to formalise the security definitions, we extend the security game from Crites and Lysyanskaya's DAC model [6] to account for the addition of attributes to the credentials, and of subset and disjoint set openings to the showing protocol. The resulting security game can be found in Appendix A.

Correctness: The correctness of the scheme in section 4.2 follows by inspection. In particular, it can be seen that it matches Crites and Lysyanskaya's original mercurial signature scheme [6], expanded to make the contents of each credential explicit, and with the addition of attribute sets, attribute witnesses, and verification checks on attributes.

Unforgeability: The proof of unforgeability for the scheme in Section 4.2 works by enumerating the ways in which an adversary could achieve a forgery in the attribute-based security game, and shows how each reduces to either a forgery in the non-attribute-based mercurial signature model or a breaking of one of the security properties of the set commitment scheme. The full proof can be found in Appendix B.

Anonymity: The proof of anonymity for the scheme in Section 4.2 is an extension of the hybrid argument in Crites and Lysyanskaya's proof of anonymity for the original mercurial signature construction, which shows that the adversary's view when attributes are included is indistinguishable from the case in which all honest parties have identical attributes; therefore the inclusion of credential attributes and subset witnesses does not enable the adversary to distinguish between credentials. The full proof can be found in Appendix C.

5 Conclusion and future work

In this paper, we have introduced a method by which mercurial signatures can encode and delegate multiple attributes of the credential holder, in keeping with the properties of selective disclosure and restricted, selective delegation as described by Blömer and Bobolz. This is an extremely important feature for an anonymous credential system, providing far more versatility than a credential that does not encode any detailed attribute information, and so it represents a significant step toward bringing mercurial signature credentials into line with the functionality available from older credential schemes.

There are still other major avenues for improving mercurial signature credentials, however. Most notably, the CL19 mercurial signature scheme suffers from a severe reduction in its anonymity property, resulting from the fact that an adversary that has delegated a credential can subsequently recognise its own key in that credential's delegation chain. This weakness is addressed by the work of Connolly et al. [4], who

use a different key structure in their scheme, but this still only solves it in the honest parameter model. In theory, a zero-knowledge proof of honest parameter generation given at the time of delegation would close this weakness, but further work is needed to accomplish this.

Mercurial signatures also have yet to be extended to a multi-authority model enabling their seamless use in systems with multiple issuing authorities, and there is interest in finding other basic constructions from which mercurial signatures could be developed, particularly any based on quantum-secure assumptions.

References

[1] Johannes Blömer and Jan Bobolz. Delegatable attribute-based anonymous credentials from dynamically malleable signatures. In Bart Preneel and Frederik Vercauteren, editors, *Applied Cryptography and Network Security*, pages 221–239, Cham, 2018. Springer International Publishing.

[2] Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In Birgit Pfitzmann, editor, *Advances in Cryptology — EUROCRYPT 2001*, pages 93–118, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[3] Melissa Chase and Anna Lysyanskaya. On signatures of knowledge. In Cynthia Dwork, editor, *Advances in Cryptology - CRYPTO 2006*, pages 78–96, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[4] Aisling Connolly, Pascal Lafourcade, and Octavio Perez Kempner. Improved constructions of anonymous credentials from structure-preserving signatures on equivalence classes. In Goichiro Hanaoka, Junji Shikata, and Yohei Watanabe, editors, *Public-Key Cryptography – PKC 2022*, pages 409–438, Cham, 2022. Springer International Publishing.

[5] Ronald Cramer, Ivan Damgård, and Philip MacKenzie. Efficient zero-knowledge proofs of knowledge without intractability assumptions. In Hideki Imai and Yuliang Zheng, editors, *Public Key Cryptography*, pages 354–372, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

[6] Elizabeth C. Crites and Anna Lysyanskaya. Delegatable anonymous credentials from mercurial signatures. In Mitsuru Matsui, editor, *Topics in Cryptology – CT-RSA 2019*, pages 535–555, Cham, 2019. Springer International Publishing.

[7] Elizabeth C Crites and Anna Lysyanskaya. Mercurial signatures for variable-length messages. In *Proceedings on Privacy Enhancing Technologies (PoPETs)*, pages 441–463. de Gruyter, 2021.

[8] 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, 2019.

[9] Christian Hanser and Daniel Slamanig. Structure-preserving signatures on equivalence classes and their application to anonymous credentials. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology – ASIACRYPT 2014*, pages 491–511, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[10] Omid Mir, Daniel Slamanig, Balthazar Bauer, and Rene Mayrhofer. Practical delegatable anonymous credentials from equivalence class signatures. In *Proceedings on Privacy Enhancing Technologies (PoPETs)*, pages 488–513. PETS, 2023.

[11] Colin Putman and Keith M. Martin. Selective delegation of attributes in mercurial signature credentials. In Elizabeth A. Quaglia, editor, *Cryptography and Coding*, pages 181–196, Cham, 2024. Springer Nature Switzerland.

A Security game for delegatable attribute-based credentials

Unlike in Crites and Lysyanskaya's scheme, it is necessary in this scheme's setup to specify a maximum length for credential chains. For the purposes of the security game, this can be assumed to be large enough that it does not obstruct the adversary's normal operations, while ignoring adversarial strategies that rely on absurd parameters, as reducing the maximum credential length only imposes tighter restrictions on the adversary. Similarly, setting the maximum cardinality for attribute sets to be arbitrarily high maximises the information available to the adversary.

The intuition of this security game is that the adversary $\mathcal{A}$ interacts with a challenger $\mathcal{C}$ whose role is to simulate all of the participants in a single-authority delegatable attribute-based credential system using the mercurial signature construction in section 4.2. The challenger's internal state includes a graph that models each participant as a node, with directed edges corresponding to credentials that have been issued or delegated.

$\mathcal{A}$ interacts with $\mathcal{C}$ by invoking a set of oracles which prompt $\mathcal{C}$ to simulate actions on behalf of the nodes in its credential system, using inputs stored in its internal state for honest nodes and inputs provided by $\mathcal{A}$ for adversarial nodes as appropriate. $\mathcal{A}$ aims to achieve one of two objectives: either it must cause $\mathcal{C}$ to flag a forgery, or it must choose two honest nodes in the system and successfully determine which is which from their anonymised outputs. The scheme is considered broken if a PPT adversary can achieve the first goal with non-negligible probability or the second goal with non-negligible advantage.

The oracles available to $\mathcal{A}$ and their purposes are as follows:

- AddHonestParty: Invoked to add a new, honest node to the graph.
- SeeNym: Invoked to create a new pseudonym (i.e. randomised public key) for an honest node and show it to $\mathcal{A}$, who can later use it as an input to other oracles.
- CertifyHonestParty: Invoked to have one honest node issue or delegate a credential to another. $\mathcal{A}$ can specify the attributes of this credential, as long as the issuing node is permitted to certify them.
- VerifyCredFrom: Invoked to have an honest node show its credential to $\mathcal{A}$ using the Prove/Verify protocol. $\mathcal{A}$ can specify the subset and disjoint set it wants the honest node to prove.
- GetCredFrom: Invoked to create an adversarial node and have an honest node issue or delegate a credential it. $\mathcal{A}$ receives the credential.
- GiveCredTo: Invoked to allow $\mathcal{A}$, under the identity of a (potentially fresh) adversarial node, to issue or delegate a credential to an honest node.
- DemoCred: Invoked to allow $\mathcal{A}$, under the identity of a (potentially fresh) adversarial node, to prove possession of a credential using the Prove/Verify protocol.
- SetAnonChallenge: Invoked to choose two honest nodes for $\mathcal{A}$ to attempt to distinguish between. A random bit is set by $\mathcal{C}$ to determine the order in which the nodes will act in future anonymised interactions, and $\mathcal{A}$'s goal is to guess the random bit.
- VerifyCredFromAnon: Invoked to have the chosen anonymity nodes both perform showings of their credentials to $\mathcal{A}$, using fresh pseudonyms unknown to $\mathcal{A}$, in

the order determined by the random bit. The showings provide the maximum amount of information about the intersection of the nodes' attribute sets.

- GetCredFromAnon: Invoked to have the chosen anonymity nodes both delegate credentials to $\mathcal{A}$, using fresh pseudonyms unknown to $\mathcal{A}$, in the order determined by the random bit. $\mathcal{A}$ chooses the attribute set to be issued, with the requirement that both anonymity nodes must be able to issue that set.
- GuessAnon: Invoked only once by $\mathcal{A}$ to guess the random anonymity bit set by SetAnonChallenge.

$\mathcal{C}$ flags a forgery if $\mathcal{A}$ causes an adversarial node to successfully show or delegate a credential whose credential chain does not correspond to a legitimate path in $\mathcal{C}$'s graph, or whose attributes are not a subset of the attributes of earlier honest credentials in the chain. Note that, for the purposes of finding a path in the graph, edges can be added from any adversarial node to any other, representing the fact that participants under the adversary's control can issue credentials to each other without operating within the system's constraints. This flag therefore corresponds either to a case where, at some point in the credential chain, $\mathcal{A}$ produces an apparent credential from an honest node that never issued such a credential, or to a case where $\mathcal{A}$ manages in some way to subvert the subset witness introduced in section 4.1 and produce a credential with unauthorised attributes; either of these represents a breach of the unforgeability property.

The objective for which $\mathcal{A}$ attempts to distinguish between two anonymised nodes corresponds to the anonymity property; after choosing nodes to distinguish between, $\mathcal{A}$ has the opportunity to see credential showings and issuances from both under their anonymised identities before guessing which is which. The choice of nodes is constrained to exclude those which would be trivial to distinguish, such as those on different levels or those which have an adversarial key on their credential chain, as per the weakened anonymity notion inherited from Crites and Lysyanskaya. Any advantage therefore implies the ability to link the anonymised showings to any non-anonymised interactions the adversary has seen from the chosen nodes.

The anonymity game has also been streamlined from Crites and Lysyanskaya's version of the game by moving the forfeit check to the moment when the challenge is set, expecting that the adversary will choose the challenge nodes after setting up their credentials, rather than before. This change is intended only to reduce the number of oracles needed by the game. Any attack that could be made against Crites and Lysyanskaya's game by choosing nodes and then setting up their credentials anonymously can still be carried out by setting up those credentials first and then choosing the nodes and anonymising them.

The full behaviour of the challenger $\mathcal{C}$ and its oracles now follows, with changes from Crites and Lysyanskaya's game coloured. The challenger has access to hard-to-compute functions f, f_{cred}, and f_{demo}, used to recover participant identities from credentials and showings, and maintains the following state information:

1. A directed graph $G(\mathsf{pk}_0) = (V(\mathsf{pk}_0), E(\mathsf{pk}_0))$ consisting of a single tree and singleton nodes. The root node of the tree is called root and has public key pk_0.
2. For each node $v \in V(\mathsf{pk}_0)$,
   (a) v's level $L(v)$ (i.e. v's distance from root).
   (b) $status(v)$, which is either *honest* or *adversarial*.
   (c) If $status(v) = honest$,

i. $(\mathsf{pk}(v)_{i\in[\lambda]})$, the public keys associated with v for each possible level.

ii. $(\mathsf{sk}(v)_{i\in[\lambda]})$, the secret keys corresponding to $(\mathsf{pk}(v)_{i\in[\lambda]})$.

iii. All pseudonyms (randomised public keys) $\mathsf{nym}_1(v), ..., \mathsf{nym}_n(v)$ associated with v, and $\mathsf{aux}_1(v), ..., \mathsf{aux}_n(v)$, their corresponding blinding factors.

iv. The node's credential cred_v, if it exists, along with its attribute set A_v and commitment opening information O_v.

v. An identifying value $\hat{\mathsf{pk}}_v$ determined using the function f, such that $\hat{\mathsf{pk}}_v = f(\mathsf{pk}_v) = f(\mathsf{nym}_i(v))$ for all $i \in [n]$.

(d) If $status(v) = adversarial$, a value $\hat{\mathsf{pk}}_v$ such that $\hat{\mathsf{pk}}_v = f(\mathsf{nym}(v))$ for any pseudonym $\mathsf{nym}(v)$ used by the adversary for the node v, and the attribute set A_v associated with the node's credential if one exists.

3. A forgery flag.

4. An anonymity bit $b \in \{0, 1\}$, a pair of challenge nodes (u_0, u_1) and the set of pseudonyms $\mathcal{A}$ has seen for them while interacting anonymously, and the status of the anonymity attack ($undefined, success, fail$, or $forfeit$).

The game is initialised as follows. The parameters $params$ are generated and given to $\mathcal{A}$. $\mathcal{A}$ then specifies whether $status(\mathsf{root})$ is $honest$ or $adversarial$. If it is $honest$, $\mathcal{C}$ generates $(\mathsf{pk}_0, \mathsf{sk}_0) \leftarrow \mathsf{KeyGen}(params)$; otherwise, $\mathcal{A}$ supplies pk_0 to $\mathcal{C}$. $\mathcal{C}$ sets the forgery flag to false, picks a random value for the anonymity bit b, and sets the status of the anonymity challenge to $undefined$. $\mathcal{C}$ stores $G(\mathsf{pk}_0) = (V(\mathsf{pk}_0), E(\mathsf{pk}_0)) = (\{\mathsf{root}\}, \emptyset)$, with $status(\mathsf{root})$ set as specified by $\mathcal{A}$, $\mathsf{pk}(\mathsf{root}) = \mathsf{pk}_0$, $\mathsf{sk}(\mathsf{root}) = \mathsf{sk}_0$ if root is $honest$, and $L(\mathsf{root}) = 0$.

$\mathcal{A}$ can then interact with $\mathcal{C}$ using the following oracles:

- $\mathsf{AddHonestParty}(u, l)$: $\mathcal{A}$ invokes this oracle to create a new, honest node u. $\mathcal{C}$ generates $(\mathsf{pk}(u), \mathsf{sk}(u)) \leftarrow \mathsf{KeyGen}_l(params)$, sets $L(u) = l$, and returns $\mathsf{pk}(u)$ to $\mathcal{A}$. $\mathcal{A}$ supplies the new node's delegation level in advance for convenience, since key length depends upon delegation level in this scheme; it would otherwise be fixed when the node received a credential, and determined by $\mathcal{A}$'s choice of which node issues that credential.

- $\mathsf{SeeNym}(u)$: $\mathcal{A}$ invokes this oracle to see a fresh pseudonym for an honest node u. $\mathcal{C}$ samples a random $\rho \leftarrow \mathbb{Z}_p$, stores $\mathsf{nym}(u) = \mathsf{pk}(u)^\rho$ and $\mathsf{aux}(u) = \rho$ together as a new pseudonym for u, and returns $\mathsf{nym}(u)$ to $\mathcal{A}$.

- $\mathsf{CertifyHonestParty}(u, v, \mathsf{S})$: $\mathcal{A}$ invokes this oracle to have the honest node u issue a credential to the honest node v on the attribute set S. If $L(v) = L(u) + 1$, $\mathcal{C}$ runs $[\mathsf{Issue}(params, L(u), \mathsf{pk}_0, \mathsf{sk}(u), \mathsf{pk}(u), O_u, \mathsf{cred}_u, \mathsf{A}_u, \mathsf{S}) \leftrightarrow \mathsf{Receive}(params, L(u), \mathsf{pk}_0, \mathsf{sk}(v)_{L(u)+1}, \mathsf{pk}(v)_{L(u)+1}, \mathsf{S})] \rightarrow (\mathsf{cred}_v, O_v)$. If the protocol succeeds, $\mathcal{C}$ adds the edge (u, v) to the graph, and sets $\mathsf{A}_v = \mathsf{S}$.

- $\mathsf{VerifyCredFrom}(u, \mathsf{S}, \mathsf{D})$: $\mathcal{A}$ invokes this oracle to have the honest node u prove to $\mathcal{A}$ that it has a credential at level $L(u)$ including the attributes in the set S and not those in the set D. $\mathcal{C}$ runs $\mathsf{CredProve}(params, L(u), \mathsf{sk}(u), \mathsf{pk}(u), O_u, \mathsf{cred}_u, \mathsf{A}_u, \mathsf{S}, \mathsf{D}) \leftrightarrow \mathcal{A}$.

- $\mathsf{GetCredFrom}(u, \mathsf{nym}_R, \mathsf{S})$: $\mathcal{A}$ invokes this oracle to have the honest node u issue a credential on the attribute set S to an adversarial pseudonym nym_R. $\mathcal{C}$ creates a new adversarial node v, sets its identity to be $\hat{\mathsf{pk}}(v) = f(\mathsf{nym}_R)$, and runs $\mathsf{Issue}(params, L(u), \mathsf{pk}_0, \mathsf{sk}(u), \mathsf{pk}(u), O_u, \mathsf{cred}_u, \mathsf{A}_u, \mathsf{S}) \leftrightarrow \mathcal{A}$, storing

the pseudonym generated for u by randomising u's credential unless u is root. If the protocol succeeds, $\mathcal{C}$ then adds the edge (u, v) to the graph and stores $L(v) = L(u) + 1$.

- $\mathsf{GiveCredTo}(L_I, \mathsf{nym}_I, v, \mathsf{S})$: $\mathcal{A}$ invokes this oracle to issue a credential on attribute set S to an honest node v under the pseudonym nym_I (which may be pk_0 if $status(\mathsf{root})$ is $adversarial$). If $L(v) \neq L_I + 1$, $\mathcal{C}$ aborts; otherwise $\mathcal{C}$ runs $[\mathcal{A} \leftrightarrow \mathsf{Receive}(params, L_I, \mathsf{pk}_0, \mathsf{sk}(v), \mathsf{pk}(v), \mathsf{S})] \rightarrow (\mathsf{cred}_v, O_v)$. If $\mathsf{cred}_v \neq \perp$, $\mathcal{C}$ computes $f_{cred}(\mathsf{cred}_v) = (\hat{\mathsf{pk}}_0, \hat{\mathsf{pk}}_1, ..., \hat{\mathsf{pk}}_{L_I})$ to reveal the identities of the nodes on the credential chain. If according to $\mathcal{C}$'s data structure there is some $\hat{\mathsf{pk}}_i$ in the chain such that $\hat{\mathsf{pk}}_i = f(\mathsf{nym}(u))$ for an honest node u but $\hat{\mathsf{pk}}_{i+1} \neq f(\mathsf{nym}(v'))$ for any v' that received a credential from u, then $\mathcal{C}$ sets the $\mathsf{forgery}$ flag to true. If $\hat{\mathsf{pk}}_{L_I} = f(\mathsf{nym}(u))$ for some honest node u (that is, the last credential in the chain was issued honestly), and $\mathsf{S} \not\subseteq \mathsf{A}_{L_I}$, $\mathcal{C}$ sets the $\mathsf{forgery}$ flag to true. Otherwise, $\mathcal{C}$ identifies the nearest honest ancestor u' of v in the chain, then finds all adversarial nodes associated with the next identity after u' on the chain. If $\mathsf{S} \not\subseteq \mathsf{S}'$ for every attribute set S' associated with a credential one of these adversarial nodes received from u', $\mathcal{C}$ sets the $\mathsf{forgery}$ flag to true. If $\mathsf{cred}_v \neq \perp$ and the $\mathsf{forgery}$ flag remains false, $\mathcal{C}$ fills in the graph as follows: starting from the nearest honest ancestor of v in the chain (or root if there is no honest ancestor), $\mathcal{C}$ creates a new node for each (adversarial) identity in the chain between that node and v, setting each node's identity to the appropriate $\hat{\mathsf{pk}}_j$ and each node's attribute set to S. $\mathcal{C}$ then adds edges between each new node and its immediate neighbours in the chain.
- $\mathsf{DemoCred}(L_P, \mathsf{nym}_P, \mathsf{S}, \mathsf{D})$: $\mathcal{A}$ invokes this oracle to prove possession of a credential at level L_P with an attribute set including the set S and excluding the set D. $\mathcal{C}$ runs $[\mathcal{A} \leftrightarrow \mathsf{CredVerify}(\mathsf{pk}_0)] \rightarrow \mathsf{output}$ (0 or 1). If $\mathsf{output} = 1$, $\mathcal{C}$ computes $f_{demo}(transcript) = (\hat{\mathsf{pk}}_0, \hat{\mathsf{pk}}_1, ..., \hat{\mathsf{pk}}_{L_P})$ to reveal the identities on the demonstrated credential chain, then determines whether a forgery has occurred by the same process as in $\mathsf{GiveCredTo}$; in the case that the credential being demonstrated was issued by an honest node, $\mathcal{C}$ also checks whether $\mathsf{D} \cap \mathsf{A}_{L_P} = \emptyset$, where A_{L_P} is the attribute set issued with that credential, and sets the $\mathsf{forgery}$ flag to true if the check fails. If $\mathsf{output} = 1$ and the $\mathsf{forgery}$ flag remains false, $\mathcal{C}$ creates a new adversarial node v for the identity nym_P and sets $L(v) = L_P$. $\mathcal{C}$ then fills in the graph by the same process as in $\mathsf{GiveCredTo}$.
- $\mathsf{SetAnonChallenge}(u_0, u_1)$: $\mathcal{A}$ invokes this oracle to choose two distinct, honest nodes u_0 and u_1 which it will attempt to distinguish between for the anonymity challenge. $\mathcal{C}$ checks that the chosen nodes are both honest with $u_0 \neq u_1$, and that both possess credentials such that $L(u_0) = L(u_1)$ and neither credential has an adversarial node on its chain other than root, which $\mathcal{C}$ can check using f_{cred}. If these checks all succeed, $\mathcal{C}$ sets these nodes as the anonymity challenge pair.
- $\mathsf{VerifyCredFromAnon}()$: $\mathcal{A}$ invokes this oracle to have the nodes u_0 and u_1 prove possession of their credentials to $\mathcal{A}$, provided that the anonymity challenge has been set. $\mathcal{C}$ runs $[\mathsf{CredProve}(params, L(u_b), \mathsf{sk}(u_b), \mathsf{pk}(u_b),$
$O_{u_b}, \mathsf{cred}_{u_b}, \mathsf{A}_{u_b}, \mathsf{S}, \mathsf{D}) \leftrightarrow \mathcal{A}]$, followed by the same protocol but for $u_{\bar{b}}$, where $\mathsf{S} = \mathsf{A}_{u_0} \cap \mathsf{A}_{u_1}$ and $\mathsf{D} = \bar{\mathsf{A}}_{u_0} \cap \bar{\mathsf{A}}_{u_1}$.
- $\mathsf{GetCredFromAnon}(b^*, \mathsf{nym}_R, \mathsf{S})$: $\mathcal{A}$ invokes this oracle to receive a credential from the node u_{b^*}, where b^* indicates either b or $\bar{b}$, provided that the anonymity

challenge has been set. If $\mathsf{S} \subseteq \mathsf{A}_{u_0} \cap \mathsf{A}_{u_1}$, $\mathcal{C}$ creates a new adversarial node v with identity $\hat{\mathsf{pk}}_v = f(\mathsf{nym}_R)$ and runs $[\mathsf{Issue}(params, L(u_{b^*}), \mathsf{pk}_0, \mathsf{sk}(u_{b^*}), \mathsf{pk}(u_{b^*}), O_{u_{b^*}}, \mathsf{cred}_{u_{b^*}}, \mathsf{A}_{u_{b^*}}, \mathsf{S}) \leftrightarrow \mathcal{A}]$. $\mathcal{C}$ then adds the edge (u_{b^*}, v) to the graph and sets $L(v) = L(u_{b^*}) + 1$.

- $\mathsf{GuessAnon}(b')$: If $b' = b$, the status of the anonymity attack is set to *success*. Otherwise, it is set to $fail$.

With this game in place, the unforgeability and anonymity security properties for delegatable attribute-based credentials are formally stated as follows:

A delegatable attribute-based credential scheme is unforgeable if there exist functions $f, f_{\mathsf{cred}}, f_{demo}$ such that for all PPT $\mathcal{A}$, there exists a negligible function υ such that the probability that the $\mathsf{forgery}$ flag will be set to true in the single-authority game is at most $\upsilon(k)$, where k is the security parameter.

A delegatable attribute-based credential scheme is anonymous if there exist functions f, f_{cred}, f_{demo} such that for all PPT $\mathcal{A}$, there exists a negligible function υ such that the probability that the status of the anonymity attack in the single-authority game will be *success* is at most $1/2 + \upsilon(k)$, where k is the security parameter.

B Unforgeability proof

Theorem 1. *If the underlying mercurial signature scheme is unforgeable, the set commitment scheme is secure, and the groups $\mathcal{G}_1$ and $\mathcal{G}_2$ are DDH-hard, the attribute-based mercurial signature scheme is unforgeable.*

Proof. The unforgeability of a DABC scheme could be broken in several ways, with each case requiring its own proof. Specifically, the adversary could provide a credential chain including a credential from an honest node that never issued such a credential (case 1), a credential whose attribute commitment has been changed since it was issued (case 2), or a credential chain that verifies successfully despite at least one credential having an associated attribute set that is not a subset of a preceding credential in the chain (case 3).

By considering each of these in turn, it can be shown that any PPT adversary $\mathcal{A}$ capable of breaking the unforgeability of this scheme can be used in a reduction $\mathcal{B}$ either to break the unforgeability of Crites-Lysyanskaya mercurial signatures or to break the security properties of extended FHS19 set commitments.

In case 1, Crites and Lysyanskaya's original proof of unforgeability [6] still holds; the reduction is capable of performing all of the functions added by this scheme when acting as the challenger for $\mathcal{A}$, even without knowledge of the challenge secret key for its own forgery game, since a node's secret key is not directly needed to generate its attributes, commitments, or witnesses.

In case 2, the adversary can be used by the same signature unforgeability reduction $\mathcal{B}$ used in the proof against credential forgery. If $\mathcal{B}$ does not detect a credential forgery, it identifies the nearest (honest) node to the end of the chain for which $\mathcal{B}$ knows the secret key (taking advantage of the weakened anonymity notion to recognise known keys on the chain).

If the node identified has issued a credential to u^*, the randomly-chosen honest node to which $\mathcal{B}$ has assigned the challenge key it is attempting to learn, $\mathcal{B}$ assumes with probability 1/2 that the next credential on the chain belongs to u^*; otherwise $\mathcal{B}$ assumes the next credential is from an adversarial node. If $\mathcal{B}$ makes the wrong assumption, $\mathcal{A}$ may learn that it is operating within the reduction, causing the reduction to fail.

$\mathcal{B}$ then checks whether there has been an attribute forgery in the same way as the challenger $\mathcal{C}$ would. If a forgery is detected and $\mathcal{B}$ has not assumed that u^* is on the chain, the reduction fails as this forgery is not associated with $\mathcal{B}$'s challenge key. If a forgery is detected with u^* assumed to be the last honest node on the chain, $\mathcal{B}$ outputs $(\mathsf{pk}^*, M^*, \sigma*) = (\mathsf{nym}_{u^*}, (\mathsf{nym}_R, C_R), \sigma_R)$, where (nym_R, C_R) is the pseudonym and commitment associated with the next credential in the chain after u^*, and σ_R is the signature on that credential.

Assuming WLOG that there is an a priori upper bound q_a on the number of calls the adversary can make to the AddHonestParty oracle, and considering that the identity of u^* is independent of the adversary's view, there is at least a $1/q_a$ probability that the last honest node in the credential chain is u^* when a forgery occurs. In this case, there is a further 1/2 probability that $\mathcal{B}$ will correctly assume u^*'s identity. If u^* is the last honest node on the chain, $\mathcal{B}$ correctly assumes its identity, and $\mathcal{A}$ achieves a forgery using a change to the set commitment in a credential issued by an honest node, $\mathcal{B}$'s output will be a successful answer to the unforgeability game for mercurial signatures.

For case 3, in which $\mathcal{A}$ achieves an attribute forgery without forging signed data,

consider $\mathcal{A}$ running within a different PPT simulation of the challenger, $\mathcal{B}'$, which has access to a valid set of *params* which it uses to set up the game, but does not know the trapdoor a for those *params*. Unlike $\mathcal{B}$, $\mathcal{B}'$ knows all secret keys associated with honest nodes; it can therefore be seen that $\mathcal{B}'$ can simulate all of the functions of $\mathcal{C}$ except for f, f_{cred}, and f_{demo}, which are outside $\mathcal{A}$'s view, and can identify each honest node on a credential chain in $\mathrm{O}(q_a)$ time using the weakened anonymity notion.

For $\mathcal{A}$ to achieve an attribute forgery without forging a signature, either the subset S or disjoint set D that $\mathcal{A}$ is disclosing or issuing must not match the attribute set A of the credential being used (case 3*a*), or there must be some adversarial node v after the last honest node in the credential chain such that the preceding witness tag W_{L_v-1} verifies successfully but either $\mathcal{A}$ cannot open v's commitment C_v to a subset of the previous credential's attributes or $\mathcal{A}$ knows multiple valid openings of C_v (case 3*b*).

In case 3*a*, we have that the final (adversarial) credential in the chain has a commitment C_A on some attribute set A, and the adversary either issues a credential on an attribute set $\mathsf{S} \not\subseteq \mathsf{A}$ or performs a subset opening to $\mathsf{S} \not\subseteq \mathsf{A}$ and/or disjoint opening to $\mathsf{D} \cap \mathsf{A} \neq \emptyset$.

To issue a credential in this way, $\mathcal{A}$ first gets a commitment $\hat{C}_R$ to S from the (honest) receiver, along with an encoding $\hat{P}^{O_R}$ of the corresponding opening information, and then must produce a partial witness tag $\bar{W}_R$ such that $e(\hat{C}_R, \bar{W}_R) = e(\hat{P}^{O_R}, C_A)$. Since the receiver is honest, $\hat{C}_R = (\hat{P}^{O_R})^{f_S(a)}$, and so this pairing equation is equivalent to $e(\hat{P}^{f_S(a)}, \bar{W}_R) = e(\hat{P}, C_A)$, and therefore $\mathsf{VerifySubset}(params, C_A, \mathsf{S}, \bar{W}_R) = 1$. This implies that $\mathcal{A}$ could break the (subset) soundness property of set commitments by submitting $(C_A, \mathsf{A}, O_A, \mathsf{S}, \bar{W}_R)$, where O_A is $\mathcal{A}$'s opening information for C_A.

Similarly, to perform a subset opening or a disjoint opening, $\mathcal{A}$ must provide a witness W_A satisfying $\mathsf{VerifySubset}(params, C_A, \mathsf{S}, W_A) = 1$ or $\mathsf{VerifyDisjoint}(params, C_A, \mathsf{D}, W_A) = 1$, respectively. If $\mathsf{S} \not\subseteq \mathsf{A}$ or $\mathsf{D} \cap \mathsf{A} \neq \emptyset$, this likewise implies that $\mathcal{A}$ could break the soundness property of set commitments.

These outcomes represent breaking the soundness property if and only if $\mathcal{A}$ does not have access to the trapdoor a. This must be the case, as $\mathcal{A}$ cannot distinguish whether or not it is running within the simulation $\mathcal{B}'$, which itself does not know a and so cannot leak it to $\mathcal{A}$.

In case 3*b*, where $\mathcal{A}$'s forgery is on the credential chain rather than in the final witness value, there must be at least one adversarial credential with an attribute commitment that $\mathcal{A}$ either cannot open to a subset of the previous credential's attribute set, or can open to multiple distinct sets, since if all commitments on the chain open only to a subset of the previous committed set then no forgery has taken place.

Given that $\mathcal{A}$ does not know the trapdoor a, if $\mathcal{A}$ is able to open a commitment to two different sets, it trivially has a solution to the computational binding game for set commitments. If, on the other hand, there is an adversarial node v whose commitment C_v $\mathcal{A}$ cannot open to a subset of the set committed to by the previous commitment $\hat{C}_I$, $\mathcal{A}$ must be able to construct a witness tag $\hat{W}$ that still satisfies $e(C_v, \hat{W}) = e(P, \hat{C}_I)$.

If $\mathcal{A}$ can find a solution when C_v commits to A_v, $\hat{C}_I$ commits to A_I, and $\mathsf{A}_v \not\subseteq \mathsf{A}_I$, then $(\hat{C}_I, \mathsf{A}_I, O_I, \mathsf{A}_v, \hat{W})$ is a solution to the soundness game for set commitments. This implies that either $\mathcal{A}$ can break the soundness property or $\mathcal{A}$ does not know all of these values and is therefore unable to open one or both commitments.

If there is any adversarial commitment that $\mathcal{A}$ is unable to open, there must be a last such commitment, C_A, from which $\mathcal{A}$ is able to show a valid subset witness W_A either for a declared subset S as part of $\mathsf{DemoCred}$ or for an openable commitment $\hat{C}_S$

in the next credential on the chain (which may also be adversarial).

For W_A to be accepted, it must satisfy $e(W_A, \hat{C}_S) = e(C_A, \hat{P})$ for the following credential case, or $e(W_A, \hat{P}^{f_S(a)}) = e(C_A, \hat{P})$ for the DemoCred case. This simplifies to a requirement that $C_A = W_A^{\psi f_S(a)}$, where ψ is the opening value for $\hat{C}_S$ in the following credential case, or 1 in the DemoCred case.

Note that if there is a following credential, its holder determines ψ; if the credential is honest, the challenger will choose ψ uniformly randomly, so $\mathcal{A}$ cannot predict it with non-negligible probability; we can therefore discount the honest credential case and assume that $\mathcal{A}$ has control of ψ.

If the credential preceding C_A also cannot be opened by the adversary, its witness must satisfy $e(C_A, \hat{W}_{A-1}) = e(P, \hat{C}_{A-1})$. Substituting the relationship found between C_A and W_A gives $e(W_A^{\psi f_S(a)}, \hat{W}_{A-1}) = e(P, \hat{C}_{A-1})$, which requires that $\hat{C}_{A-1} = \hat{W}_{A-1}^{w_A \psi f_S(a)}$, where $P^{w_a} = W_A$. This relationship can be extended similarly over an indefinite number of delegation levels.

While running GetCredFrom, $\mathcal{A}$ must prove the ability to open its commitment; there must therefore be a last adversarial commitment, C_{A_0}, before those the $\mathcal{A}$ cannot open, which opens to a set A. Since an attribute forgery must occur, we have $\mathsf{S} \not\subseteq \mathsf{A}$. For the credential chain to verify, it must be the case that $C_{A_0} = P^{\rho f_A(a)} = P^{w_{a_0} \ldots w_{a_n} \psi f_S(a)}$, where $P^{w_{a_0}}, ..., P^{w_{a_n}} = W_{A_0}, ..., W_{A_n}$ are the witness tags attached to n unopenable credentials. This gives the relationship $w_{a_0} ... w_{a_n} = \rho f_A(a) \psi^{-1} f_S(a)^{-1}$.

Cancelling out any overlapping attributes in A and S, it must be the case that one or more of the witness exponents $w_{a_0}, ..., w_{a_n}$ and/or $\rho\psi^{-1}$ include a term in $f_{S \setminus (A \cap S)}(a)^{-i}$ for some $0 < i \leq 1$, with $\mathcal{A}$ able to construct $\rho\psi^{-1}$ and $P^{w_{a_0}}, ..., P^{w_{a_n}}$. However, $\mathcal{A}$ cannot distinguish whether it is running within $\mathcal{B}'$, which does not know a or $P^{F(a)^{-i}}$ for any polynomial F and positive i and so cannot give $\mathcal{A}$ what it needs to construct these values.

$\mathcal{A}$ therefore cannot reliably construct a valid credential chain with unopenable commitments unless the result is still that later attribute sets are subsets of earlier ones, so any instance of a case 3 forgery corresponds to a solution to the soundness game or the computational binding game for set commitments. Combined with the reductions from cases 1 and 2 to breaking unforgeability of mercurial signatures, this completes the proof of unforgeability for attribute-based mercurial signature credentials as presented in section 4.2. □

C Anonymity proof

Theorem 2. *If the underlying mercurial signature scheme is anonymous, the set commitment scheme is secure, and the groups $\mathcal{G}_1$ and $\mathcal{G}_2$ are DDH-hard, the attribute-based mercurial signature scheme is unforgeable.*

Proof. Let $\Gamma(k)$ be a polynomial and λ be the a priori maximum number of delegation levels for an instantiation of the scheme. For $0 \leq i \leq \Gamma(k)$, let $\mathcal{H}_i$ be the hybrid experiment defined by the following modification to the single-authority security game.

Upon initialisation, the challenger generates λ fixed key pairs $(\mathsf{sk}_1, \mathsf{pk}_1)$, ..., $(\mathsf{sk}_\lambda, \mathsf{pk}_\lambda)$, such that each key pair $(\mathsf{sk}_l, \mathsf{pk}_l)$ can be used by a node u with $L(u) = l$. For an adversary $\mathcal{A}$'s j^{th} query to $\mathsf{AddHonestParty}$, $j \leq i$, parametrised with level $l \leq \lambda$, the challenger generates a new, honest node and generates its public key using $\mathsf{ConvertPK}(\mathsf{pk}_l)$. For the j^{th} query to $\mathsf{AddHonestParty}$, $j > i$, the challenger creates a new, honest node with its public key generated as in the unmodified security game.

Suppose that for (not necessarily easy-to-compute) functions f, f_{cred}, and f_{demo}, there exists a polynomial Γ and a PPT adversary $\mathcal{A}$ making at most $\Gamma(k)$ queries to the $\mathsf{AddHonestParty}$ oracle in the single-authority hybrid security game such that there exists a non-negligible $\epsilon(k)$ such that for some $i(k)$, $\mathcal{A}$ can distinguish $\mathcal{H}_{i(k)}$ from $\mathcal{H}_{i(k)+1}$ with advantage $\epsilon(k)$.

Using the same process as in the anonymity proof in the full version of Crites and Lysyanskaya's original mercurial signature paper [6], a PPT reduction $\mathcal{B}$ can be constructed to break public-key class-hiding of mercurial signatures, running $\mathcal{A}$ as a subroutine and acting as its challenger.

$\mathcal{B}$ receives as input the public parameters *params* and two public keys pk and pk', with the goal of determining whether or not these public keys are of the same equivalence class. $\mathcal{B}$ chooses some $l \leq \lambda$ and sets its fixed public key $\mathsf{pk}_l = \mathsf{pk}$, setting the key lengths in its scheme parameters such that pk is of the appropriate length for delegation level l.

$\mathcal{B}$ then operates as a normal challenger for the game $\mathcal{H}_{i(k)}$, using a zero-knowledge simulator to convince $\mathcal{A}$ that it knows the secret key for pk as is standard. As Crites and Lysyanskaya show, it is capable of performing all the challenger functions despite not having access to f, f_{cred}, and f_{demo}, as these functions are only used to maintain the edges of the graph which remains independent of the adversary's view unless the adversary produces a forgery, which can only occur with negligible probability.

When $\mathcal{A}$ makes its $(i+1)^{th}$ query to $\mathsf{AddHonestParty}$, unless it requests the new node's level be l, $\mathcal{B}$ aborts. If the requested level is l, $\mathcal{B}$ sets the new node's public key as pk', again using a zero-knowledge simulator to convince $\mathcal{A}$ that it knows the corresponding secret key.

This reduction differs from the original only in that it must handle attribute commitments, which it can do exactly as the challenger in the security game would, and in that it succeeds only if $\mathcal{A}$'s $(i+1)^{th}$ query to $\mathsf{AddHonestParty}$ is parametrised with the delegation level corresponding to the length of the challenge public keys pk and pk' that $\mathcal{B}$ is attempting to distinguish.

Since the length of $\mathcal{B}$'s challenge keys is independent of $\mathcal{A}$'s view, this failure occurs with probability $\frac{\lambda-1}{\lambda}$. In the remaining case, the $(i+1)^{th}$ honest node has underlying public key pk_l if and only if pk and pk' are of the same equivalence class, meaning that $\mathcal{B}$ is running $\mathcal{H}_{i+1}$ if they are equivalent and $\mathcal{H}_i$ otherwise.

Thus, $\mathcal{B}$'s probability of determining whether pk and pk' are in the same equivalence class is proportional to $\mathcal{A}$'s probability of distinguishing the hybrids, and $\mathcal{B}$ succeeds

with probability $\frac{\epsilon(k)}{\lambda}$, which is non-negligible if $\epsilon(k)$ is non-negligible. Therefore, assuming public-key class-hiding for mercurial signatures holds, none of the hybrids $\mathcal{H}_i$ can be distinguished from $\mathcal{H}_0$, the unmodified security game.

Now consider a further hybrid experiment, $\mathcal{H}_A$, defined by the following security game, the effect of which is to replace all attribute commitments with random group elements and use knowledge of the trapdoor a to forge subset witnesses. The game is initialised in the same way as $\mathcal{H}_{\Gamma(k)}$, maintains the same state information plus an additional value ρ_u for each honest node u, and provides the same oracles to $\mathcal{A}$, but behaves as follows when the oracles are called:

- $\mathsf{AddHonestParty}(u,\ l)$: If $u \in V(\mathsf{pk}_0)$ or $l > \lambda$, abort. Otherwise, sample a random $\rho_1, \rho_2 \leftarrow \mathbb{Z}_p$, add a new node u to the graph, set $status(u) = honest$, $\rho_u = \rho_2$, and $(\mathsf{sk}(u), \mathsf{pk}(u)) = (\mathsf{ConvertSK}(\mathsf{sk}_l, \rho_1), \mathsf{ConvertPK}(\mathsf{pk}_l, \rho_1))$, and return $\mathsf{pk}(u)$ to $\mathcal{A}$.
- $\mathsf{SeeNym}(u)$: If $u \notin V(\mathsf{pk}_0)$ or $status(u) \neq honest$, abort. Otherwise, sample a random $\rho \leftarrow \mathbb{Z}_p$, generate $(\mathsf{nym}(u),\ \mathsf{nym}(u)) \leftarrow (\mathsf{ConvertPK}(u, \rho),\ \rho)$, store $\mathsf{nym}(u)$ and $\mathsf{aux}(u)$ at u, and return $\mathsf{nym}(u)$ to $\mathcal{A}$.
- $\mathsf{CertifyHonestParty}(u,\ v,\ \mathsf{S})$: If $u \notin V(\mathsf{pk}_0)$, $v \notin V(\mathsf{pk}_0)$, $status(u) \neq honest$, $status(v) \neq honest$, $\mathsf{cred}_u = \perp \wedge L(u) \neq 0$, $\mathsf{cred}_v \neq \perp$, or $L(v) \neq L(u) + 1$, abort. Otherwise, if $L(u) = 0$, compute $\sigma \leftarrow \mathsf{Sign}_0(\mathsf{sk}_0, (\mathsf{pk}(v),$ $P^{\rho_v}))$ and set $\mathsf{cred}_v = (\mathsf{pk}(v),\ \hat{P}^{\rho_v},\ \sigma)$. If $L(u) > 0$, compute $(\mathsf{cred}'_u,\ \mathsf{sk}'(u),$ $\psi) \leftarrow \mathsf{RandCred}(\mathsf{cred}_u,\ \mathsf{sk}(u),\ \mathsf{pk}_0,\ L(u))$ and $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}'(u),\ (\mathsf{pk}(v),\ P^{\rho_v}))$, and set $\mathsf{cred}_v = (\mathsf{cred}'_u, P^{\psi\rho_u\rho_v^{-1}}, \mathsf{pk}(v), \hat{P}^{\rho_v}, \sigma)$. Add (u, v) to $E(\mathsf{pk}_0)$ and set $\mathsf{A}_v = \mathsf{S}$.
- $\mathsf{VerifyCredFrom}(u,\ \mathsf{S},\ \mathsf{D})$: If $u \notin V(\mathsf{pk}_0)$, $status(u) \neq honest$, $\mathsf{cred}_u = \perp$, $\mathsf{S} \not\subseteq \mathsf{A}_u$, or $\mathsf{D} \cap \mathsf{S} \neq \emptyset$, abort. Otherwise, run $\mathsf{CredProve}(params,\ L(u),\ \mathsf{sk}(u),\ \mathsf{pk}(u),\ (0,$ $\rho_u f_S(a)^{-1}),\ \mathsf{cred}_u,\ \mathsf{S},\ \mathsf{S},\ \mathsf{D}) \leftrightarrow \mathcal{A}$.
- $\mathsf{GetCredFrom}(u,\ \mathsf{nym}_R,\ \mathsf{S})$: If $u \notin V(\mathsf{pk}_0)$, $status(u) \neq honest$, $\mathsf{cred}_u = \perp$, $\mathsf{S} \not\subseteq \mathsf{A}_u$, or the length of nym_R is not suitable for $L(u)+1$, abort. Otherwise, add a new node v to $V(\mathsf{pk}_0)$ and set $status(v) = adversarial$, $\hat{\mathsf{pk}}(v) = f(\mathsf{nym}_R)$, and $L(v) = L(u)+1$, then run $\mathsf{Issue}(params,\ L(u),\ \mathsf{pk}_0,\ \mathsf{sk}(u),\ \mathsf{pk}(u),\ (0,\ \rho_u f_S(a)^{-1}),$ $\mathsf{cred}_u,\ \mathsf{S},\ \mathsf{S}) \leftrightarrow \mathcal{A}$, storing the pseudonym generated for u by randomising u's credential unless u is root. If the protocol succeeds, add the edge (u, v) to $E(\mathsf{pk}_0)$ and store $\mathsf{A}_v = \mathsf{S}$.
- $\mathsf{GiveCredTo}(L_I,\ \mathsf{nym}_I,\ v,\ \mathsf{S})$: If $v \notin V(\mathsf{pk}_0)$, $status(v) \neq honest$, $\mathsf{cred}_v \neq \perp$, or the length of nym_I is not suitable for level L_I, abort. Otherwise, run a variation on $[\mathcal{A} \leftrightarrow \mathsf{Receive}(params, L_I, \mathsf{pk}_0, \mathsf{sk}(v), \mathsf{pk}(v), \mathsf{S})] \rightarrow (\mathsf{cred}_v, O_v)$ in which (C_R, O_R) is not generated from $\mathsf{Commit}(params,\ \mathsf{A}_R)$, but is set to $(P^{\rho_v},\ (0,\ \rho_v f_S(a)^{-1}))$. If $\mathsf{cred}_v \neq \perp$, update the graph in the same way as in $\mathcal{H}_{\Gamma(k)}$, and store cred_v.
- $\mathsf{DemoCred}(L_P,\ \mathsf{nym}_P,\ \mathsf{S},\ \mathsf{D})$: Run $[\mathcal{A} \leftrightarrow \mathsf{CredVerify}(\mathsf{pk}_0)] \rightarrow \mathsf{output}$ (0 or 1). If $\mathsf{output} = 1$, update the graph as in $\mathcal{H}_{\Gamma(k)}$.
- $\mathsf{SetAnonChallenge}(u_0,\ u_1)$: If the anonymity game's status is $undefined$, $u_0, u_1 \in V(\mathsf{pk}_0)$, $status(u_0) = status(u_1) = honest$, $\mathsf{cred}_{u_0} \neq \perp$, $\mathsf{cred}_{u_1} \neq \perp$, and $L(u_0) = L(u_1)$, use f_{cred} to identify all nodes on the credential chains of u_0 and u_1. If all of these nodes (except potentially root) are honest, set u_0 and u_1 as the anonymity challenge nodes and choose a random $b \in \{0, 1\}$ to store.
- $\mathsf{VerifyCredFromAnon}()$: If the anonymity game's status is not $undefined$ or u_0 and u_1 have not been set, abort. Otherwise, set $\mathsf{S} = \mathsf{A}_{u_0} \cap \mathsf{A}_{u_1}$ and $\mathsf{D} = \bar{\mathsf{A}}_{u_0} \cap \bar{\mathsf{A}}_{u_1}$,

and run [$\mathsf{CredProve}(params, L(u_b), \mathsf{sk}(u_b), \mathsf{pk}(u_b), (0, \rho_{u_b} f_S(a)^{-1}), \mathsf{cred}_{u_b}, \mathsf{S}, \mathsf{S}, \mathsf{D}) \leftrightarrow \mathcal{A}$], followed by the same protocol but for $u_{\bar{b}}$.

- $\mathsf{GetCredFromAnon}(b^*, \mathsf{nym}_R, \mathsf{S})$: If the anonymity game's status is not $undefined$, u_0 and u_1 have not been set, $\mathsf{S} \not\subseteq \mathsf{A}_{u_0} \cap \mathsf{A}_{u_1}$, or the length of nym_R is not suitable for $L(u_b) + 1$, abort. Otherwise, create a new adversarial node v with identity $\hat{\mathsf{pk}}_v = f(\mathsf{nym}_R)$ and $L(v) = L(u_b) + 1$, and run [$\mathsf{Issue}(params, L(u_{b^*}), \mathsf{pk}_0, \mathsf{sk}(u_{b^*}), \mathsf{pk}(u_{b^*}), (0, \rho_{u_{b^*}} f_S(a)^{-1}), \mathsf{cred}_{u_{b^*}}, \mathsf{S}, \mathsf{S}) \leftrightarrow \mathcal{A}$], then if the protocol succeeds add the edge (u_{b^*}, v) to the graph.
- $\mathsf{GuessAnon}(b')$: If the status of the anonymity attack is not $undefined$, abort. If $b' = b$, the status of the anonymity attack is set to $success$; otherwise, it is set to $fail$.

This experiment omits forgery checks, as well as the special handling case for when the trapdoor a appears in an attribute set, for brevity. These changes can only be distinguished by $\mathcal{A}$ if $\mathcal{A}$ achieves a forgery that $\mathcal{H}_{\Gamma(k)}$ would recognise, or if a appears in an attribute set, in which case $\mathcal{A}$ would learn a if operating in $\mathcal{H}_{\Gamma(k)}$, and could use that knowledge to achieve an attribute forgery. Since forgeries cannot be achieved with non-negligible probability in the unmodified security game, and $\mathcal{H}_{\Gamma(k)}$ is indistinguishable from the unmodified game, neither of these outcomes can occur with non-negligible probability.

With the exception of those cases, the adversary's view in $\mathcal{H}_A$ is identical to that in $\mathcal{H}_{\Gamma(k)}$, since for each honest node u, ρ_u in $\mathcal{H}_A$ is distributed identically to $\psi f_{A_u}(a)$ in $\mathcal{H}_{\Gamma(k)}$, and the honest commitments and witness values are therefore indistinguishable between the two games.

For $0 \leq i \leq \Gamma(k)$, let $\hat{\mathcal{H}}_i$ be the hybrid experiment defined by the following modifications to $\mathcal{H}_A$. When initialising the game, for $i \in [\lambda]$, $\mathcal{C}$ samples and stores $z_i \leftarrow^R \mathbb{Z}_p$. For an adversary $\mathcal{A}$'s j^{th} valid query to $\mathsf{AddHonestParty}$, $j \leq i$, parametrised with level $l \leq \lambda$, $\mathcal{C}$ chooses $\rho_2 = \rho_1 z_l$. For the j^{th} query to $\mathsf{AddHonestParty}$, $j > i$, $\mathcal{C}$ samples ρ_2 randomly as in $\mathcal{H}_A$. By definition, $\hat{\mathcal{H}}_0$ is identical to $\mathcal{H}_A$.

It can be seen by inspection that $\mathcal{A}$'s view in $\hat{\mathcal{H}}_{\Gamma(k)}$ is independent of the anonymity bit, since if u_0 and u_1 are chosen such that $\mathsf{SetAnonChallenge}$ succeeds, then $L(u_0) = L(u_1)$, $\mathsf{sk}(u_1) \in [\mathsf{sk}(u_0)]_{\mathcal{R}_s k}$, $\mathsf{pk}(u_1) \in [\mathsf{pk}(u_0)]_{\mathcal{R}_p k}$, and $\mathsf{cred}_{u_1} \in [\mathsf{cred}_{u_0}]_{\mathcal{R}_M}$, so the output of $\mathsf{RandCred}$ (and by extension every value shown to $\mathcal{A}$ during the Issue and $\mathsf{CredProve}$ protocols) is distributed identically regardless of whether it is run on u_0's or u_1's credential.

Now suppose that for the functions f, f_{cred}, and f_{demo}, there exists a polynomial Γ and a PPT adversary $\mathcal{A}$ making at most $\Gamma(k)$ queries to the $\mathsf{AddHonestParty}$ oracle in the single-authority hybrid security game such that there exists a non-negligible $\epsilon(k)$ such that for some $i(k)$, $\mathcal{A}$ can distinguish $\hat{\mathcal{H}}_{i(k)}$ from $\hat{\mathcal{H}}_{i(k)+1}$ with advantage $\epsilon(k)$.

Let $\mathcal{B}_1$ be a PPT reduction running $\mathcal{A}$ as a subroutine and acting as its challenger, attempting to win the following game for some level $l \leq \lambda$ and with $\mathbb{G}_1$ and $\mathbb{G}_2$ swapped if l is even:

$$\begin{aligned}&\Pr[PP \leftarrow \mathsf{PPGen}_l(1^k);\ (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}_l(PP);\ r_0, r_1, \rho \leftarrow \mathbb{Z}_p^*;\\ &\quad (\mathsf{pk}', \mathsf{sk}') \leftarrow (\mathsf{ConvertPK}(\mathsf{pk}, \rho), \mathsf{ConvertSK}(\mathsf{sk}, \rho));\ b \leftarrow \{0, 1\};\\ &b' \leftarrow \mathcal{A}^{\mathsf{Sign}(\mathsf{sk}, \cdot),\ \mathsf{Sign}(\mathsf{sk}', \cdot)}(\mathsf{pk}, P^{r_0}, \mathsf{pk}', P^{r_b \rho}) : b = b'] \leq \tfrac{1}{2} + \upsilon(k)\end{aligned}$$

$\mathcal{B}_1$ can win this game as follows. When initialising the game for $\mathcal{A}$, $\mathcal{B}_1$ sets $\mathsf{pk}_l = \mathsf{pk}$ and $\mathsf{sk}_l = (1)^{\lambda+1-l}$. Whenever $\mathcal{B}_1$ needs to produce a signature on a message M using $\mathsf{pk}^* \in [\mathsf{pk}_l]_{\mathcal{R}_{pk}}$, it forwards M to the signing oracle for pk to obtain σ,

extracts the blinding factor ρ from the stored sk^* corresponding to pk^*, and produces $\mathsf{ConvertSig}_l(\mathsf{pk}, M, \sigma, \rho)$. $\mathcal{B}_1$ also stores P^{r_0} in place of z_l; using knowledge of P^{r_0} and the trapdoor a, along with a zero-knowledge simulator, it is possible for $\mathcal{B}_1$ to construct all outputs in $\mathcal{A}$'s view as if $z_l = r_0$, even without knowing r_0.

When $\mathcal{A}$ calls $\mathsf{AddHonestParty}(u, l')$ for the j^{th} time, $j \leq i$, $\mathcal{B}_1$ randomly samples ρ_1 and sets $\rho_2 = \rho_1 z_{l'}$. If $l' = l$, $P^{\rho_1 z_l}$ is stored in place of ρ_2, since $\mathcal{B}_1$ does not directly know z_l. If u is later issued a credential, $P^{\rho_1 z_l}$ will become its attribute commitment. When $\mathcal{A}$ calls $\mathsf{AddHonestParty}(u, l')$ for the j^{th} time, $j > i+1$, $\mathcal{B}_1$ samples ρ_1 and ρ_2 randomly, as in $\mathcal{H}_A$.

When $\mathcal{A}$ calls $\mathsf{AddHonestParty}(u, l')$ for the j^{th} time, $j = i+1$, $\mathcal{B}_1$ checks whether $l' = l$. If not, $\mathcal{B}_1$ aborts the game. If so, $\mathcal{B}_1$ sets $\mathsf{pk}(u) = \mathsf{pk}'$ and $\mathsf{sk}(u) = (1)^{\lambda+1-l}$, and stores $P^{r_b \rho}$ in place of ρ_2. Whenever $\mathcal{B}_1$ needs to produce a signature using $\mathsf{pk}(u)$, it uses the signing oracle for pk', and when $\mathcal{B}_1$ needs to use ρ_u, it can produce all the necessary outputs in $\mathcal{A}$'s view using $P^{r_b \rho}$, the trapdoor a, and a zero-knowledge simulator, in the same way as for z_l.

When $\mathcal{B}_1$ handles the $(i+1)^{th}$ node u created in this way, the result is that if $b = 0$ in $\mathcal{B}_1$'s game, there is some ρ such that $\mathsf{pk}(u) = \mathsf{ConvertPK}(\mathsf{pk}_l, \rho)$ and u's commitment $P^{\rho_u} = P^{z_l \rho}$, and so $\mathcal{B}_1$ emulates $\hat{\mathcal{H}}_i$, whereas if $b = 1$, $P^{\rho_u} = P^{r_1 \rho}$ which is distributed identically to a randomly sampled P^{ρ_2}, and so $\mathcal{B}_1$ emulates $\hat{\mathcal{H}}_{i+1}$.

The reduction succeeds only if $l' = l$ for the $(i+1)^{th}$ call to $\mathsf{AddHonestParty}(u, l')$; since a version of this reduction exists for any $l \leq \lambda$, let $\mathcal{B}_1$ be the one with the highest chance of success, which succeeds with probability at least $\frac{1}{\lambda}$. Since $\mathcal{A}$ can distinguish between $\hat{\mathcal{H}}_{i(k)}$ and $\hat{\mathcal{H}}_{i(k)+1}$ with advantage $\epsilon(k)$, $\mathcal{B}_1$ can win its game with advantage $\frac{\epsilon(k)}{\lambda}$, which is non-negligible if $\epsilon(k)$ is non-negligible.

Next, consider the following game, defined for the same l as $\mathcal{B}_1$'s game:

$$\begin{gathered}\Pr[PP \leftarrow \mathsf{PPGen}_l(1^k);\ (\mathsf{pk}, \mathsf{sk}), (\mathsf{pk}'_1, \mathsf{sk}'_1) \leftarrow \mathsf{KeyGen}_l(PP);\ r_0, r_1, \rho \leftarrow \mathbb{Z}_p^*; \\ (\mathsf{pk}'_0, \mathsf{sk}'_0) \leftarrow (\mathsf{ConvertPK}(\mathsf{pk}, \rho), \mathsf{ConvertSK}(\mathsf{sk}, \rho)); b \leftarrow \{0,1\}; \\ b' \leftarrow \mathcal{A}^{\mathsf{Sign}(\mathsf{sk},\ \cdot),\ \mathsf{Sign}(\mathsf{sk}'_b,\ \cdot)}(\mathsf{pk}, P^{r_0}, \mathsf{pk}'_b, P^{r_b \rho}) : b = b'] \leq \tfrac{1}{2} + \upsilon(k)\end{gathered}$$

This game is the same as $\mathcal{B}_1$'s game with the exception that if $b = 1$, the adversary is given a fresh public key instead of a randomised copy of pk. Assume there exists a PPT adversary $\mathcal{A}^*$ capable of distinguishing these two games; then a reduction $\mathcal{B}_2$ can be constructed to break public key class-hiding for mercurial signatures by running $\mathcal{A}^*$ as a subroutine as follows.

Given $(\mathsf{pk}_1, \mathsf{pk}_2^\beta)$ by the class-hiding challenger, $\mathcal{B}_2$ extracts the first element, $X_1 = \mathsf{pk}_1[1], X_2 = \mathsf{pk}_2^\beta[1]$, from each key. $\mathcal{B}_2$ then sets $b = 1$, samples r_0 and r_1 randomly, and passes $(\mathsf{pk}_1, X_1^{r_0}, \mathsf{pk}_2^\beta, X_2^{r_1})$ to $\mathcal{A}^*$. The two games are identical when $b = 0$, so setting $b = 1$ can only increase the probability of $\mathcal{A}^*$ successfully distinguishing them, and $X_1^{r_0}$ and $X_2^{r_1}$ are distributed identically to P^{r_0} and $P^{r_1 \rho}$. If $\beta = 0$, $\mathsf{pk}_2^\beta = \mathsf{ConvertPK}(\mathsf{pk}_1, \rho)$, so $\mathcal{B}_2$ simulates $\mathcal{B}_1$'s game. If $\beta = 1$, pk_2^β is independently generated, so $\mathcal{B}_2$ simulates the new game. Therefore, if $\mathcal{A}^*$ can distinguish the two games with non-negligible probability, $\mathcal{B}_2$ can break public key class-hiding.

Because the two games are indistinguishable, if $\mathcal{B}_1$ can win its original game with non-negligible advantage, it can also win the modified game with non-negligible advantage. However, $\mathcal{B}_1$ can therefore also be used by $\mathcal{B}_2$, as the challenge of distinguishing the two games when $b = 1$ is the same as the challenge of solving the second game when b is random.

Therefore, if public key class-hiding holds for Crites-Lysyanskaya mercurial signatures, $\mathcal{B}_1$ cannot win either game with non-negligible advantage. $\mathcal{A}$ therefore cannot

have a non-negligible advantage to distinguish $\hat{\mathcal{H}}_i$ from $\hat{\mathcal{H}}_{i+1}$, and so $\hat{\mathcal{H}}_{\Gamma(k)}$ is indistinguishable from $\hat{\mathcal{H}}_0$, and by extension from $\mathcal{H}_A$, $\mathcal{H}_{\Gamma(k)}$, and finally $\mathcal{H}_0$, which is the unmodified single-authority security game in section 5.1. Finally, because the adversary's view in $\hat{\mathcal{H}}_{\Gamma(k)}$ is independent of the anonymity bit, and $\hat{\mathcal{H}}_{\Gamma(k)}$ is indistinguishable from the unmodified game, the adversary's view must be independent of the anonymity bit in the unmodified game. □