UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft

Fakultät für Elektrotechnik, Informatik und Mathematik
Arbeitsgruppe Codes und Kryptographie

A Formal Comparison of Advanced Digital Signature Primitives

Master's Thesis

in Partial Fulfillment of the Requirements for the Degree of

Master of Science

by
Patrick Schürmann

submitted to:
Prof. Dr. Johannes Blömer
and
Prof. Dr.-Ing. Juraj Somorovsky

Paderborn, April 14, 2023

Contents

1 Introduction **1**

2 State of research **3**
2.1 Our contribution 9

3 Preliminaries **11**
3.1 Groups, homomorphisms and pairings 13
3.2 Digital signatures 16

4 Advanced signature primitives **21**
4.1 Key-homomorphic signatures 21
4.1.1 Key-homomorphic signatures 21
4.1.2 Publicly key-homomorphic signatures 32
4.2 Structure-preserving signatures on equivalence classes 35
4.3 Signatures with flexible public keys 44
4.4 Mercurial signatures 58
4.4.1 Mercurial signature by Crites and Lysyanskaya 66

5 Relations between the signature primitives **69**
5.1 Connection of mercurial signatures to SFPK and SPS-EQ 69
5.1.1 Flexible public key signatures and structure-preserving signatures on equivalence classes from mercurial signatures 70
5.1.2 Mercurial signature from flexible public key signatures and structure-preserving signatures on equivalence classes 75
5.2 Relation between key-homomorphic signatures and SFPK 80
5.2.1 SFPK from key-homomorphic signatures 81
5.2.2 Key-homomorphic signatures from SFPK 85
5.3 Analysis of key-homomorphic properties of the Backes warm-up SFPK . . 87

6 Applications of the advanced signature primitives **93**

7 Conclusion and future work **101**

Bibliography **107**

A Computational Hardness Assumptions **111**

B Group signatures **113**

1 Introduction

Digital signatures are one of the most important primitives in modern cryptography, since they provide integrity and authenticity of data. Thus, they complement encryption which ensures confidentiality. In basic digital signature schemes, a signer can produce a signature σ on a message m using a secret signing key sk only known to herself. This signature can then be publicly validated using a public key pk that belongs to sk. Thus, a secure signature scheme protects the authenticity and integrity of m since it is computationally infeasible to come up with a valid signature for m without knowing sk. One can imagine an abundance of use cases for these signatures, for example e-mail content authentication, ensuring integrity and authenticity of files downloaded from the web and authorizing transactions in cryptocurrency systems.

On the other hand, there are scenarios where we want to achieve a certain level of anonymity for the signer. In basic digital signature schemes, only the signer knows the secret key sk belonging to the public key pk. A person publicly verifying a signature σ under pk thus knows exactly who created that signature. For example, in the scenario of leaking confidential documents to arise public awareness of a crime committed by some company, this might put the signer in danger. Still, we want a mechanism to ensure the integrity of these documents as well as a possibility to check that they originate from some authorized person, but we do not need to know who that person is exactly.

To accomodate these needs, a lot of advanced digital signature primitives have been introduced in the last decades of cryptographic research, four of which we will analyze and compare in this thesis. These primitives provide more flexible ways of signing, revolving around the possibility to (publicly) randomize key-message-signature-triples (pk, m, σ) in various ways.

In this thesis, we will formally compare signatures with flexible public key (SFPK) [BHKS18], structure-preserving signatures on equivalence classes (SPS-EQ) [HS14, FHS14], key-homomorphic signatures (introduced in [DS19], full version of this paper is [DS16]) and mercurial signatures [CL19]. We will compare their formal syntax and security definitions and also look at advanced primitives that can be constructed from them.

2 State of research

In this section, we will explain the general ideas behind the advanced signature primitives covered in this thesis, which are SFPK [BHKS18], SPS-EQ [HS14, FHS14], mercurial signatures [CL19] and key-homomorphic signatures [DS16]. We list the state of the art about assumptions under which the signature types can be instantiated as well as their applications to further advanced cryptographic primitives.

SFPK and SPS-EQ One example of an advanced digital signature primitive are *signatures with flexible public key* (alternatively called *flexible public key signatures* or *SFPK* as a shorthand). These signatures were introduced and securely instantiated from standard assumptions by Backes et al. in [BHKS18]. The public key space of such a signature scheme is partitioned into equivalence classes of some public key equivalence relation R_{pk}. A user of an SFPK key pair $(\mathsf{pk}, \mathsf{sk})$ can efficiently randomize it to obtain a new key pair $(\mathsf{pk}', \mathsf{sk}')$ where the two public keys pk and pk' are related via R_{pk} but unlinkable. Unlinkable hereby means that $(\mathsf{pk}, \mathsf{pk}')$ is indistinguishable from a tuple of two indepently generated keys. Because of this, however, messages signed with the randomized secret key sk' cannot be distinguished from messages signed with an independently generated fresh secret key. This means that without an additional proof of the legitimacy of the signer, authenticity cannot be achieved in applications like group signatures. Such a proof of legitimacy can be achieved by signing the original public key pk with a *structure-preserving signature scheme on equivalence classes* (SPS-EQ). This signature type was first introduced by Hanser et al. in [HS14], but however, their first instantiation was proven insecure soon after [Fuc14]. The message space of an SPS-EQ is partitioned into the equivalence classes of a message equivalence relation R_m. SPS-EQ allow to randomize a valid message-signature pair (m, σ) to a new valid pair (m', σ') where m' is a different representative of the equivalence class of m. The randomization is done in a way that σ' looks like a fresh signature on m' and, additionally, m and m' cannot be linked together, i.e. there is no efficient way to decide whether m and m' are related via R_m or not. To tie this back to the authenticity issue above, the original SFPK public key pk can be signed with an SPS-EQ to obtain a certificate σ_{cert} for pk which then can be randomized together with pk to obtain a randomized certificate σ'_{cert} for the randomized public key pk'. This means that pk' is authenticated as a legitimate signer's public key by σ'_{cert} while being unlinkable to the public key tied to the signer's identity. The above combination of SFPK and SPS-EQ is the basic idea behind the generic *group signature scheme* from [BHKS18] which gives a signer anonymity among the other group members where each group member is equipped with a key-pair-certificate triple $(\mathsf{pk}_i, \mathsf{sk}_i, \sigma^i_{\mathsf{cert}})$ as above.

As applications for standalone SFPK, Backes et al. [BHKS18] discussed how to use standalone SFPK to construct *stealth address protocols* [Tod] which allow to hide the receiver of transferred funds in a cryptocurrency transaction. In the same paper [BHKS18], the authors constructed ring signatures [RST06] from SFPK. Ring signatures allow a member of an ad-hoc set R of possible signers (called a *ring*, not to be confused with mathematical rings) to sign a message on behalf of R. Ad-hoc hereby means that, in contrast to group signatures, no dedicated prior setup of R is required for signing. When in possession of the public keys of all ring members, one can publicly verify a ring signature by ring R. However, it is not possible to efficiently determine the actual signer from R who created that very signature. This means that the signer stays anonymous among the ring members which gives ring signatures several interesting applications, for example concerning the privacy of transactions in cryptocurrency systems.

From SPS-EQ, *anonymous crendential systems (ACS)* can be constructed, as seen in [HS14, FHS19]. These can be used to restrict the access to web resources and services to authorized users only, while letting users authenticate in a way that only information that is required to check the authorization is revealed to the web service. Note that the ACS construction from [HS14] was based on their SPS-EQ construction from the same paper, which was later proven insecure in [Fuc14]. Fuchsbauer et al. [FHS19] give an ACS construction from a secure SPS-EQ. A different example for constructing ACS from SPS-EQ can be found in [CLPK22] where the authors elaborate on the ACS definition from [FHS19] by allowing users to prove that a given set of attributes is explicitly not present in their credential.

Furthermore, Bobolz et al. [BEK+20] constructed *privacy-preserving incentive systems* using SPS-EQ amongst other cryptographic building blocks. These are customer loyalty systems that allow a store customer to collect points in an anonymous way, i.e. without the store learning the point count or being able to link purchases to particular customers.

As mentioned above, the first instantiation of SPS-EQ [HS14] was proven insecure by Fuchsbauer [Fuc14]. The first secure instantiation of EUF-CMA SPS-EQ (in the generic group model for type-3 bilinear groups) was given in [FHS14] by Fuchsbauer et al. Khalili et al. subsequently constructed the first EUF-CMA SPS-EQ from standard assumptions in [KSD19]. As an extension to the SPS-EQ primitive, Mir et al. [MSBM22] introduce *structure-preserving signatures on equivalence classes on updatable commitments (SPS-EQ-UC)*. These signatures do not only allow randomizing message-signature pairs as basic SPS-EQ but also allow to extend the message vector by adding further elements while updating the original signature in a way that it stays valid.

As another extension for SPS-EQ, Hanzlik and Slamanig [HS21] introduced *aggregatable attribute-based equivalence class signatures (AAEQ)*. These signatures distinguish between two types of key pairs: the main key pair $(\mathsf{pk}, \mathsf{sk})$ and attribute signing key pairs $(\mathsf{pk}_{\mathsf{attr}}, \mathsf{sk}_{\mathsf{attr}})$. An attribute signing key pair $(\mathsf{pk}_{\mathsf{attr}}, \mathsf{sk}_{\mathsf{attr}})$ can be used to sign a message m w.r.t a certain attribute attr with a value v_{attr} (as an example, think of attr as the age of a user). The main key pair $(\mathsf{pk}, \mathsf{sk})$ is used to issue and authenticate these attribute signing key pair. Furthermore, multiple signatures $\sigma^{(1)}_{\mathsf{attr}}, \ldots, \sigma^{(n)}_{\mathsf{attr}}$ w.r.t. attributes

$\mathsf{attr}_1, \ldots, \mathsf{attr}_n$ can be aggregated into a compact signature $\hat{\sigma}$ for the set $\{\mathsf{attr}_1, \ldots, \mathsf{attr}_n\}$ of the attributes. Compact hereby means that the size of $\hat{\sigma}$ is significantly smaller than the size of the $\sigma^{(i)}_{\mathsf{attr}}$ combined. Hanzlik and Slamanig [HS21] combine AAEQ with SFPK to construct an extended type of ACS which they call *core/helper anonymous credentials (CHAC)*. In CHAC, there are two parties involved in the showing of a credential: a core device with very constrained computational power (e.g. consider the SIM card of a phone) and a helper device (e.g. consider the phone that the SIM card is inserted into). The computations at the core device should have constant complexity in the size of the credential in order to accomodate its constraints in computational power. On the other hand, the helper device should not be able to show a credential without the help of the core device. In their CHAC construction, they use signatures w.r.t. attributes (created using the AAEQ signature scheme) to encode attribute values into a credential.

Mercurial signatures *Mercurial signatures*, which were first introduced by Crites and Lysyanskaya [CL19], are another example of an advanced signature scheme. Analogously to SFPK and SPS-EQ, the key and message spaces of a mercurial signature scheme are partitioned into equivalence classes of individual equivalence relations. One can transform a triple (pk, m, σ) such that signature σ is valid for message m under public key pk, to a new triple $(\mathsf{pk}', m', \sigma')$ such that σ' is a valid signature for m' under pk' where pk' and m' are equivalent but unlinkable to pk and m, respectively. More precisely, a mercurial signature needs to fulfill *class-hiding* for both the message and the public key space. This means that for any two messages m and m' (analogously public keys pk and pk') it is hard to decide whether they belong to the same class of the respective equivalence relation or not. So mercurial signatures allow to jointly randomize both messages and signatures together with the corresponding public key, which gives rise to the question whether mercurial signatures are equivalent to a combination of SFPK and SPS-EQ.

In [CL19], Crites and Lysyanskaya proposed a mercurial signature that is secure in the generic group model for type-3 bilinear groups. This mercurial signature had keys of the same length as the messages that were signed with them. In [CL20], Crites and Lysyanskaya overcame this shortcoming by presenting a mercurial signature where the keys consist of a constant number of group/ring elements while the messages are still group element vectors of arbitrary length.

Mercurial signatures can be used to construct advanced anonymous credential systems, namely *delegatable anonymous credentials (DACs)* [BCC+09] as shown in [CL19]. As regular (i.e. not necessarily delegatable) ACS, they allow users to anonymously present credentials for a web resource i.e. to access the resource without revealing information about themselves beyond that they are authorized to access the resource. However, in addition, a user can also delegate her credentials in an anonymous way to other users of the system. This results in *certification chains*, which are used as credentials. A user with a certification chain of length l can issue level-$(l+1)$ credentials (i.e. certification chains of length $l+1$) to other users. As an example application scenario for delegatable anonymous credentials, consider the following scientific publication access system

(analogously to [CL19]). Imagine some publishing house official Adelle as a root authority (holding a level-1 credential) who issues a level-2 credential to university official Bernard. Bernard can then issue a level-3 credential to a student Castor from his university, allowing Castor to access the publications from Adelle's publishing house without having to receive the respective credential from Adelle herself. When Adelle issues the credential to Bernard, it is neither revealed who she is nor to whom else she has issued credentials so far. Furthermore, when Castor receives his credential from Bernard, it is neither revealed who Bernard is, who issued him his credential nor who else received credentials from him. When Castor uses his credential, neither the identity of Adelle, Bernard nor Castor himself is revealed. This means that Castor can remain anonymous upon authenticating with his credential, even in the case that Bernard will never issue a credential to anyone else. If, for example, a verifier could see that Castor has received his credential from Bernard, she could infer that Castor is a student at Bernard's university. If she furthermore somehow knew that Castor is the only student with a credential from Bernard, she could perfectly identify Castor.

The first construction of DACs from mercurial signatures was given by Crites and Lysyanskaya in [CL19]. In their construction, credentials are chains $(\mathsf{pk}'_1, \sigma'_1), \ldots, (\mathsf{pk}'_l, \sigma'_l)$ of certified public keys, ending with a randomized version pk'_l of the owners public key pk_l. To delegate a credential to another user Bernard, a user Adelle randomizes all public keys pk'_i in her chain (including their certificates σ'_i) and then extends the chain by signing Bernards public key $\mathsf{pk}_B = \mathsf{pk}_{l+1}$ with the secret key sk''_l belonging to the new randomized version pk''_l of her public key that currently is at the end of the chain. Credential verification is done by verifying all signatures σ'_i in the chain under the previous public key pk_{i-1}. Furthermore, the owner of a credential needs to prove that she knows the corresponding secret key sk'_l for the public key pk'_l at the end of the certification chain. Anonymity is achieved by the class-hiding of the underlying mercurial signature, i.e. the public keys $\mathsf{pk}'_1, \ldots, \mathsf{pk}'_l$ that are exposed when showing a credential chain $(\mathsf{pk}'_1, \sigma'_1), \ldots, (\mathsf{pk}'_l, \sigma'_l)$ cannot be linked to the original public keys $\mathsf{pk}_1, \ldots, \mathsf{pk}_l$ of the l users involved in the creation of this credential chain. This DAC construction by Crites and Lysyanskaya obviously requires the ability to sign the public keys of the underlying mercurial signature scheme in a randomizable way. As sketched in [CL19], the easiest way to achieve this would be to construct a mercurial signature scheme whose message space contains the public key space as a subset. Crites and Lysyanskaya [CL19] are not aware of a mercurial signature construction fulfilling this property. However, their mercurial signature construction from [CL19] stays correct and secure if the message and public key spaces (which are the groups G_1 and G_2 of a bilinear group without their respective neutral elements) are swapped throughout. The black-box DAC construction from [CL19] uses two mercurial signature schemes $\Sigma^{(1)}_{\mathsf{Merc}}, \Sigma^{(2)}_{\mathsf{Merc}}$ with message spaces $\mathcal{M}_i$, public key spaces E_i and secret key spaces H_i which fulfill $E_1 = M_2$, $E_2 = M_1$ and $H_1 = H_2$. Every user holds a key pair for each of the two mercurials and uses the Σ_1 key pair for credentials she receives on an even level of the credential chain and the Σ_2 key pair for credentials she receives on odd levels. This behaviour can be achieved with the mercurial signature from [CL19] as Σ_1 and the variant with switched message and

public key spaces as Σ_2.

Conolly et al. [CLPK22] constructed an SPS-EQ and extended it to a mercurial signature scheme. The authors claim in [CLPK22] that, with minor changes in the delegation process, their mercurial signature can be used to implement DACs from [CL19]. They argue that these changes are necessary since the Connolly SPS-EQ uses tags to distinguish fresh from randomized signatures.

In [MSBM22], Mir et al. mention the following drawbacks of the DACs constructions from [CL19] and [CL20]:

- The constructions do not support attribute-based credentials in a practical way. Attribute-based credentials means that credentials come with additional attributes (such as the age of the owner or the expiration date of the credential) that the owner of the credential can reveal at will during a showing of the credential. The construction from [CL19] does not support attributes at all while the construction from [CL20] does support attributes, but does not allow the user to selectively disclose them. This means that upon showing a credential, the owner cannot decide to reveal only some attributes of the credential while the others remain hidden.
- As pointed out in [BF20] by Bauer and Fuchsbauer, users in the DACs construction from [CL19] are not anonymous to the users involved in the creation of their credential chain. More precisely, if a user Adelle delegates a credential to another user Bernard, she will be able to identify Bernard whenever he uses this credential.
- The size of the credentials is linear in the number of delegation processes. More precisely, if a credential is delegated l times, it consists of a chain of l public keys and l signatures.

In [MSBM22], Mir et al. present a SPS-EQ-UC-based DACs construction that mitigates the above problems.

Key-homomorphic signatures In [DS16], Derler and Slamanig formally introduced the idea of *key-homomorphic signatures*. Key-homomorphic signatures allow to adapt an existing signature σ for a message m under a public key pk to a new signature σ' for the same message m under an adapted public key pk'. In the same paper, the authors prove the practical relevance of their new definition by analyzing several existing signature schemes [Sch91, GQ88, BLS04, KW03, BFG13, PS16, AGOT14, Gha16] and prove that they are key-homomorphic. Derler and Slamanig [DS16] show how to use key-homomorphic signatures to construct *ring signature schemes* (introduced in [RST06]) and *universal designated-verifier signatures* (introduced in [SBWP03]). As already explained when discussing applications of SFPK, ring signatures allow a member of an ad-hoc set R of possible signers to sign a message on behalf of R. Universal designated-verifier signatures (UDVS) [SBWP03] are signature schemes that allow for the creation of a signature σ that cannot be verified by any arbitrary party but only by a designated verifier. The designation process (i.e. the process of turning a regular signature into one

that can only be verified by a specific party) is capsulated from the signature creation procedure, allowing a different party than the original signer to designate a signature to a specific verifier.

A third application for key-homomorphic signatures that Derler and Slamanig explore in [DS16] are *simulation-sound extractable non-interactive zero-knowledge proof systems* (SSE-NIZK proof systems). Such proof systems extend the notion of a proof of knowledge to the scenario where both the setup and the proofs observed by the adversary are done by a simulator instead of a party using the proof system algorithms. An SSE-NIZK proof system basically fulfills an analogous proof-of-knowledge notion in this simulated setting, i.e. it is hard for the adversary to efficiently craft an accepted proof that no witness can be extracted from.

Publicly key-homomorphic signatures On the other hand, *publicly key-homomorphic signatures* (introduced by Derler and Slamanig in [DS16]) allow to combine multiple signatures σ_i for the same message m under public keys pk_i into one signature $\hat{\sigma}$ for m under $\hat{\mathsf{pk}} := \Pi_{i=1}^{n} \mathsf{pk}_i$. In the same paper, the authors furthermore prove the practical relevance of their new definition by examining several existing signature schemes [BLS04, BFG13, PS16, AGOT14, Gha16] and proving that they are publicly key-homomorphic signatures.

In [DS16], Derler and Slamanig furthermore show how publicly key-homomorphic signatures can be used to construct so-called *multisignature schemes* (first introduced by Itakura and Nakamura in [IN83]). In a multisignature, a group of independent signers executes an interactive protocol to jointly sign a message m. The multisignature construction by Derler and Slamanig [DS16] is straightforward since every party signs m individually and transmits the resulting signature σ_i. Once all individual signatures σ_i have been transmitted, every party (deterministically) combines the σ_i to retrieve the jointly computed signature $\hat{\sigma}$.

Multikey-homomorphic signatures [DS16] are another example for a signature primitive that allows to combine multiple signatures σ_i into one. In contrast to the above-mentioned publicly key homomorphic signatures, the σ_i do not need to be signatures on the same message m. Instead, given signatures σ_i for messages m_i that are valid under public keys pk_i, one can combine them to a signature $\hat{\sigma}$ on a message $\hat{m} = f(m_1, \ldots, m_n)$ valid under a public key $\hat{\mathsf{pk}}$ that contains all input public keys pk_i. Hereby, $f \in \mathcal{F}$ is a function from a class $\mathcal{F}$ of admissible functions defined by the scheme, furthermore every multikey-homomorphic signature needs to define individually what key containment means. Note that in contrast to publicly key-homomorphic signatures, multikey-homomorphic are less restrictive in terms of public key combination. Every multikey-homomorphic signature scheme can define an individual way to combine the public keys pk_i into $\hat{\mathsf{pk}}$ while for publicly key-homomorphic signatures we always have $\hat{\mathsf{pk}} := \Pi_{i=1}^{n} \mathsf{pk}_i$. Derler and Slamanig require practical multikey-homomorphic signature schemes to have succinct combined public keys $\hat{\mathsf{pk}}$ and succinct combined signatures $\hat{\sigma}$, where succinctness means that $\hat{\mathsf{pk}}$ and $\hat{\sigma}$ have polynomial length in the security parameter used to generate the original public keys pk_i and signatures σ_i. In [DS16], the key succinctness

problem was solved by presenting a generic construction for a key-succinct multikey-homomorphic signature from key-homomorphic signatures. However, in [DS16], Derler and Slamanig remark that the construction of signature-succinct multikey-homomorphic signatures under mild assumptions seems to be a non-trivial task.

In [DS16], Derler and Slamanig prove that perfectly adaptable and perfectly publicly adaptable key-homomorphic signatures can be instantiated under standard assumptions. As an example, consider the shared hashing parameters variant of Waters signatures [BFG13] which fulfills both perfect adaptability and perfect public adaptability, as proven in [DS16].

2.1 Our contribution

In this thesis, we first define the four advanced signature primitives (which are key-homomorphic signatures [DS16] (Sect. 4.1), SPS-EQ [HS14, FHS14] (Sect. 4.2), SFPK [BHKS18] (Sect. 4.3) and mercurial signatures [CL19] (Sect. 4.4)) in a unified notation in Chapter 4. On the way, we make remarks about their basic properties. For example, we prove a necessary condition for unforgeability for SFPKs in Lem. 4.32.

After that, we analyze the connections between the four signature types from Chapter 4. We start with the connection between mercurial signatures, SFPK and SPS-EQ (Sect. 5.1). Next, we elaborate the fundamental differences between key-homomorphic signatures [DS16] and SFPK [BHKS18] in Sect. 5.2. Lastly, as an example, we examine the warm-up SFPK construction by Backes et al. [BHKS18] for key-homomorphic properties from [DS16] as we defined them in Sect. 4.1 (see Sect. 5.3).

In Chapter 6, we then briefly cover applications of the advanced signature types from Chapter 4. As an extended example, we examine whether group signatures can be constructed from mercurial signatures (instead of a combination of SFPK and SPS-EQ as done by Backes et al. in [BHKS18]). We furthermore highlight the differences between SFPK and key-homomorphic signatures by examining the key-homomorphic-signature-based SSE-NIZK proof system construction from [DS16] and discussing the difficulties when attempting to rebuild it based on SFPK instead of key-homomorphic signatures.

3 Preliminaries

In this chapter, we are going to recap the basics of group theory and digital signatures. Before we do so, we first coin the notation used in this work.

- If the execution of a probablistic algorithm $\mathcal{A}$ on input x yields result y, we denote this by $y \leftarrow\!\!\$\ \mathcal{A}(x)$.
- The *support* of a probablistic algorithm $\mathcal{A}$ on input x is the set of all results that have positive probability of being output by $\mathcal{A}$ on input x. It is denoted by $[\mathcal{A}(x)]$.
- If we do not explicitly state that an algorithm in this work is probablistic then it is deterministic.
- If an algorithm $\mathcal{A}$ has access to to oracle $\mathcal{O}$, we denote this by a superscript, like $\mathcal{A}^{\mathcal{O}}$.
- If we say that algorithm $\mathcal{A}$ is *probablistic polynomial-time (ppt)*, this means that for any input $x \in \{0,1\}^*$, the call $\mathcal{A}(x)$ uses internal randomness and terminates in polynomial time in the bitlength of x.
- poly denotes some arbitrary polynomial function.
- $r \leftarrow\!\!\$\ S$ means that element r is chosen uniformly at random from set S.
- $[n]$ denotes the set $\{1, \ldots, n\}$.
- The tuple v that contains all random variables that an adversary $\mathcal{A}$ is input during a security experiment Exp (in order of appearance) is called the *view* of $\mathcal{A}$ in Exp.

We continue by defining the important concept of negligible functions which will be used to define computational security for cryptographic primitives. Intuitively, a negligible function is a function that vanishes faster than any inverse polynomial.

Definition 3.1 (**negligible function**) *A function $f : \mathbb{N} \to \mathbb{R}$ is called* negligible *if*

$$\forall c > 0 : \exists n_0 \in \mathbb{N} : \forall n \geq n_0 : |f(n)| < \frac{1}{n^c}$$

Next, we define one-way functions which are the most basic cryptographic primitive that almost all others are built upon. Basically a one-way function is a function f that can be evaluated efficiently but given a random element y from its image, it is hard to find a preimage x that maps to y under f.

Definition 3.2 (**one-way function**) *Let $f : \{0,1\}^* \to \{0,1\}^*$ be a function. We define the following security experiment for f between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$:*

$\mathsf{Exp}^{\mathsf{OWF}}_{\mathcal{A},f}(\lambda)$

1: $x \leftarrow\!\!\$\ \{0,1\}^{\mathsf{poly}(\lambda)}$
2: $y := f(x)$
3: $x^* \leftarrow\!\!\$\ \mathcal{A}(\lambda, y)$
4: **return** 1 **if** $f(x) = f(x^*)$

We define the advantage of $\mathcal{A}$ in the above security experiment as

$$\mathsf{Adv}^{\mathsf{OWF}}_{\mathcal{A},f}(\lambda) := \Pr[\mathsf{Exp}^{\mathsf{OWF}}_{\mathcal{A},f}(\lambda) = 1]$$

We say that f is a one-way function (OWF) *if the following two conditions are met:*

(i) $\exists$ deterministic ppt algorithm $\mathcal{F}$: $\forall x \in \{0,1\}^* : \mathcal{F}(x) = f(x)$

(ii) $\forall$ ppt adversaries $\mathcal{A}$: $\mathsf{Adv}^{\mathsf{owf}}_{\mathcal{A},f}(\lambda)$ is negligible in λ

While we usually assume that the security parameter λ of a cryptosystem is publicly known, we explicitly pass it to $\mathcal{A}$ here to stress that $\mathcal{A}$ knows the length of the original preimage x in bits.

Many of the signature schemes that we look at in this work have a key or a message space that is partitioned into equivalence classes of a certain equivalence relation. We recall the formal definition of equivalence relations in the following.

Definition 3.3 (**equivalence relation**) *Let M be a set, $R \subseteq M \times M$ a relation on M.*

(i) R is reflexive *if for all $x \in M$, we have $(x,x) \in R$.*

(ii) R is symmetrical *if for all $(x,y) \in R$, we have $(y,x) \in R$.*

(iii) R is transitive *if it holds that*

$$(x,y),(y,z) \in R \Rightarrow (x,z) \in R$$

R is an equivalence relation *if R is reflexive, symmetrical and transitive.*

For $(x,y) \in R$, we say that x and y are related *via R and sometimes denote this by $x \sim_R y$. If the relation is clear from the context, we sometimes just write $x \sim y$.*

If R is an equivalence relation, any $x, y \in M$ that are related via R are called equivalent*.*

The set

$$[x]_R := \{y \in M \mid x \sim_R y\}$$

of all elements equivalent to x is called the (equivalence) class *of x.*

The set

$$M/R := \{[x]_R \mid x \in M\}$$

of all equivalence classes of R on M is called the quotient set *of R on M.*

For an equivalence relation R over set M, it follows by definition that

$$[x]_R = [y]_R \Leftrightarrow x \sim_R y$$

i.e. the classes of two elements are equal if and only if the elements are equivalent. This comes from the fact that any element $z \in M$ must either be contained in both classes or in none of them. This also concludes

$$\neg(x \sim_R y) \Rightarrow [x]_R \cap [y]_R = \emptyset$$

i.e. the classes of unrelated elements are always disjoint.

3.1 Groups, homomorphisms and pairings

Most digital signature scheme types covered in this thesis require a group structure on their key and/or message space, key-homomorphic signatures [DS19] even require a group homomorphism from their secret to their public key space. We thus quickly recap some basics about groups, homomorphisms and pairings in this subsection, starting with the most basic definition of a mathematical group.

Definition 3.4 (**group**) *Let G be a set. We call a mapping $\cdot : G \times G \to G$ an* operation *on G.*

(i) The operation $\cdot$ is called associative *if for all $a, b, c \in G$, we have $a \cdot (b \cdot c) = (a \cdot b) \cdot c$.*

(ii) $e \in G$ is called a neutral element *in G (w.r.t. operation $\cdot$) if $\forall g \in G : e \cdot g = g = g \cdot e$.*

(iii) Let $g \in G$. $h \in G$ is called the inverse *of g (w.r.t. $\cdot$) if $h \cdot g = e = g \cdot h$. We denote the inverse of element g with g^{-1}.*

(iv) The operation $\cdot$ is commutative *if $\forall g, h \in G : g \cdot h = h \cdot g$.*

$(G, \cdot)$ is a group *if $\cdot$ is associative and there exist a neutral element $1_G \in G$ (w.r.t. $\cdot$) and an inverse g^{-1} for every element $g \in G$ (w.r.t. $\cdot$). A group with a commutative operation on it is called an* abelian group.

For brevity, we sometimes omit the group operation if it is clear from the context and simply write $gh := g \cdot h$. As shown in the next lemma, an operation always has at most one neutral element, so the neutral element of a group is always unique.

Lemma 3.5 *Let G be a set with operation $\cdot$. Let $e, \tilde{e}$ be neutral elements w.r.t. $\cdot$. Then we have $e = \tilde{e}$.*

Proof. $e = e \cdot \tilde{e} = \tilde{e}$ □

Next we define an important class of groups that will play a great role when it comes to constructing concrete digital signatures.

Definition 3.6 (**cyclic group**) *A group $(G, \cdot)$ is called* cyclic *if*

$$\exists g \in G : \langle g \rangle := \{g^x \mid x \in \mathbb{N}\} = G$$

Such a g is called a generator *for G.*

So a cyclic group has a very simple structure since every element in it can be obtained by rising a generator of G to a certain power. It follows from inspection that every cyclic group is abelian. Note that generators do not have to be unique. One can prove that every group with a prime number of elements is a cyclic group, for this we need the concepts of element and group orders which we recall in the next definition.

Definition 3.7 (**element order, group order**) *Let G be a group.*

(i) For an element $g \in G$, the (element) order *of g is defined as*

$$\mathsf{ord}(g) := min\{n \in \mathbb{N} \mid g^n = 1_G\}$$

(ii) The (group) order *of G is its cardinality $|G|$.*

It follows from $\langle g \rangle \subseteq G$ that $\mathsf{ord}(g) \leq |G|$ for any $g \in G$ and g is a generator for a finite group G if and only if $\mathsf{ord}(g) = |G|$. With this, we can now prove that every prime order group is cyclic, i.e. possesses a generator.

Lemma 3.8 *Let G be a group with $|G| = p \in \mathbb{P}$. Then G is cyclic.*

Proof. Let $g \in G$. Lagrange's theorem yields that $\mathsf{ord}(g)$ divides $|G|$. Since p is a prime, this means $\mathsf{ord}(g) \in \{1, p\}$. By definition of the element order (Def. 3.7), we see that every element with order 1 is a neutral element for the operation on G.

Since the neutral element of G is unique according to Lem. 3.5 and G contains $|G| = p > 1$ elements, we get that there exists an element $h \in G$ with $\mathsf{ord}(h) = p = |G|$ which thus is a generator for G. So G is cyclic, as claimed. □

We directly get that every prime order group is abelian as well.

When it comes to mappings between groups, it is a natural question whether there are mappings that respect the operations on the groups. Such mappings, called homomorphisms, are formalized in the next definition.

Definition 3.9 (**group homomorphism**) *Let $(G, \cdot_G), (H, \cdot_H)$ be groups. A map $\mu : G \to H$ is called a* (group) homomorphism *if*

$$\forall g_1, g_2 \in G : \mu(g_1 \cdot_G g_2) = \mu(g_1) \cdot_H \mu(g_2)$$

A bijective group homomorphism is called a group isomorphism.

Group homomorphisms play a role in the context of key-homomorphic signatures [DS19] where the secret and public key spaces are groups (additive and multiplicative, respectively). The public key pk corresponding to a secret key sk is computed as the image $\mu(\mathsf{sk})$ under the group homomorphism μ associated to the key-homomorphic signature scheme. The group homomorphism property of μ basically says that one can

compute the public key corresponding to a sum of two secret keys as the product of the respective corresponding public keys.

We recap some further useful properties of group homomorphisms in the following lemma.

Lemma 3.10 *Let $(G, \cdot_G), (H, \cdot_H)$ be groups, $\mu : G \to H$ a group homomorphism. Then it holds that*

(i) $\mu(1_G) = 1_H$

(ii) $\mu(g^{-1}) = \mu(g)^{-1}$

Proof. Using the homomorphic property of μ, we get

$$1_H \cdot_H \mu(1_G) = \mu(1_G) = \mu(1_G \cdot_G 1_G) = \mu(1_G) \cdot_H \mu(1_G)$$

Multiplying both sides of the equation with $\mu(1_G)^{-1}$ from the right yields (i). Again using the homomorphic property of μ and (i), we see

$$1_H = \mu(1_G) = \mu(g \cdot_G g^{-1}) = \mu(g) \cdot_H \mu(g^{-1})$$

Multiplying with $\mu(g)^{-1}$ from the left yields statement (ii) and finishes the proof. □

In some contexts, the set of group elements that a homomorphism μ maps to the neutral element of its codomain play a special role. This is why we give this set a special name in the following definition.

Definition 3.11 (**kernel**) *Let G, H be groups, $\mu : G \to H$ be a group homomorphism. We call*

$$\mathsf{ker}(\mu) := \{g \in G \mid \mu(g) = 1_H\}$$

the kernel *of μ.*

To conclude this section, we recap the important idea of a *bilinear group* which basically is a triple of prime order groups equipped with a *pairing*. We start with defining pairings.

Definition 3.12 (**pairing**) *Let $(A, +_A), (B, +_B), (C, +_C)$ be groups. A map $e : A \times B \to C$ is called a* bilinear map *or a* pairing *if it holds that*

$$\forall a, a' \in A, b \in B : e(a +_A a', b) = e(a, b) +_C e(a', b)$$

and

$$\forall a \in A, b, b' \in B : e(a, b +_B b') = e(a, b) +_C e(a, b')$$

So a pairing basically is a map defined on the cartesian product of two groups which provides the homomorphism property w.r.t. each of the two components of its domain's elements.

Definition 3.13 (**bilinear group**) *A* bilinear group *is a tuple $(G_1, G_2, G_T, p, e, g_1, g_2)$ fulfilling the following requirements:*

(i) G_1, G_2, G_T *are groups of prime order* p

(ii) $G_1 = \langle g_1 \rangle$, $G_2 = \langle g_2 \rangle$

(iii) $e : G_1 \times G_2 \to G_T$ *is an efficiently computable pairing with* $G_T = \langle e(g_1, g_2) \rangle$

The last requirement for the pairing e is called *non-degeneracy* and, in the setting of prime order groups, basically translates to the statement that some tuple (g_1, g_2) of generators is not mapped to the neutral element of the codomain group G_T. We conclude this section by defining the syntax of a *bilinear group generator*, which is an efficient algorithm that generates bilinear groups of specified types. The following definition (with explicit passing of the bilinear group type as a parameter) is based on [DS16].

Definition 3.14 (**bilinear group generator**) *A* bilinear group generator BGGen *is a ppt algorithm that takes as input the security parameter* $\lambda \in \mathbb{N}$ *and a type parameter* $t \in \{1, 2, 3\}$ *and outputs a (type-t) bilinear group* $\mathrm{BG} = (G_1, G_2, G_T, p, e, g_1, g_2)$.

- *if* $t = 1$, *it holds that* $G_1 = G_2$
- *if* $t = 2$, BGGen *additional outputs an efficiently computable isomorphism* $\psi : G_2 \to G_1$
- *if* $t = 3$, *it holds that no efficiently computable isomorphism* $\psi : G_2 \to G_1$ *exists*

It is plain to see that every type-1 bilinear group is type-2 since we can always choose $\psi := id_{G_1}$ in this case.

3.2 Digital signatures

In this section, we will formally define syntax and security for digital signature schemes and use this definition as a basis for our definitions of the advanced signatures in Chapter 4. We start with the syntax of a digital signature scheme.

Definition 3.15 (**digital signature scheme**) *A* digital signature scheme *with* public key space E, secret key space H, message space $\mathcal{M}$ *and* signature space $\mathcal{S}$ *is a quadruple* Σ *of ppt algorithms defined as follows:*

$\mathsf{PGen}(1^\lambda)$ *probablistic parameter generation algorithm; takes as input the security parameter* $\lambda \in \mathbb{N}$ *and outputs* public parameters pp

$\mathsf{KGen}(\mathsf{pp})$ *probablistic key generation algorithm; takes as input public parameters* pp *and outputs a* key pair $(\mathsf{pk}, \mathsf{sk})$, *consisting of a* public key $\mathsf{pk} \in E$ *and a* secret key $\mathsf{sk} \in H$

$\mathsf{Sig}(\mathsf{sk}, m)$ *probablistic signing algorithm; takes as input a secret key* $\mathsf{sk} \in H$ *and a message* $m \in \mathcal{M}$ *and outputs a signature* $\sigma \in \mathcal{S}$

$\mathsf{Vfy}(\mathsf{pk}, m, \sigma)$ *deterministic verification algorithm; takes as input a public key* pk*, a message* $m \in \mathcal{M}$ *and a signature* $\sigma \in \mathcal{S}$ *and outputs a bit* $b \in \{0, 1\}$

Remark 3.16 *(i) If not explicitly stated otherwise, we always assume the algorithms of a digital signature scheme to be named as in Def. 3.15.*

(ii) Note that we omit the key, message and signature spaces if they are clear from the context.

(iii) Furthermore, these spaces can depend on the public parameters $\mathsf{pp} \leftarrow\!\!\$\ \mathsf{PGen}(\lambda)$ *which (implicitly) contain the security parameter* $\lambda \in \mathbb{N}$*. This means that e.g. instead of a fixed public key space* E *for a scheme we have a function* $E(\mathsf{pp})$ *in the public parameters that returns the respective public key space. If this does not play a role in our current considerations, we also omit this for simplicity and just say that a signature scheme has some public key space* E*. This allows us to restrict e.g. the public key space of a signature scheme to the support of* $\mathsf{KGen}(\mathsf{pp})$*, i.e. a set only consisting public keys that actually can be output by the key generation algorithm with the respective setup and thus definitely have a corresponding secret key.*

The idea behind the parameter generation algorithm is that one often wants to sample multiple independent key pairs using the same parameters pp (for example, some prime order group $G = \langle g \rangle$ together with a generator g). Thus we introduce the parameter generation algorithm PGen to sample these parameters once and subsequently input them to the key generation algorithm KGen. Note that key pairs depend on the security parameter λ since the public parameters pp do. Furthermore, the public parameters are implicitly input to all algorithms of the scheme.

Above definition only lists the algorithms that a basic signature scheme consists of and gives names to important in- and outputs of them. Next, we define the correctness constraint that digital signature schemes must fulfill in order to ensure integrity and authenticity of messages.

Definition 3.17 (**digital signature correctness**) *Let* Σ *be a digital signature scheme with message space* $\mathcal{M}$*, public key space* E*, secret key space* H *and signature space* $\mathcal{S}$*.* Σ *is* correct *if for all* $\lambda \in \mathbb{N}$*,* $\mathsf{pp} \leftarrow\!\!\$\ \mathsf{PGen}(1^\lambda)$*,* $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{KGen}(\mathsf{pp})$*,* $m \in \mathcal{M}$ *we have*

$$\Pr[\mathsf{Vfy}(\mathsf{pk}, m, \mathsf{Sig}(\mathsf{sk}, m)) = 1] = 1$$

Having defined syntax and correctness, we now turn towards security definitions for digital signatures. They all ensure that it is not possible to efficiently forge signatures that are valid under a given public key if one does not know the corresponding secret key. However, the level of security guaranteed depends on the capabilities of the forging party that we assume. We start with the weakest security definition for digital signatures which is *universal unforgeability under no-message attacks (UUF-NMA).*

Definition 3.18 (**UUF-NMA**) *Let* $\lambda \in \mathbb{N}$ *be the security parameter,* Σ *a digital signature scheme. We define the following security experiment between an adversary* $\mathcal{A}$ *and*

a challenger $\mathcal{C}$:

$\mathsf{Exp}^{\mathsf{uuf\text{-}nma}}_{\mathcal{A},\Sigma}(\lambda)$

1: $\mathsf{pp} \leftarrow\!\!\$\ \mathsf{PGen}(1^\lambda)$
2: $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{KGen}(\mathsf{pp})$
3: $m^* \leftarrow\!\!\$\ \mathcal{M}$
4: $\sigma^* \leftarrow\!\!\$\ \mathcal{A}(\mathsf{pp}, \mathsf{pk}, m^*)$
5: **return** $\mathsf{Vfy}(\mathsf{pk}, m^*, \sigma^*)$

The advantage of $\mathcal{A}$ in the above security experiment is defined as

$$\mathsf{Adv}^{\mathsf{uuf\text{-}nma}}_{\mathcal{A},\Sigma}(\lambda) := \Pr[\mathsf{Exp}^{\mathsf{uuf\text{-}nma}}_{\mathcal{A},\Sigma}(\lambda) = 1]$$

Σ is universally unforgeable under no-message attacks*, or in short* UUF-NMA secure*, if for all ppt adversaries $\mathcal{A}$ $\mathsf{Adv}^{\mathsf{uuf\text{-}nma}}_{\mathcal{A},\Sigma}(\lambda)$ is negligible.*

The idea behind UUF-NMA security is that is should be computationally hard to efficiently forge a signature for a random message if one gets no further information about a system that issues signatures apart from the users public keys and the public parameters of the signature scheme instance.

However, UUF-NMA makes strong assumptions about the capabilities of the adversary since it assumes that she is not able to observe any signatures issued by the system she tries to attack. In reality, the adversary could be able to intercept signatures on messages and might even influence the messages that are signed. Furthermore, UUF-NMA security assumes that the adversary intends to forge a signature for a random message m and does not let the adversary choose the message according to the environment it attacks (i.e., the public parameters and the public key). This motivates defining a stronger adversary model, resulting in the following stronger security definition, which is called *existential unforgeability under chosen-message attacks*:

Definition 3.19 (**EUF-CMA**) *Let $\lambda \in \mathbb{N}$ be the security parameter, Σ a digital signature scheme. We define the following security experiment between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$:*

$\mathsf{Exp}^{\mathsf{euf\text{-}cma}}_{\mathcal{A},\Sigma}(\lambda)$

1: $\mathsf{pp} \leftarrow\!\!\$\ \mathsf{PGen}(1^\lambda)$
2: $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{KGen}(\mathsf{pp})$
3: $Q^{\mathsf{Sig}} := \emptyset$
4: $(m^*, \sigma^*) \leftarrow\!\!\$\ \mathcal{A}^{\mathcal{O}(\mathsf{sk},\cdot)}(\mathsf{pp}, \mathsf{pk})$
5: **return** $\mathsf{Vfy}(\mathsf{pk}, m^*, \sigma^*) = 1 \wedge m \notin Q^{\mathsf{Sig}}$

where $\mathcal{O}$ is a signing oracle *as follows:*

$\mathcal{O}(\mathsf{sk}, m)$

1 : $\sigma \leftarrow\!\!\$\ \mathsf{Sig}(\mathsf{sk}, m)$

2 : $Q^{\mathsf{Sig}} \leftarrow Q^{\mathsf{Sig}} \cup \{m\}$

3 : **return** σ

The advantage of $\mathcal{A}$ in the above security experiment is defined as

$$\mathsf{Adv}^{\mathsf{euf\text{-}cma}}_{\mathcal{A},\Sigma}(\lambda) := \Pr[\mathsf{Exp}^{\mathsf{euf\text{-}cma}}_{\mathcal{A},\Sigma}(\lambda) = 1]$$

Σ is existentially unforgeable under chosen-message attacks*, or in short* EUF-CMA secure*, if for all ppt adversaries $\mathcal{A}$ $\mathsf{Adv}^{\mathsf{euf\text{-}cma}}_{\mathcal{A},\Sigma}(\lambda)$ is negligible.*

So EUF-CMA means that it is infeasible to forge a signature on a self-chosen message m when one can observe signatures on arbitrary other messages $m' \neq m$. We can define a stronger security notion for digital signature schemes which is called *strong existential unforgeability*. It basically weakens the winning condition of the above existential unforgeability game by also allowing the adversary to submit a distinct signature for an already-signed message as a forgery.

Definition 3.20 (**sEUF-CMA**) *Let $\lambda \in \mathbb{N}$ be the security parameter, Σ be a digital signature scheme. We define the security experiment $\mathsf{Exp}^{\mathsf{seuf\text{-}cma}}_{\mathcal{A},\Sigma}(\lambda)$ for an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$ by replacing line 5 from $\mathsf{Exp}^{\mathsf{euf\text{-}cma}}_{\mathcal{A},\Sigma}(\lambda)$ from Def. 3.19 with*

$$\textbf{\textit{return}}\ \mathsf{Vfy}(\mathsf{pk}, m^*, \sigma^*) = 1 \wedge (m^*, \sigma^*) \notin Q^{\mathsf{Sign}}$$

and line 2 in the oracle $\mathcal{O}$ from Def. 3.19 with

$$Q^{\mathsf{Sign}} \leftarrow Q^{\mathsf{Sign}} \cup \{(m, \sigma)\}$$

The advantage of $\mathcal{A}$ in the above security experiment is defined as

$$\mathsf{Adv}^{\mathsf{seuf\text{-}cma}}_{\mathcal{A},\Sigma}(\lambda) := \Pr[\mathsf{Exp}^{\mathsf{seuf\text{-}cma}}_{\mathcal{A},\Sigma}(\lambda) = 1]$$

Σ is strongly existentially unforgeable under chosen-message attacks*, or in short* sEUF-CMA secure*, if for all ppt adversaries $\mathcal{A}$ $\mathsf{Adv}^{\mathsf{seuf\text{-}cma}}_{\mathcal{A},\Sigma}(\lambda)$ is negligible.*

4 Advanced signature primitives

In this section, we will formally define the syntax and security of each of the four different advanced signature primitives we will look at in this thesis, which are key-homomorphic signatures [DS19] (Sect. 4.1), SPS-EQ [HS14, FHS14] (Sect. 4.2), SFPK [BHKS18] (Sect. 4.3) and mercurial signatures [CL19] (Sect. 4.4).

4.1 Key-homomorphic signatures

We begin with key-homomorphic signatures. This type of signatures was introduced by Derler and Slamanig in [DS19] but we will base our work about them on the full version of the above paper [DS16]. Roughly speaking, these signatures provide a *secret-to-public key homomorphism* μ from the secret key space to the public key space that maps any secret key to its corresponding public key. Furthermore, they provide an adaption algorithm for signatures. This adaption algorithm allows to turn a valid key-message-signature triple (pk, m, σ) into a distinct valid triple $(\mathsf{pk}', m, \sigma')$ where the public key pk' is the image of $\mathsf{sk} + \Delta$ under μ where Δ is some secret key that can be seen as a *shift amount* for the original secret key sk.

A related primitive also introduced by Derler et al. [DS16] are *publicly-key-homomorphic signatures* which provide the same type of secret-to-public-key homomorphism but, instead of the above adaption algorithm, allow to publicly (i.e. without knowledge of any secret key) combine multiple signatures on the same message m into one.

4.1.1 Key-homomorphic signatures

After having given a rough intuition for the different types of key-homomorphic signatures now, we will introduce them formally. We will start by formally defining secret-to-public key homomorphisms as in [DS16].

Definition 4.1 (**secret-to-public key homomorphism**) *Let Σ be a digital signature scheme with two groups $(H, +)$, $(E, \cdot)$ as secret- and public key spaces, respectively. A group homomorphism (Def. 3.9) $\mu : H \to E$ is called a* secret-to-public-key homomorphism *if for all security parameters $\lambda \in \mathbb{N}$, $\mathsf{pp} \leftarrow\!\!\$\ \mathsf{PGen}(\lambda)$, $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{KGen}(\mathsf{pp})$, we have $\mathsf{pk} = \mu(\mathsf{sk})$.*

So a secret-to-public key homomorphism is a group homomorphism that maps a secret key to its corresponding public key. As an example for a signature scheme with such a homomorphism, consider *Schnorr signatures* [Sch91]. Here, the secret key space is $H = \mathbb{Z}_p$ and the public key space is $E = G$ for a group G of prime order $p \in \mathbb{P}$, the public key corresponding to a secret key $\mathsf{sk} = x$ is $\mathsf{pk} = g^x$. Thus, the mapping

$\mu : \mathbb{Z}_p \to G, x \mapsto g^x$ obviously is a secret-to-public key homomorphism for Schnorr signatures.

With secret-to-public-key homomorphisms defined, we can now continue to define key-homomorphic signature schemes which are digital signatures with a secret-to-public-key homomorphism μ and an additional ppt adaptation algorithm $\mathsf{KH.Adapt}$. $\mathsf{KH.Adapt}$ allows to randomize a given valid key-message-signature triple (pk, m, σ) to a new triple $(\mathsf{pk}', m, \sigma')$ such that pk' is the image of $\mathsf{sk} + \Delta$ under μ for some secret key Δ that serves as a shift amount for the original corresponding secret key sk. The following formal definition of the syntax and correctness of a key-homomorphic signature scheme is based on [DS16]:

Definition 4.2 (**Key-homomorphic signature scheme**) *Let $(H, +), (E, \cdot)$ be groups. A* key-homomorphic signature scheme *is a tuple of ppt algorithms* $\Sigma_{\mathsf{KH}} = (\mathsf{KH.PGen}, \mathsf{KH.KGen}, \mathsf{KH.Sign}, \mathsf{KH.Vfy}, \mathsf{KH.Adapt})$ *fulfilling the following requirements:*

(i) $\Sigma := (\mathsf{KH.PGen}, \mathsf{KH.KGen}, \mathsf{KH.Sign}, \mathsf{KH.Vfy})$ *is a correct digital signature scheme according to Def. 3.17 with public key space E, secret key space H, some message space $\mathcal{M}$ and some signature space $\mathcal{S}$.*

(ii) Σ *provides a secret-to-public-key homomorphism μ*

(iii) $\mathsf{KH.Adapt}$ *is a ppt adaption algorithm, i.e. takes as input a public key $\mathsf{pk} \in E$, a message $m \in \mathcal{M}$, a signature $\sigma \in \mathcal{S}$ and a* shift amount *$\Delta \in H$ and outputs an adapted public key $\mathsf{pk}' \in E$ and an adapted signature $\sigma' \in \mathcal{S}$ which satisfy the following requirement: For all $\Delta \in H$, $\lambda \in \mathbb{N}$, $\mathsf{pp} \leftarrow\!\!\$\ \mathsf{PGen}(\lambda)$, $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{KGen}(\mathsf{pp})$, $m \in \mathcal{M}$, $\sigma \in \mathcal{S}$, we have*

$$\begin{aligned} &\mathsf{KH.Vfy}(\mathsf{pk}, m, \sigma) = 1 \wedge (\mathsf{pk}', \sigma') \leftarrow\!\!\$\ \mathsf{KH.Adapt}(\mathsf{pk}, m, \sigma, \Delta) \\ \Rightarrow &\Pr[\mathsf{KH.Vfy}(\mathsf{pk}', m, \sigma') = 1] = 1 \wedge \mathsf{pk}' = \mu(\Delta) \cdot \mathsf{pk} \end{aligned}$$

We say that such a Σ_{KH} is key-homomorphic with respect to μ.

If not explicitly stated otherwise, we always assume that the algorithms of a key-homomorphic signature scheme are named as in Def. 4.2. Note that the order of $\mu(\Delta)$ and pk in the equation $\mathsf{pk}' = \mu(\Delta) \cdot \mathsf{pk}$ in Def. 4.2 does not actually matter since the public key spaces of all key-homomorphic signature instantiations listed in [DS16] (and Rem. 4.7) are abelian groups. Recall that the Schnorr signature scheme [Sch91] (with secret key space $H = Z_p$ and public key space $E = G$ for a prime order group G) provides the secret-to-public-key homomorphism $\mu : \mathbb{Z}_p \to G, x \mapsto g^x$. In groups where the discrete logarithm problem is hard, this is a one-way function (Def. 3.2). In general, we can prove that for a signature scheme with a secret-to-public-key homomorphism μ and canonical key generation (i.e. a secret key is drawn uniformly at random and the corresponding public key is computed using μ), it holds that unforgeability can only be achieved if μ is a one-way function. This observation is not explicitly mentioned in [DS16]. We first formally define what canonical key generation is (in order to be able to

reference it later on), before continuing to prove the above claim about the one-wayness of μ.

Definition 4.3 (**canonical key generation**) *Let Σ be a signature scheme that provides a secret-to-public-key homomorphism μ, let the groups $(H,+),(E,\cdot)$ be the secret- and public key spaces of Σ, respectively. We say that Σ has* canonical key generation *if the algorithm* KGen *looks as follows:*

KGen(pp)

1 : $\mathsf{sk} \leftarrow\!\!\$\ H$
2 : $\mathsf{pk} := \mu(\mathsf{sk})$
3 : **return** $(\mathsf{pk}, \mathsf{sk})$

Remark 4.4 *For a signature scheme Σ with a secret-to-public-key homomorphism μ that uses canonical key generation as in Def. 4.3, we can assume without loss of generality that μ is injective. To see this, consider the case that μ is not injective, i.e. there exist a public key* pk *and two different secret keys $\mathsf{sk} \neq \mathsf{sk}'$ with*

$$\mu(\mathsf{sk}) = \mathsf{pk} = \mu(\mathsf{sk}')$$

In this case, both $(\mathsf{pk}, \mathsf{sk})$ *and* $(\mathsf{pk}, \mathsf{sk}')$ *are valid key pairs of users u_1 and u_2, respectively (since they have positive probability to be output by* KGen*). However, signatures under* sk *(and thus verifying under* pk*) cannot be linked to user u_1 since u_2 has the same public key* pk *as u_1. So Σ does not guarantee authenticity of the messages signed with it if μ is not injective.*

With canonical key generation being formally defined, we continue to prove that unforgeable signature schemes with a canonical key generation procedure must have a one-way secret-to-public-key homomorphism μ.

Lemma 4.5 *Let Σ be a digital signature scheme with two groups $(H,+)$, $(E,\cdot)$ as secret- and public key spaces, respectively, let $\mu : H \to E$ be a secret-to-public key homomorphism for Σ. Furthermore, assume that Σ has canonical key generation (Def. 4.3). Then it holds that*

$$\Sigma \textit{ unforgeable (Def. 3.19)} \Rightarrow \mu \textit{ OWF (Def. 3.2)}$$

Proof. We prove the claim by contraposition, using a reduction, i.e. we assume the existence of a ppt adversary $\mathcal{A}$ with $\mathsf{Adv}^{\mathsf{OWF}}_{\mathcal{A},\mu}(\lambda)$ not negligible and construct a ppt adversary $\mathcal{B}$ with $\mathsf{Adv}^{\mathsf{euf\text{-}cma}}_{\mathcal{B},\Sigma}(\lambda)$ not negligible.

Let λ be the security parameter, $\mathcal{M}$ be the message space of Σ. When input a public key $\mathsf{pk} \in E$, $\mathcal{B}$ behaves as follows:

1. $\mathcal{B}$ executes $\mathcal{A}(\lambda, \mathsf{pk})$, eventually obtaining a secret key sk'.
2. $\mathcal{B}$ draws $m \leftarrow\!\!\$\ \mathcal{M}$ uniformly at random.
3. $\mathcal{B}$ computes $\sigma \leftarrow\!\!\$\ \mathsf{Sign}(\mathsf{sk}', m)$.

4. $\mathcal{B}$ outputs (m, σ).

$\mathcal{B}$ obviously is a ppt, since $\mathcal{A}$ and Sign are ppts. We see that by definition of the canonical key generation algorithm KGen from Def. 4.3, adversary $\mathcal{A}$ is given an image pk of a uniformly random sk under μ as required in the one-way function security game from Def. 3.2.

Note that if $\mathsf{sk}' = \mathsf{sk}$, where sk is the secret key corresponding to the challenge public key pk from the EUF-CMA game, we have that (m, σ) is a valid message-signature pair under pk because of the correctness of Σ. If and only if $\mathcal{A}$ outputs sk when given input pk, $\mathcal{A}$ has won the one-way function game from Def. 3.2. This is because μ is a secret-to-public-key homomorphism according to Def. 4.1 (so $\mathsf{pk} = \mu(\mathsf{sk})$) and we can assume w.l.o.g. that μ is injective (see Rem. 4.4). Thus we get

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{euf\text{-}cma}}_{\mathcal{B},\Sigma}(\lambda) &= \Pr[\mathsf{Exp}^{\mathsf{euf\text{-}cma}}_{\mathcal{B},\Sigma}(\lambda) = 1] \\
&= \Pr[\mathsf{Vfy}(\mathsf{pk}, m, \sigma) = 1] \\
&\geq \Pr[\mathsf{pk} = \mu(\mathsf{sk}')] \\
&= \mathsf{Adv}^{\mathsf{OWF}}_{\mathcal{A},f}(\lambda)
\end{aligned}$$

which proves the lemma. □

As a second observation, we prove that all signature schemes with canonical key generation have surjective secret-to-public-key homomorphisms.

Lemma 4.6 *Let Σ be a key-homomorphic signature w.r.t. secret-to-public-key homomorphism μ, let the groups $(H, +), (E, \cdot)$ be the secret- and public key spaces of Σ, respectively. Assume that Σ has canonical key generation (Def. 4.3). Then μ is a surjective function.*

Proof. It is clear by definition of canonical key generation (Def. 4.3) that KGen defines

$$E = \mu(H) := \{\mu(\mathsf{sk}) \mid \mathsf{sk} \in H\}$$

So obviously, for all $\mathsf{pk} \in E$, there is an $\mathsf{sk} \in H$ s.t. $\mu(\mathsf{sk}) = \mathsf{pk}$. Thus, μ is surjective by definition. □

Many examples of key-homomorphic signature schemes with canonical key generation as defined in Def. 4.3 exist. The following remark lists the ones that were named by Derler and Slamanig in [DS16].

Remark 4.7 *The following signature schemes are key-homomorphic signatures with canonical key generation Def. 4.3:*

- *Schnorr signatures [Sch91]*
- *Guillou-Quisqater signatures [GQ88]*
- *BLS signatures [BLS04]*

- *Katz-Wang signatures [KW03]*
- *Waters' signatures [Wat05], more precisely the variant with shared hash parameters described in [BFG13]*
- *Pointcheval-Sanders signatures [PS16]*
- *Abe's structure-preserving signature (SPS) [AGOT14]*
- *Ghadafi's SPS [Gha16]*

By Lem. 4.5 and Lem. 4.6, these schemes all have surjective one-way secret-to-public-key homomorphisms.

So Rem. 4.7 proves that Lem. 4.5 and Lem. 4.6 have practical relevance despite severely restricting the key generation algorithm.

In [DS16], Derler and Slamanig ask the question whether it is possible to use $\mu(\Delta)$ instead of Δ as input to the $\mathsf{KH.Adapt}$ algorithm of a key-homomorphic signature scheme. They sketch a proof that a key-homomorphic signature scheme with this syntax would be inherently insecure, namely not even fulfilling UUF-NMA security (Def. 3.18) which is the weakest security notion for digital signatures. We will formally state and prove their claim in the following.

Lemma 4.8 *Let $(H,+)$, $(E,\cdot)$ be groups, $\Sigma'_{\mathsf{KH}} = (\mathsf{KH.PGen}, \mathsf{KH.KGen}, \mathsf{KH.Sign}, \mathsf{KH.Vfy}, \mathsf{KH.Adapt}')$ be a tuple of ppt algorithms such that*

(i) $\Sigma = (\mathsf{KH.PGen}, \mathsf{KH.KGen}, \mathsf{KH.Sign}, \mathsf{KH.Vfy})$ *is a digital signature scheme with public key space E, secret key space H, some message space $\mathcal{M}$ and some signature space $\mathcal{S}$*

(ii) Σ *provides a secret-to-public-key homomorphism μ*

(iii) $\mathsf{KH.Adapt}'$ *is a ppt algorithm that takes as input a public key pk, a message m, a signature σ and a shift amount image $\mu(\Delta) \in E$ for a $\Delta \in H$ and outputs an adapted public key pk' and an adapted signature σ' which satisfy the following requirement: For all $\lambda \in \mathbb{N}$, $\Delta \in H$, $\mathsf{pp} \leftarrow\!\!\$\ \mathsf{PGen}(1^\lambda)$, $(\mathsf{pk},\mathsf{sk}) \leftarrow\!\!\$\ \mathsf{KGen}(\mathsf{pp})$, $m \in \mathcal{M}$, $\sigma \in \mathcal{S}$, we have*

$$\begin{aligned} &\mathsf{KH.Vfy}(\mathsf{pk}, m, \sigma) = 1 \wedge (\mathsf{pk}', \sigma') \leftarrow\!\!\$\ \mathsf{KH.Adapt}'(\mathsf{pk}, m, \sigma, \mu(\Delta)) \\ \Rightarrow\ &\Pr[\mathsf{KH.Vfy}(\mathsf{pk}', m, \sigma') = 1] = 1 \wedge \mathsf{pk}' = \mu(\Delta) \cdot \mathsf{pk} \end{aligned}$$

Then Σ'_{KH} is not UUF-NMA secure.

Proof. Let λ be the security parameter. We prove the statement by constructing an adversary $\mathcal{A}$ with non-negligible advantage $\mathsf{Adv}^{\mathsf{uuf\text{-}nma}}_{\mathcal{A},\Sigma'_{\mathsf{KH}}}(\lambda)$ in the UUF-NMA experiment for Σ'_{KH}. Our construction is from the sketch from [DS16] and even has advantage 1. The intuition behind the attack is to sign the challenge message m with a fresh key pair $(\mathsf{pk}', \mathsf{sk}')$ and adapt the resulting signature to the challenge key pk.

Let $\mathsf{pp} \leftarrow\!\!\$\ \mathsf{PGen}(1^\lambda)$, $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{KGen}(\mathsf{pp})$, $m \leftarrow\!\!\$\ \mathcal{M}$ as in $\mathsf{Exp}^{\mathsf{uuf\text{-}nma}}_{\mathcal{A},\Sigma'}(\lambda)$ from Def. 3.18. Consider an adversary $\mathcal{A}$ that acts as follows on input pk:

1 : $(\mathsf{pk}', \mathsf{sk}') \leftarrow\!\!\$\ \mathsf{KGen}(\mathsf{pp})$
2 : $\sigma' \leftarrow\!\!\$\ \mathsf{Sign}(\mathsf{sk}', m)$
3 : $\hat{\mathsf{pk}} := \mathsf{pk} \cdot (\mathsf{pk}')^{-1}$
4 : $(\sigma'', \mathsf{pk}'') \leftarrow\!\!\$\ \mathsf{KH.Adapt}'(\mathsf{pk}', m, \sigma', \hat{\mathsf{pk}})$

First, observe that $\mathcal{A}$ obviously is a ppt since $\mathsf{KGen}, \mathsf{KH.Sign}$ and $\mathsf{KH.Adapt}'$ are. Because Σ'_{KH} is a correct digital signature scheme, we get that $\Pr[\mathsf{KH.Vfy}(\mathsf{pk}', m, \sigma') = 1] = 1$ and thus $\Pr[\mathsf{KH.Vfy}(\mathsf{pk}'', m, \sigma'') = 1] = 1$. Furthermore, we have

$$\mathsf{pk}'' = \hat{\mathsf{pk}} \cdot \mathsf{pk}' = \mathsf{pk} \cdot (\mathsf{pk}')^{-1} \cdot \mathsf{pk}' = \mathsf{pk}$$

so σ'' is a valid forgery for the challenge message m under the challenge public key pk. Thus, we get

$$\mathsf{Adv}^{\mathsf{uuf\text{-}nma}}_{\mathcal{A},\Sigma'_{\mathsf{KH}}}(\lambda) = \Pr[\mathsf{Exp}^{\mathsf{uuf\text{-}nma}}_{\mathcal{A},\Sigma'_{\mathsf{KH}}}(\lambda) = 1] = 1$$

which proves the lemma. □

Unforgeability Attentive readers might have noticed that we did not define a dedicated unforgeability notion for key-homomorphic signatures. This is because they can use the same as regular digital signature schemes from Sect. 3.2. However, there is a special class of key-homomorphic schemes that inherently cannot fulfill sEUF-CMA which we will analyze in the following. We will make use of the fact that calling $\mathsf{KH.Adapt}$ with a shift amount from the kernel (Def. 3.11) of the secret-to-public-key homomorphism μ which alters the input signature allows to win the strong existential-unforgeability game for a key-homomorphic signature scheme.

Lemma 4.9 *Let Σ_{KH} be a key-homomorphic signature scheme (Def. 4.2) with message space $\mathcal{M}$, secret key space $(H, +)$, public key space $(E, \cdot)$ and secret-to-public key homomorphism $\mu : H \to E$. Assume there exists an efficiently computable shift amount $\Delta^* \in H$ and an efficiently computable message $m \in \mathcal{M}$ with the following properties:*

(i) $\mu(\Delta^*) = 1_E$

(ii) $\Pr[\sigma' \neq \sigma]$ *not negligible for all* $\lambda \in \mathbb{N}$, $\mathsf{pp} \leftarrow\!\!\$\ \mathsf{KH.PGen}(1^\lambda)$, $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{KH.KGen}(\mathsf{pp})$, $\sigma \leftarrow\!\!\$\ \mathsf{KH.Sign}(\mathsf{sk}, m)$, $(\mathsf{pk}', \sigma') \leftarrow\!\!\$\ \mathsf{KH.Adapt}(\mathsf{pk}, m, \sigma, \Delta^*)$ *where the probability is over the internal random choices of* $\mathsf{KH.Adapt}$, $\mathsf{KH.PGen}$, $\mathsf{KH.KGen}$ *and* $\mathsf{KH.Sign}$

Then Σ_{KH} is not strongly existentially unforgeable.

Proof. We prove the lemma by constructing an adversary $\mathcal{A}$ that has non-negligible advantage $\mathsf{Adv}^{\mathsf{seuf\text{-}cma}}_{\mathcal{A},\Sigma_{\mathsf{KH}}}(\lambda)$ in the sEUF-CMA game $\mathsf{Exp}^{\mathsf{seuf\text{-}cma}}_{\mathcal{A},\Sigma_{\mathsf{KH}}}(\lambda)$ from Def. 3.20. Upon input a public key $\mathsf{pk} \in E$, $\mathcal{A}$ behaves as follows:

1. $\mathcal{A}$ computes $\Delta^* \in H$, $m \in \mathcal{M}$ with the properties from the prerequisite.

2. $\mathcal{A}$ queries the signing oracle $\mathcal{O}$ for a signature on m, eventually obtaining $\sigma \leftarrow\!\!\$\ \mathsf{Sign}(\mathsf{sk}, m)$.

3. $\mathcal{A}$ computes $(\mathsf{pk}', \sigma') \leftarrow\!\!\$\ \mathsf{KH.Adapt}(\mathsf{pk}, m, \sigma, \Delta^*)$.

4. $\mathcal{A}$ outputs (m, σ').

By prequisites for m, Δ^* and because Sign and $\mathsf{KH.Adapt}$ are ppt algorithm, $\mathcal{A}$ is a ppt. Since Σ_{KH} is correct (Def. 4.2), we have $\Pr[\mathsf{KH.Vfy}(\mathsf{pk}, m, \sigma) = 1] = 1$ and thus $\Pr[\mathsf{KH.Vfy}(\mathsf{pk}', m, \sigma') = 1] = 1$. Furthermore, because of the properties of Δ^*, we have

$$\mathsf{pk}' = \mu(\Delta^*) \cdot \mathsf{pk} = 1_E \cdot \mathsf{pk} = \mathsf{pk}$$

and thus $\Pr[\mathsf{KH.Vfy}(\mathsf{pk}, m, \sigma') = 1] = 1$, which yields

$$\begin{aligned}\mathsf{Adv}^{\mathsf{seuf\text{-}cma}}_{\mathcal{A},\Sigma_{\mathsf{KH}}}(\mathcal{A}) &= \Pr[\mathsf{Exp}^{\mathsf{seuf\text{-}cma}}_{\mathcal{A},\Sigma_{\mathsf{KH}}}(\lambda) = 1] \\ &= \Pr[\mathsf{KH.Vfy}(\mathsf{pk}, m, \sigma') = 1 \wedge \sigma \neq \sigma'] \\ &= \Pr[\sigma' \neq \sigma] \\ &= \eta(\kappa)\end{aligned}$$

for a non-negligible function η. This proves the lemma. □

We briefly summarize for which of the key-homomorphic signature schemes from [DS16] (see Rem. 4.7) the attack from the above lemma works.

- For Schnorr signatures [Sch91], BLS signatures [BLS04], Katz-Wang signatures [KW03], the Waters signature variant [BFG13] and Pointcheval-Sanders signatures [PS16], there is no shift amount Δ^* with the required properties since for all shift amounts from the kernel of the secret-to-public-key homomorphism μ, the signature $\sigma' \leftarrow\!\!\$\ \mathsf{KH.Adapt}(\cdot, \cdot, \sigma, \Delta^*)$ will always be the same as the input signature σ.

- For Abe's randomizable SPS [AGOT14] and the SPS proposed by Ghadafi [Gha16], Δ^* and m as required in the lemma do exist but both schemes are not sEUF-CMA secure (even if one prohibits the use of the $\mathsf{KH.Adapt}$ algorithm) since they provide a ppt signature randomization algorithm Rand that given a message signature pair (m, σ), outputs a new signature σ' for m which is valid under the same public key as the original signature σ. In fact, for an arbitrary public key pk and a shift amount Δ^* as in the above lemma, $\mathsf{KH.Adapt}(\mathsf{pk}, \cdot, \cdot, \Delta^*)$ is exactly such a signature randomization algorithm Rand with a certain failure probability.

- Consider the Guillou-Quisqater signature scheme [GQ88] with the notation used in [DS16]. It has the following parameter generation algorithm $\mathsf{GQ.PGen}$:

$\mathsf{GQ.PGen}(\lambda)$

1 : choose $p, q \in \mathbb{P}$ of length $\lambda/2$ bits with $p \neq q$
2 : $N := p \cdot q$
3 : choose odd $e < \varphi(N)$ with $e \in \mathbb{P}$
4 : $(H : \mathbb{Z}_N \times \mathcal{M} \to \mathbb{Z}_e) \leftarrow\!\!\$\ \{H_k\}_k$
5 : **return** $\mathsf{pp} := (H, N, e)$

Hereby, $\mathcal{M}$ denotes the message space of the Guillou-Quisqater signature scheme, $\{H_k\}_k$ denotes some hash function family with some parameter k and φ denotes Euler's totient function which is defined as

$$\varphi(N) := |\{x \in \mathbb{N} \mid 0 \leq x < N, \gcd(x, N) = 1\}|$$

for positive integers N where gcd denotes the greatest common divisor. When considering the attack from Lem. 4.9 for Guillou-Quisqater signatures, the candidates for the shift amount Δ^* from Lem. 4.9 are exactly the members of

$$A_{N,e,c} = \{x \in \mathbb{Z}_N^* \mid x^e = 1, x^c \neq 1\}$$

It seems to be a hard problem to compute a lower bound for

$$\Pr[A_{N,e,c} \neq \emptyset]$$

where the probability is over the random choices of the algorithms of the Guillou-Quisqater signature scheme in the strong existential unforgeability game from Def. 3.20. Computing this probability is required to compute the success probability of the adversary $\mathcal{A}$ from the attack from the proof of Lem. 4.9 on the Guillou-Quisqater signature scheme [GQ88]. This is because if $A_{N,e,c} = \emptyset$ (for the tuple (N, e) that $\mathcal{A}$ is passed as part of the public parameters and the first element c of the signature σ returned by the signing oracle), then $\mathcal{A}$ cannot win this instance of the sEUF-CMA game Def. 3.20 for Guillou-Quisqater signatures.

Adaptability To be able to use adapted signatures like "normal", fresh signatures, we need those two types of signatures to be indistinguishable. Formally, this translates to identical distributions of fresh and adapted signatures, captured in the following two definitions which are based on [DS16].

Definition 4.10 (**adaptability**) *Let Σ_{KH} be a key-homomorphic signature scheme with secret-to-public-key homomorphism μ, secret key space H and message space $\mathcal{M}$.* Σ_{KH} provides adaptability of signatures *if for every security parameter $\lambda \in \mathbb{N}$, message $m \in \mathcal{M}$ and $\mathsf{pp} \leftarrow\!\!\$\ \mathsf{PGen}(1^\lambda)$, we have that*

$$v_1 := ((\mathsf{pk}, \mathsf{sk}), \mathsf{KH.Adapt}(\mathsf{pk}, m, \mathsf{KH.Sign}(\mathsf{sk}, m), \Delta))$$

with $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{KGen}(\mathsf{pp})$, $\Delta \leftarrow\!\!\$\ H$ *and*

$$v_2 := ((\mu(\mathsf{sk}), \mathsf{sk}), (\mu(\mathsf{sk}) \cdot \mu(\Delta), \mathsf{KH.Sign}(\mathsf{sk} + \Delta, m)))$$

with $\mathsf{sk} \leftarrow\!\!\$\ H$ *and* $\Delta \leftarrow\!\!\$\ H$ *are identically distributed.*

Adaptability in the above definition means that adapted and fresh signatures look the same together with the corresponding verification key and the original public key that it was derived from. A stronger notion is *perfect adaption* [DS16] which captures the above indistinguishability with the additional requirement that the original signature that the adapted one was computed from is known.

Definition 4.11 (**perfect adaptability**) *Let* Σ_{KH} *be a key-homomorphic signature scheme with secret-to-public-key homomorphism* μ*, secret key space* H *and message space* $\mathcal{M}$*.* Σ_{KH} provides perfect adaptability of signatures *if for every security parameter* $\lambda \in \mathbb{N}$*, message* $m \in \mathcal{M}$ *and* $\mathsf{pp} \leftarrow\!\!\$\ \mathsf{PGen}(1^\lambda)$*, we have that*

$$v_1 := (\sigma, (\mathsf{pk}, \mathsf{sk}), \mathsf{KH.Adapt}(\mathsf{pk}, m, \sigma, \Delta))$$

with $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{KGen}(1^\lambda)$, $\sigma \leftarrow\!\!\$\ \mathsf{KH.Sign}(\mathsf{sk}, m)$, $\Delta \leftarrow\!\!\$\ H$ *and*

$$v_2 := (\sigma, (\mu(\mathsf{sk}), \mathsf{sk}), (\mu(\mathsf{sk}) \cdot \mu(\Delta), \mathsf{KH.Sign}(\mathsf{sk} + \Delta, m))$$

with $\mathsf{sk} \leftarrow\!\!\$\ H$ *and* $\Delta \leftarrow\!\!\$\ H$*,* $\sigma \leftarrow\!\!\$\ \mathsf{KH.Sign}(\mathsf{sk}, m)$ *are identically distributed.*

BLS signatures As an example for perfectly adaptable key-homomorphic signatures, we will consider BLS signatures [BLS04]. We formally define the scheme and then prove its adaptability according to Def. 4.11.

Definition 4.12 (**BLS signature scheme**) *Let* $\{H_k\}_k$ *be a hash function family. The* BLS signature scheme Σ_{BLS} *is a digital signature scheme with message space* $\mathcal{M}$ *defined as follows:*

$\mathsf{PGen}(1^\lambda)$

1: $\mathrm{BG} := (G_1, G_2, G_T, p, e, g_1, g_2) \leftarrow\!\!\$\ \mathsf{BGGen}(1^\lambda, 3)$
2: $(H : \mathcal{M} \to G_1) \quad \leftarrow\!\!\$\ \{H_k\}_k$
3: **return** $\mathsf{pp} := (\mathrm{BG}, H)$

$\mathsf{KGen}(\mathsf{pp})$

1: ***parse*** $(BG, H) := \mathsf{pp}$
2: $x \leftarrow\!\!\$\ \mathbb{Z}_p$
3: $\mathsf{pk} := g_2^x$
4: $\mathsf{sk} := x$
5: **return** $(\mathsf{pk}, \mathsf{sk})$

$\mathsf{Sign}(\mathsf{sk}, m)$

1: ***parse*** $\mathsf{sk} =: x$
2: $\sigma := H(m)^x$
3: **return** σ

$\mathsf{Vfy}(\mathsf{pk}, m, \sigma)$

1: ***parse*** $g_2^x := \mathsf{pk}$
2: **return** 1 **if** $e(H(m), g_2^x) = e(\sigma, g_2)$
3: **else return** 0

Observe that the BLS signature scheme has a deterministic signing algorithm. The hash function used in it is modeled as a random oracle for the security analysis in [BLS04]. To prepare the proof of the perfect adaptability of BLS signatures, we first prove that they indeed are a correct digital signature scheme.

Lemma 4.13 *Σ_{BLS} is a correct digital signature scheme (with secret key space $\mathbb{Z}_p$, public key space G_2 signature space G_1) according to Def. 3.17.*

Proof. The key and signature spaces are obvious from inspection. Let $\lambda \in \mathbb{N}$, $\mathrm{BG} := (G_1, G_2, G_T, p, e, g_1, g_2) \leftarrow\!\$\ \mathsf{BGGen}(1^\lambda, 3)$, $H : \mathcal{M} \to G_1 \leftarrow\!\$\ \{H_k\}_k$, $\mathsf{pp} := (\mathrm{BG}, H)$, $x \leftarrow\!\$\ \mathbb{Z}_p$, $\mathsf{pk} := g_2^x$, $\mathsf{sk} := x$, $\sigma := H(m)^x$, $m \in \mathcal{M}$. Exploiting the bilinearity of the pairing e, we get

$$e(H(m), g_2^x) = e(H(m), g_2)^x = e(H(m)^x, g_2) = e(\sigma, g_2)$$

which implies $\Pr[\mathsf{Vfy}(\mathsf{pk}, m, \sigma) = 1] = 1$, so Σ_{BLS} is correct according to Def. 3.17. □

We next prove the perfect adaptability of BLS signatures. Note that while the $\mathsf{KH.Adapt}$ algorithm was given in [DS16], the proof that the two views v_1, v_2 from Def. 4.11 are identically distributed was only sketched. The basic idea of adaption for BLS signatures is to sign the message with the shift amount and compute the public key corresponding to the shift amount. Then the signatures and public keys are multiplied using the respective group operation. Although it was omitted in [DS16], we first need to prove that Σ_{BLS} is a key-homomorphic signature scheme according to Def. 4.2.

Lemma 4.14 *Σ_{BLS} is perfectly adaptable according to Def. 4.11.*

Proof. We first prove that Σ_{BLS} is a key-homomorphic signature scheme according to Def. 4.2.

The correctness of Σ_{BLS} as a digital signature scheme was proven in Lem. 4.13. Furthermore, by definition of KGen and a simple computation, $\mu : \mathbb{Z}_p \to G_2, x \mapsto g_2^x$ is a secret-to-public-key homomorphism for Σ_{BLS}.

We next define the $\mathsf{KH.Adapt}$ algorithm for Σ_{BLS} from [DS16]:

$\mathsf{KH.Adapt}(\mathsf{pk}, m, \sigma, \Delta)$

1 : **parse** $g_2^x := \mathsf{pk}$
2 : $\mathsf{pk}' := g_2^x g_2^\Delta$
3 : $\sigma' := \sigma \cdot H(m)^\Delta$
4 : **return** (pk', σ')

Before proving perfect adaptability according to Def. 4.11, we need to prove that the $\mathsf{KH.Adapt}$ algorithm fulfills the requirements from Def. 4.2. Let $\lambda, \mathrm{BG}, H, x, \mathsf{pk}, \mathsf{sk}, \sigma, m$ as in Lem. 4.13. Furthermore, assume $\mathsf{Vfy}(\mathsf{pk}, m, \sigma) = 1$ and $(\mathsf{pk}', \sigma') \leftarrow\!\$\ \mathsf{KH.Adapt}(\mathsf{pk}, m, \sigma, \Delta)$. Exploiting the secret-to-public-key homomorphism property of μ and the fact that the

prime order group G_2 is abelian (Lem. 3.8), we get

$$\mathsf{pk}' = g_2^x \cdot g_2^{\Delta} = \mu(x) \cdot \mu(\Delta) = \mathsf{pk} \cdot \mu(\Delta) = \mu(\Delta) \cdot \mathsf{pk}$$

furthermore, using the bilinearity of the pairing e, we have

$$\begin{aligned} & e(H(m), g_2^x \cdot g_2^{\Delta}) = e(H(m)^x \cdot H(m)^{\Delta}, g_2) \\ \Leftrightarrow & e(H(m), g_2^{x+\Delta}) = e(H(m)^{x+\Delta}, g_2) \\ \Leftrightarrow & e(H(m), g_2)^{x+\Delta} = e(H(m), g_2)^{x+\Delta} \end{aligned}$$

which proves the validity of (m, σ') under pk'. So Σ_{BLS} is a key-homomorphic signature scheme according to Def. 4.2.

Next we prove its perfect adaptability according to Def. 4.11. Let $\lambda, \mathrm{BG}, H, \mathsf{pp}, m$ as above. Define

$$v_1 := (v_{1,1}, v_{1,2}, v_{1,3}, v_{1,4}, v_{1,5}) = (\sigma, (\mathsf{pk}, \mathsf{sk}), (\mathsf{pk}', \sigma'))$$

with $(\mathsf{sk}, \mathsf{pk}) \leftarrow\!\!\$\ \mathsf{KGen}(\mathsf{pp})$, $\sigma \leftarrow\!\!\$\ \mathsf{Sign}(\mathsf{sk}, m)$, $(\mathsf{pk}', \sigma') \leftarrow\!\!\$\ \mathsf{KH.Adapt}(\mathsf{pk}, m, \sigma, \Delta)$, $\Delta \leftarrow\!\!\$\ \mathbb{Z}_p$. Define

$$v_2 := (v_{2,1}, v_{2,2}, v_{2,3}, v_{2,4}, v_{2,5}) := (\sigma, (\mu(\mathsf{sk}), \mathsf{sk}), (\mu(\mathsf{sk}) \cdot \mu(\Delta), \mathsf{Sign}(\mathsf{sk} + \Delta, m)))$$

with $\mathsf{sk} \leftarrow\!\!\$\ \mathbb{Z}_p$ and $\Delta \leftarrow\!\!\$\ \mathbb{Z}_p$ and $\sigma \leftarrow\!\!\$\ \mathsf{Sign}(\mathsf{sk}, m)$.

By definition of KGen of Σ_{BLS}, we see that $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{KGen}(\mathsf{pp})$ is identically distributed to $(\mu(\mathsf{sk}), \mathsf{sk})$ with $\mathsf{sk} \leftarrow\!\!\$\ \mathbb{Z}_p$. So $(\sigma, (\mathsf{pk}, \mathsf{sk}))$ in v_1 and $(\sigma, (\mu(\mathsf{sk}), \mathsf{sk}))$ in v_2 are identically distributed.

The $v_{1,4}$ and $v_{2,4}$ are adapted versions of the public keys $v_{1,2}$ and $v_{2,2}$, respectively. Since for $v_{1,2}$, we have

$$v_{1,2} = \mathsf{pk}' = g_2^{\mathsf{sk}} \cdot g_2^{\Delta} = \mu(\mathsf{sk}) \cdot \mu(\Delta)$$

which yields that $(\sigma, (\mathsf{pk}, \mathsf{sk}), \mathsf{pk}')$ in v_1 and $(\sigma, (\mu(\mathsf{sk}), \mathsf{sk}), \mu(\mathsf{sk}) \cdot \mu(\Delta))$ in v_2 are identically distributed.

By definition of the (deterministic) signing algorithm Sign of Σ_{BLS}, we have

$$\sigma' = \sigma \cdot H(m)^{\Delta} = H(m)^{\mathsf{sk}} \cdot H(m)^{\Delta} = H(m)^{\mathsf{sk}+\Delta} = \mathsf{Sign}(\mathsf{sk} + \Delta, m)$$

so the $v_{1,5}$ and $v_{2,5}$ are only depending on $(v_{1,1}, v_{1,2}, v_{1,3}, v_{1,4})$ and $(v_{2,1}, v_{2,2}, v_{2,3}, v_{2,4})$, respectively. So since $(v_{1,1}, v_{1,2}, v_{1,3}, v_{1,4})$ and $(v_{2,1}, v_{2,2}, v_{2,3}, v_{2,4})$ are identically distributed, we get that v_1 and v_2 are identically distributed.

Overall, Σ_{BLS} is perfectly adaptable key-homomorphic signature scheme according to Def. 4.11. □

Derler and Slamanig [DS16] have sketched adaptability proofs for other signature schemes than BLS (see Rem. 4.7 for a list) by providing the respective $\mathsf{KH.Adapt}$ algo-

rithm and sketching the distribution argument needed for adaptability.

4.1.2 Publicly key-homomorphic signatures

Next, we define *publicly key-homomorphic signature schemes* which basically allow to publicly combine multiple signatures on the same message into one. This is another flavour of key-homomorphic signatures that was introduced by Derler and Slamanig in [DS16]. We will start with the basic syntax and correctness of such signature schemes.

Definition 4.15 (**publicly key-homomorphic signature schemes**) *Let* $(H,+)$, $(E,\cdot)$ *groups. A* publicly key-homomorphic signature scheme *is a tuple of ppt algorithms* $\Sigma_{\mathsf{PKH}} = (\mathsf{PKH.PGen}, \mathsf{PKH.KGen}, \mathsf{PKH.Sign}, \mathsf{PKH.Vfy}, \mathsf{PKH.Combine})$ *fulfilling the following requirements:*

(i) $\Sigma := (\mathsf{PKH.PGen}, \mathsf{PKH.KGen}, \mathsf{PKH.Sign}, \mathsf{PKH.Vfy})$ *is a correct digital signature scheme with public key space* E*, secret key space* H*, some message space* $\mathcal{M}$ *and some signature space* $\mathcal{S}$*.*

(ii) Σ *provides a secret-to-public-key homomorphism* μ

(iii) $\mathsf{PKH.Combine}$ *is a ppt combination algorithm which takes as input public keys* $\mathsf{pk}_1, \ldots, \mathsf{pk}_n \in E$*, a message* $m \in \mathcal{M}$*, signatures* $\sigma_1, \ldots, \sigma_n \in \mathcal{S}$ *and outputs a combined public key* $\hat{\mathsf{pk}} \in E$ *and a signature* $\hat{\sigma} \in \mathcal{S}$ *that satisfy the following requirement: For all* $n > 1$*,* $\mathsf{pp} \leftarrow\!\!\$\ \mathsf{PGen}(1^\lambda)$*, message* $m \in \mathcal{M}$*,* $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow\!\!\$\ \mathsf{KGen}(\mathsf{pp})$*,* $\sigma_i \leftarrow\!\!\$\ \mathsf{Sign}(\mathsf{sk}_i, m)$ *for* $1 \le i \le n$*,* $(\hat{\mathsf{pk}}, \hat{\sigma}) \leftarrow\!\!\$\ \mathsf{PKH.Combine}((\mathsf{pk}_i)_{i=1}^n, m, (\sigma_i)_{i=1}^n)$*:*

$$\hat{\mathsf{pk}} = \Pi_{i=1}^n \mathsf{pk}_i \wedge \Pr[\mathsf{PKH.Vfy}(\hat{\mathsf{pk}}, m, \hat{\sigma}) = 1] = 1$$

So the distinctive correctness requirement for publicly key-homomorphic signatures is that combined signatures are valid under combined public keys. Furthermore, if not explicitly stated otherwise, we always assume that the algorithms of a publicly key-homomorphic signature scheme are named as in Def. 4.15.

Public adaptability Analogously to key-homomorphic signatures (Sect. 4.1.1), we can define adaptability for publicly key-homomorphic signature schemes. Such a scheme is (publicly) adaptable if combined signatures look the same as fresh signatures under manually combined public keys.

Definition 4.16 (**public adaptability**) *Let* Σ_{PKH} *be a publicly key-homomorphic signature scheme with secret-to-public-key homomorphism* μ*, secret key space* $(H,+)$ *and message space* $\mathcal{M}$*.* Σ_{PKH} provides public adaptability of signatures *if for every security parameter* $\lambda \in \mathbb{N}$*,* $n \in \mathsf{poly}(\lambda)$*,* $\mathsf{pp} \leftarrow\!\!\$\ \mathsf{PKH.PGen}(1^\lambda)$*,* $m \in \mathcal{M}$ *it holds that*

$$v_1 := (((\mathsf{pk}_i, \mathsf{sk}_i))_{i=1}^n, \mathsf{PKH.Combine}((\mathsf{pk}_i)_{i=1}^n, m, (\sigma_i)_{i=1}^n))$$

with $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow\!\$\ \mathsf{PKH.KGen}(\mathsf{pp})$, $\sigma_i \leftarrow\!\$\ \mathsf{PKH.Sign}(\mathsf{sk}_i, m)$ *for* $1 \leq i \leq n$ *and*

$$v_2 := (((\mathsf{pk}_i, \mathsf{sk}_i))_{i=1}^n, (\Pi_{i=1}^n \mathsf{pk}_i, \mathsf{PKH.Sign}(\Sigma_{i=1}^n \mathsf{sk}_i, m)))$$

with $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow\!\$\ \mathsf{PKH.KGen}(\mathsf{pp})$ *for* $1 \leq i \leq n$ *are identically distributed.*

Again, analogously to key-homomorphic signatures (Sect. 4.1.1), we define a stronger notion called *perfect public adaptability*. It requires that combined and fresh signatures are indistinguishable even in the case that the original signatures that were combined are known.

Definition 4.17 (perfect public adaptability) *Let* Σ_{PKH} *be a publicly key-homomorphic signature scheme with secret key space* $(H, +)$ *and message space* $\mathcal{M}$. Σ_{PKH} provides perfect public adaptability of signatures *if for every security parameter* $\lambda \in \mathbb{N}$, $n \in \mathsf{poly}(\lambda)$, $\mathsf{pp} \leftarrow\!\$\ \mathsf{PKH.PGen}(1^\lambda)$, $m \in \mathcal{M}$ *it holds that*

$$v_1 := (((\mathsf{pk}_i, \mathsf{sk}_i, \sigma_i))_{i=1}^n, \mathsf{PKH.Combine}((\mathsf{pk}_i)_{i=1}^n, m, (\sigma_i)_{i=1}^n))$$

with $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow\!\$\ \mathsf{PKH.KGen}(\mathsf{pp})$, $\sigma_i \leftarrow\!\$\ \mathsf{PKH.Sign}(\mathsf{sk}_i, m)$ *for* $1 \leq i \leq n$ *and*

$$v_2 := (((\mathsf{pk}_i, \mathsf{sk}_i, \sigma_i))_{i=1}^n, (\Pi_{i=1}^n \mathsf{pk}_i, \mathsf{PKH.Sign}(\Sigma_{i=1}^n \mathsf{sk}_i, m)))$$

with $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow\!\$\ \mathsf{PKH.KGen}(\mathsf{pp})$, $\sigma_i \leftarrow\!\$\ \mathsf{PKH.Sign}(\mathsf{sk}_i, m)$ *for* $1 \leq i \leq n$ *are identically distributed.*

BLS signatures revisited In the following, we revisit the BLS signature scheme (Def. 4.12) and prove that it is perfectly publicly adaptable. Again, we are using the algorithm idea for the PKH.Combine algorithm from [DS16] but do the full proof instead of just presenting the PKH.Combine algorithm as in [DS16].

Lemma 4.18 Σ_{BLS} *from Def. 4.12 is perfectly publicly adaptable according to Def. 4.17.*

Proof. Σ_{BLS} is a key-homomorphic signature scheme (Def. 4.2) according to Lem. 4.14. So Σ_{BLS} is a digital signature scheme that provides a secret-to-public key homomorphism $\mu : \mathbb{Z}_p \to G_2, x \mapsto g_2^x$. What is left to prove is that a PKH.Combine algorithm with the properties from Def. 4.15 and Def. 4.17 exists for Σ_{BLS},

Define the PKH.Combine algorithm as follows:

$\mathsf{PKH.Combine}((\mathsf{pk}_i)_{i=1}^n, m, (\sigma_i)_{i=1}^n)$

1 : **parse** $g_2^{x_i} := \mathsf{pk}_i$ **for** $1 \leq i \leq n$
2 : $\hat{\mathsf{pk}} = \Pi_{i=1}^n \mathsf{pk}_i$
3 : $\hat{\sigma} = \Pi_{i=1}^n \sigma_i$
4 : **return** $(\hat{\mathsf{pk}}, \hat{\sigma})$

Let $n > 1$, $\lambda \in \mathbb{N}$, $\mathrm{BG} := (G_1, G_2, G_T, p, e, g_1, g_2) \leftarrow\!\$\ \mathsf{BGGen}(1^\lambda, 3)$, $H \leftarrow\!\$\ \{H_k\}_k$, $\mathsf{pp} := (\mathrm{BG}, H)$,$(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow\!\$\ \mathsf{PKH.KGen}(\mathsf{pp})$ for $1 \leq i \leq n$, $m \in \mathcal{M}$, $\sigma_i \leftarrow\!\$\ \mathsf{Sign}(\mathsf{sk}_i, m)$

for $1 \leq i \leq n$, $(\hat{\mathsf{pk}}, \hat{\sigma}) \leftarrow\!\!\$\ \mathsf{PKH.Combine}((\mathsf{pk}_i)_{i=1}^n, m, (\sigma_i)_{i=1}^n)$. We have

$$\hat{\mathsf{pk}} = \Pi_{i=1}^n g_2^x = \Pi_{i=1}^n \mathsf{pk}_i$$

furthermore

$$\begin{aligned}
& e(H(m), \hat{\mathsf{pk}}) = e(\hat{\sigma}, g_2) \\
\Leftrightarrow & e(H(m), \Pi_{i=1}^n g_2^{x_i}) = e(\Pi_{i=1}^n \sigma_i, g_2) \\
\Leftrightarrow & e(H(m), g_2^{\Sigma_{i=1}^n x_i}) = e(\Pi_{i=1}^n \sigma_i, g_2) \\
\Leftrightarrow & e(H(m), g_2)^{\Sigma_{i=1}^n x_i} = e(\Pi_{i=1}^n H(m)^{x_i}, g_2) \\
\Leftrightarrow & e(H(m), g_2)^{\Sigma_{i=1}^n x_i} = e(H(m)^{\Sigma_{i=1}^n x_i}, g_2) \\
\Leftrightarrow & e(H(m), g_2)^{\Sigma_{i=1}^n x_i} = e(H(m), g_2)^{\Sigma_{i=1}^n x_i}
\end{aligned}$$

which yields $\Pr[\mathsf{PKH.Vfy}(\hat{\mathsf{pk}}, m, \hat{\sigma}) = 1] = 1$. So we have proven the correctness requirement for $\mathsf{PKH.Combine}$ from Def. 4.15, what is left to prove is the perfect public adaptability of Σ_{BLS} according to Def. 4.17. Let $\lambda, \mathrm{BG}, H, \mathsf{pp}$ as above, $n = \mathsf{poly}(\lambda)$, $m \in \mathcal{M}$. Define

$$v_1 = (v_{1,1}, v_{1,2}, v_{1,3}) := ((\mathsf{sk}_i, \mathsf{pk}_i, \sigma_i)_{i=1}^n, (\hat{\mathsf{pk}}, \hat{\sigma}))$$

where $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow\!\!\$\ \mathsf{PKH.KGen}(\mathsf{pp})$, $\sigma_i \leftarrow\!\!\$\ \mathsf{PKH.Sign}(\mathsf{sk}_i, m)$ for $1 \leq i \leq n$, $(\hat{\mathsf{pk}}, \hat{\sigma}) \leftarrow\!\!\$\ \mathsf{PKH.Combine}((\mathsf{pk}_i)_{i=1}^n, m, (\sigma_i)_{i=1}^n)$ and

$$v_2 = (v_{2,1}, v_{2,2}, v_{2,3}) := ((\mathsf{sk}_i, \mathsf{pk}_i, \sigma_i)_{i=1}^n, (\Pi_{i=1}^n \mathsf{pk}_i, \mathsf{PKH.Sign}(\Sigma_{i=1}^n \mathsf{sk}_i, m)))$$

where $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow\!\!\$\ \mathsf{PKH.KGen}(\mathsf{pp})$, $\sigma_i \leftarrow\!\!\$\ \mathsf{PKH.Sign}(\mathsf{sk}_i, m)$ for $1 \leq i \leq n$.

We see that the key-pair-signature triples $v_{1,1}$ and $v_{2,1}$ are identically distributed by definition. Furthermore

$$\hat{\mathsf{pk}} = \Pi_{i=1}^n g_2^{x_i} = \Pi_{i=1}^n \mathsf{pk}_i$$

so the public keys $v_{1,2}$ and $v_{2,2}$ are the product of the public keys from $v_{1,1}$ and $v_{1,2}$, respectively. So $(v_{1,1}, v_{1,2})$ and $(v_{2,1}, v_{2,2})$ are identically distributed.

Finally, the fact that $\mathsf{PKH.Sign}$ is deterministic for Σ_{BLS} yields that

$$\hat{\sigma} = \Pi_{i=1}^n \sigma_i = \Pi_{i=1}^n H(m)^{\mathsf{sk}_i} = H(m)^{\Sigma_{i=1}^n \mathsf{sk}_i}$$

so $v_{1,3}$ and $v_{2,3}$ are identically distributed and only depend on $(v_{1,1}, v_{1,2})$ and $(v_{2,1}, v_{2,2})$. Since $(v_{1,1}, v_{1,2})$ and $(v_{2,1}, v_{2,2})$ are identically distributed, $v_1 = (v_{1,1}, v_{1,2}, v_{1,3})$ and $v_2 = (v_{2,1}, v_{2,2}, v_{2,3})$ are identically distributed.

So Σ_{BLS} is a perfectly publicly adaptable key-homomorphic signature scheme according to Def. 4.17. □

4.2 Structure-preserving signatures on equivalence classes

In this chapter we introduce *structure-preserving signatures on equivalence classes (SPS-EQ)* which were first defined by Hanser and Slamanig in [HS14] and first securely instantiated in [FHS14]. An SPS-EQ is a signature scheme that is equipped with an equivalence relation R on its message space. One can subsequently randomize a message-signature pair (m, σ) to a new one (m', σ') where $m' \in [m]_R$ is a distinct representative of the class of m. This of course requires a change in the unforgeability game if it should stay meaningful. To break unforgeability, an adversary must sign a message m^* from a class that he has not seen any signed message for, since computing valid signatures for different representatives of some already-signed class is trivial in an SPS-EQ. Besides unforgeability, secure SPS-EQ provide *class-hiding* which basically means that there is no efficient way to tell whether two given messages m and m' are from the same class or not. An additional security notion for SPS-EQ is *perfect adaption of signatures* (see for example [BHKS18]) which demands that fresh and randomized SPS-EQ signatures look alike.

SPS-EQ can be seen as a complementary primitive to the flexible public key signatures from [BHKS18] which have an equivalence relation with analogous properties on their public key space instead of their message space. In the following, we formally introduce SPS-EQ and their correctness and security requirements. If not otherwise stated, we roughly follow Connolly et al. [CLPK22].

Definition 4.19 (**SPS-EQ syntax**) *Let G be a group, $\mathcal{M} := G^l$ for some natural $l > 1$, $R \subseteq \mathcal{M} \times \mathcal{M}$ an equivalence relation. An* structure-preserving signature scheme on equivalence classes (SPS-EQ) *with message space $\mathcal{M}$, public key space E, secret key space H, signature space $\mathcal{S}$ and trapdoor space T is a tuple Σ_{SPSEQ} of ppt algorithms as follows:*

$\mathsf{SPSEQ.PGen}(1^\lambda)$ *probablistic parameter generation algorithm, takes as input the security parameter $\lambda \in \mathbb{N}$ and outputs public parameters* pp

$\mathsf{SPSEQ.TPGen}(1^\lambda)$ *probablistic trapdoor parameter generation algorithm, takes as input the security parameter $\lambda \in \mathbb{N}$ and outputs public parameters* pp *and an associated trapdoor $\tau \in T$*

$\mathsf{SPSEQ.KGen}(\mathsf{pp}, l)$ *probablistic key generation algorithm, takes as input public parameters* pp *and the vector length $l \in \mathbb{N}$ and outputs a key pair $(\mathsf{pk}, \mathsf{sk}) \in E \times H$, consisting of a public key* pk *and a secret key* sk

$\mathsf{SPSEQ.Sign}(\mathsf{sk}, m)$ *probablistic signing algorithm, takes as input a secret key $\mathsf{sk} \in H$ and a message $m \in \mathcal{M} = G^l$ and outputs a signature $\sigma \in \mathcal{S}$*

$\mathsf{SPSEQ.ConvSigGen}(\mathsf{pp})$ *probablistic signature conversion randomness generation algorithm, takes as input public parameters* pp *and outputs signature conversion randomness α*

$\mathsf{SPSEQ.ChgRep}(m, \sigma, \alpha, \mathsf{pk})$ *probablistic change-representative algorithm, takes as input a message* $m \in \mathcal{M} = G^l$*, a signature* $\sigma \in \mathcal{S}$*, a scalar* α *and a public key* pk *and outputs an adapted message-signature pair* $(m', \sigma') \in \mathcal{M} \times \mathcal{S}$

$\mathsf{SPSEQ.Vfy}(\mathsf{pk}, m, \sigma)$ *deterministic verification algorithm, takes as input a public key* $\mathsf{pk} \in E$*, a message* $m \in \mathcal{M}$ *and a signature* $\sigma \in \mathcal{S}$ *and outputs a bit* $b \in \{0, 1\}$

$\mathsf{SPSEQ.VKey}(\mathsf{pk}, \mathsf{sk})$ *deterministic key pair verification algorithm, takes as input a public key* $\mathsf{pk} \in E$ *and a secret key* $\mathsf{sk} \in H$ *and outputs a bit* $b \in \{0, 1\}$

If not explicitly stated otherwise, we always assume that the algorithms of an SPS-EQ are named like in Def. 4.19. Conolly et al [CLPK22] defined SPS-EQ for signatures that possibly have a tag t associated with them, which we omitted here for simplicity. The set of values α that can be input into the $\mathsf{SPSEQ.ChgRep}$ algorithm as randomness is implicitly defined by the particular SPS-EQ and contains the support of $\mathsf{SPSEQ.ConvSigGen}$ as a subset. The $\mathsf{SPSEQ.VKey}$ algorithm was missing in [CLPK22], it however is required to formally include valid key pairs that cannot be output by the key generation algorithm into the following SPS-EQ correctness definition.

Definition 4.20 (**SPS-EQ correctness**) *Let* G *be a group,* $\mathcal{M} := G^l$ *for some* $l > 1$*,* $R \subseteq \mathcal{M} \times \mathcal{M}$ *an equivalence relation. Furthermore let* Σ_{SPSEQ} *be an SPS-EQ with message space* $\mathcal{M}$*, equivalence relation* R*, public key space* E*, secret key space* H*, signature space* $\mathcal{S}$ *and trapdoor space* T*.* Σ_{SPSEQ} *is* correct *if the following requirements are met:*

(i) For all $\lambda \in \mathbb{N}$*,* $\mathsf{pp} \leftarrow\!\!\$\ \mathsf{SPSEQ.PGen}(\lambda)$*,* $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{SPSEQ.KGen}(\mathsf{pp}, l)$*, we have* $\mathsf{SPSEQ.VKey}(\mathsf{pk}, \mathsf{sk}) = 1$*.*

(ii) For all $\lambda \in \mathbb{N}$*,* $\mathsf{pp} \leftarrow\!\!\$\ \mathsf{SPSEQ.PGen}(\lambda)$ *and* pp *in* $(\mathsf{pp}, \cdot) \leftarrow\!\!\$\ \mathsf{SPSEQ.TPGen}(1^\lambda)$ *are identically distributed.*

(iii) For all $\lambda \in \mathbb{N}$*,* $\mathsf{pp} \leftarrow\!\!\$\ \mathsf{SPSEQ.PGen}(\lambda)$*,* $(\mathsf{pk}, \mathsf{sk}) \in E \times H$*,* $\mathsf{SPSEQ.VKey}(\mathsf{pk}, \mathsf{sk}) = 1$*,* $m \in \mathcal{M}$*,* $\alpha \leftarrow\!\!\$\ \mathsf{SPSEQ.ConvSigGen}(\mathsf{pp})$*,* $\sigma \leftarrow\!\!\$\ \mathsf{SPSEQ.Sign}(\mathsf{sk}, m)$*,* $(m', \sigma') \leftarrow\!\!\$\ \mathsf{SPSEQ.ChgRep}(m, \mathsf{SPSEQ.Sign}(\mathsf{sk}, m), \alpha, \mathsf{pk})$*, we have*

a) $\Pr[\mathsf{SPSEQ.Vfy}(\mathsf{pk}, m, \sigma) = 1] = 1$

b) $\Pr[\mathsf{SPSEQ.Vfy}(\mathsf{pk}, m', \sigma') = 1] = 1$

c) $m' \in [m]_R$

The first requirement states that honestly generated key pairs are correct, i.e. can be validated using $\mathsf{SPSEQ.VKey}$. The second requirement demands that parameters generated with and without associated trapdoor look the same. This requirement was not explicitly listed in [CLPK22]. Note that the original SPS-EQ syntax definition from [HS14, FHS14] did not contain the trapdoor parameter generation algorithm $\mathsf{SPSEQ.TPGen}$ since the SPS-EQ constructions from these papers did not require a trapdoor for the public parameters. We listed the $\mathsf{SPSEQ.TPGen}$ algorithm in this paper to include SPS-EQ schemes that do need such a trapdoor into our framework. If no trapdoor is needed, the empty string ϵ can be returned by $\mathsf{SPSEQ.TPGen}$, making $\mathsf{SPSEQ.PGen}$

and SPSEQ.TPGen technically identical. The third requirement comprises regular digital signature correctness as well as the validity of randomized message-signature pairs obtained from valid ones via SPSEQ.ChgRep.

Unforgeability Now that we are done with syntax and correctness of SPS-EQ, we can turn towards their security. As already pointed out in the introduction of this section, the winning condition of the standard EUF-CMA game from Def. 3.19 needs an adjustment for EUF-CMA to be meaningful for SPS-EQ. More precisely, the adversary should not be allowed to submit a forgery for a representative of any class of messages that he has already seen a signature for. This is because computing such a forgery can be done trivially using the SPSEQ.ChgRep algorithm.

Definition 4.21 (**SPS-EQ EUF-CMA**) *Let $\lambda \in \mathbb{N}$ be the security parameter, $l > 1$ the vector length, Σ_{SPSEQ} an SPS-EQ over some equivalence relation R. We define the following security game for and adversary $\mathcal{A}$ and a challenger $\mathcal{C}$:*

$\mathsf{Exp}^{\mathsf{spseq\text{-}euf}}_{\mathcal{A},\Sigma_{\mathsf{SPSEQ}}}(\lambda)$

1: $\mathsf{pp} \leftarrow\!\$\ \mathsf{SPSEQ.PGen}(1^\lambda)$
2: $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\$\ \mathsf{KGen}(\mathsf{pp}, l)$
3: $(m^*, \sigma^*) \leftarrow\!\$\ \mathcal{A}^{\mathsf{Sign}(\mathsf{sk},\cdot)}(\mathsf{pk})$
4: **return** 1 **if** $(\forall m \in Q : m^* \notin [m]_R) \wedge \mathsf{SPSEQ.Vfy}(\mathsf{pk}, m^*, \sigma^*) = 1$
5: **else return** 0

with $\mathsf{Sign}(\mathsf{sk}, \cdot)$ *being a signing oracle as of the following:*

$\mathsf{Sign}(\mathsf{sk}, m)$

1: $\sigma \leftarrow\!\$\ \mathsf{SPSEQ.Sign}(\mathsf{sk}, m)$
2: $Q := Q \cup \{m\}$
3: **return** σ

We define the advantage of $\mathcal{A}$ in the above security game as

$$\mathsf{Adv}^{\mathsf{spseq\text{-}euf}}_{\mathcal{A},\Sigma_{\mathsf{SPSEQ}}}(\lambda) := \Pr[\mathsf{Exp}^{\mathsf{spseq\text{-}euf}}_{\mathcal{A},\Sigma_{\mathsf{SPSEQ}}}(\lambda) = 1]$$

Σ_{SPSEQ} *is* existentially unforgeable under chosen-message attacks (EUF-CMA secure) *if for all ppt adversaries $\mathcal{A}$ $\mathsf{Adv}^{\mathsf{spseq\text{-}euf}}_{\mathcal{A},\Sigma_{\mathsf{SPSEQ}}}(\lambda)$ is negligible.*

Class-hiding Next, we define class-hiding for SPS-EQ. Intuitively, there should be no efficient way to tell whether two given messages m and m' are related via the message relation R or not. We define two distinct class-hiding notions. The first one is based on the one given in [FHS14] by Fuchsbauer et al. which corrects a severe error that Hanser and Slamanig overlooked when they first presented such a class-hiding notion in [HS14].

Definition 4.22 (**SPS-EQ real-or-random class-hiding**) *Let $\lambda \in \mathbb{N}$ be the security parameter, $l > 1$ the vector length, G a group, Σ_{SPSEQ} an SPS-EQ with message space*

$\mathcal{M} := G^l$ *over some equivalence relation* R*. We define the following security game for and adversary* $\mathcal{A}$ *and a challenger* $\mathcal{C}$*:*

$\mathsf{Exp}^{\mathsf{spseq\text{-}ror\text{-}ch}}_{\mathcal{A},\Sigma_{\mathsf{SPSEQ}}}(\lambda)$

1 : $\mathsf{pp} \leftarrow\!\$\ \mathsf{SPSEQ.PGen}(1^\lambda)$
2 : $b \leftarrow\!\$\ \{0,1\}$
3 : $(\mathsf{state}, \mathsf{pk}, \mathsf{sk}) \leftarrow\!\$\ \mathcal{A}(\mathsf{pp}, l)$
4 : $b' \leftarrow\!\$\ \mathcal{A}^{\mathcal{O}_{\mathsf{msg}}(l),\mathcal{O}_{\mathsf{ror}}(\mathsf{sk},\mathsf{pk},b,\cdot,\cdot)}(\mathsf{state}, \mathsf{pk}, \mathsf{sk})$
5 : **return** 1 **if** $b = b' \wedge \mathsf{SPSEQ.VKey}(\mathsf{pk}, \mathsf{sk}) = 1$
6 : **else return** 0

with oracles

$\mathcal{O}_{\mathsf{msg}}(l)$

1 : $m \leftarrow\!\$\ \mathcal{M}$
2 : $Q := Q \cup \{m\}$
3 : **return** m

$\mathcal{O}_{\mathsf{ror}}(\mathsf{sk}, \mathsf{pk}, b, m, \sigma)$

1 : ***if*** $m \notin Q \vee \mathsf{SPSEQ.Vfy}(\mathsf{pk}, m, \sigma) = 0$
2 : ***return*** $\perp$
3 : ***if*** $\mathcal{O}_{\mathsf{ror}}$ *was not called before*
4 : $r \leftarrow\!\$\ \mathcal{M}$
5 : $M \leftarrow\!\$\ ((m, \sigma), (r, \mathsf{SPSEQ.Sign}(\mathsf{sk}, r)))$
6 : *persistently store* $\bar{m} := m$ *and* $M[b]$
7 : ***return***
8 : ***else***
9 : ***if*** $m \neq \bar{m}$
10 : ***return*** $\perp$
11 : ***else***
12 : $\alpha \leftarrow\!\$\ \mathsf{SPSEQ.ConvSigGen}(\mathsf{pp})$
13 : ***return*** $\mathsf{SPSEQ.ChgRep}(M[b], \alpha, \mathsf{pk})$

We define the advantage of $\mathcal{A}$ *in the above security game as*

$$\mathsf{Adv}^{\mathsf{spseq\text{-}ror\text{-}ch}}_{\mathcal{A},\Sigma_{\mathsf{SPSEQ}}}(\lambda) = |Pr[\mathsf{Exp}^{\mathsf{spseq\text{-}ror\text{-}ch}}_{\mathcal{A},\Sigma_{\mathsf{SPSEQ}}}(\lambda) = 1] - \frac{1}{2}|$$

Σ_{SPSEQ} *is* real-or-random class-hiding *if for all ppt adversaries* $\mathcal{A}$ $\mathsf{Adv}^{\mathsf{spseq\text{-}ror\text{-}ch}}_{\mathcal{A},\Sigma_{\mathsf{SPSEQ}}}(\lambda)$ *is negligible.*

In the beginning of the game, the adversary $\mathcal{A}$ determines the challenge key pair $(\mathsf{pk}, \mathsf{sk})$. Note that since it is not the goal of the game to forge some signature, it is no problem that $\mathcal{A}$ knows the secret key that the challenger $\mathcal{C}$ will use. Depending on the bit b, the challenger $\mathcal{C}$ will then operate the real-or-random oracle $\mathcal{O}_{\mathsf{ror}}$ in one of two modes: in case $b = 0$, it will sign the first message m that $\mathcal{A}$ sends to it and upon subsequent oracle queries for that m, it will output randomized versions of the first created message-signature pair. In case $b = 1$, the message m that $\mathcal{A}$ submitted will be ignored, instead $\mathcal{C}$ will draw and sign a uniformly random message r. Subsequent calls

for m will be answered with randomized versions of the signed r, analogously to case $b = 0$. The adversary $\mathcal{A}$ must decide whether it has seen randomized signed versions of its message m or some random other message, possibly from a different class. If $\mathcal{A}$ cannot do so in reasonable time (i.e. polynomial time in the security parameter), it is hard to tell whether two given signed messages are in the same class or not. Note that if the adversary had the freedom to choose the challenge message $m \in G^l$ itself, it could choose a vector with very remarkable features (such as identical entries) that it might very easily distinguish from a signed random message. This is why $\mathcal{C}$ only accepts uniformly random input messages for $\mathcal{O}_{\mathsf{ror}}$ that it has issued itself via the message oracle $\mathcal{O}_{\mathsf{msg}}$. But $\mathcal{A}$ still has the freedom to query $\mathcal{O}_{\mathsf{msg}}$ multiple times and submit the answer it likes most to $\mathcal{O}_{\mathsf{ror}}$.

In the class-hiding definition from [HS14], the oracle $\mathcal{O}_{\mathsf{ror}}$ returned message-signature $M[b]$ on the first call, which made it trivially for the adversary to distinguish the two modes of operation $b = 0$ and $b = 1$ by just checking whether it obtained a signed version his submitted message m or some other message r. Thus, no SPS-EQ could fulfill the class-hiding definition from [HS14], rendering it entirely useless. So in the above corrected version, the real-or-random oracle $\mathcal{O}_{\mathsf{ror}}$ does not return any response when first called but just records the message it was queried for.

In [FHS19], Fuchsbauer et al. gave a distinct class-hiding notion which does not involve signing but only requires the adversary to tell whether two given messages are related via the underlying equivalence relation or not. The following class-hiding definition for SPS-EQ is based on [FHS19].

Definition 4.23 (**SPS-EQ (message) class-hiding**) *Let $\lambda \in \mathbb{N}$ be the security parameter, $l > 1$ the vector length, G a group, Σ_{SPSEQ} an SPS-EQ with message space $\mathcal{M} := G^l$ over some equivalence relation R. We define the following security game for and adversary $\mathcal{A}$ and a challenger $\mathcal{C}$:*

$\mathsf{Exp}^{\mathsf{spseq\text{-}ch}}_{\mathcal{A},\Sigma_{\mathsf{SPSEQ}}}(\lambda)$

1: $\mathsf{pp} \leftarrow\$ \mathsf{SPSEQ.PGen}(\lambda)$
2: $m \leftarrow\$ \mathcal{M}$
3: $b \leftarrow\$ \{0,1\}$
4: $m_0 \leftarrow\$ [m]_R$
5: $m_1 \leftarrow\$ \mathcal{M}$
6: $b' \leftarrow\$ \mathcal{A}(\mathsf{pp}, m, m_b)$
7: **return** 1 **if** $b = b'$

We define the advantage of $\mathcal{A}$ in the above security game as

$$\mathsf{Adv}^{\mathsf{spseq\text{-}ch}}_{\mathcal{A},\Sigma_{\mathsf{SPSEQ}}}(\lambda) = |Pr[\mathsf{Exp}^{\mathsf{spseq\text{-}ch}}_{\mathcal{A},\Sigma_{\mathsf{SPSEQ}}}(\lambda) = 1] - \frac{1}{2}|$$

Σ_{SPSEQ} is (message) class-hiding *if for all ppt adversaries $\mathcal{A}$ $\mathsf{Adv}^{\mathsf{spseq\text{-}ch}}_{\mathcal{A},\Sigma_{\mathsf{SPSEQ}}}(\lambda)$ is negligible.*

To conclude the class-hiding discussion for SPS-EQ, we formally prove that under rea-

sonable assumptions, real-or-random class-hiding is stronger than message class-hiding. We start with the proof that message class-hiding does not imply real-or-random class-hiding. For this, we do not use any further assumptions but simply construct an SPS-EQ that is message class-hiding but not real-or-random class-hiding.

Lemma 4.24 *Let $\lambda \in N$ be the security parameter, $l := l(\lambda)$ the length parameter which is a polynomial in λ. Furthermore let Σ_{SPSEQ} be a message class-hiding SPS-EQ with message space $\mathcal{M}$ and signature space $\mathcal{S}$. We construct an SPS-EQ Σ'_{SPSEQ} consisting of the same algorithms as Σ_{SPSEQ} except for $\mathsf{SPSEQ.ChgRep}$ which is changed so that for all $m \in \mathcal{M}$, $\sigma \in \mathcal{S}$, $\mathsf{SPSEQ.ChgRep}(m, \sigma, \cdot, \cdot)$ just outputs (m, σ).*

Then Σ'_{SPSEQ} is message class-hiding but not real-or-random class-hiding.

Proof. We see that for any adversary $\mathcal{A}$, the distribution of the input of $\mathcal{A}$ in the message class-hiding game $\mathsf{Exp}^{\mathsf{spseq\text{-}ch}}_{\mathcal{A},\Sigma_{\mathsf{SPSEQ}}}(\lambda)$ from Def. 4.23 is identical to the distribution of the input of $\mathcal{A}$ in $\mathsf{Exp}^{\mathsf{spseq\text{-}ch}}_{\mathcal{A},\Sigma'_{\mathsf{SPSEQ}}}(\lambda)$. This yields that

$$\mathsf{Adv}^{\mathsf{spseq\text{-}ch}}_{\mathcal{A},\Sigma_{\mathsf{SPSEQ}}}(\lambda) = \mathsf{Adv}^{\mathsf{spseq\text{-}ch}}_{\mathcal{A},\Sigma'_{\mathsf{SPSEQ}}}(\lambda)$$

and thus proves that Σ'_{SPSEQ} is message class-hiding according to Def. 4.23 since Σ_{SPSEQ} is.

What is left to prove is that Σ'_{SPSEQ} is not real-or-random class-hiding according to Def. 4.22. For this, we construct an adversary $\mathcal{A}$ that has non-negligible advantage in the real-or-random class-hiding game for Σ'_{SPSEQ}. On input $\mathsf{pp} \leftarrow\!\!\$\ \mathsf{SPSEQ.PGen}(\lambda), l$, $\mathcal{A}$ acts as follows:

1. $\mathcal{A}$ creates a key pair $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{SPSEQ.KGen}(\mathsf{pp}, l)$ and outputs it together with state information state.

2. When called on input state information state and key pair $(\mathsf{pk}, \mathsf{sk})$, $\mathcal{A}$ queries its message oracle $\mathcal{O}_{\mathsf{msg}}(l)$ for a message, eventually obtaining $m \leftarrow\!\!\$\ \mathcal{M}$.

3. $\mathcal{A}$ then signs m by computing $\sigma \leftarrow\!\!\$\ \mathsf{SPSEQ.Sign}(\mathsf{sk}, m)$

4. $\mathcal{A}$ subsequently submits (m, σ) to $\mathcal{O}_{\mathsf{ror}}(\mathsf{sk}, \mathsf{pk}, b, \cdot, \cdot)$, eventually obtaining (m', σ').

5. $\mathcal{A}$ checks whether $m = m'$. If so, it outputs $b' = 0$, else it outputs $b' = 1$.

$\mathcal{A}$ obviously is a ppt adversary. Let $b = 0$. Then, for the call (m, σ) to the oracle $\mathcal{O}_{\mathsf{ror}}$, we have

$$\mathsf{SPSEQ.ChgRep}(m, \sigma, \cdot, \cdot) \rightarrow (m', \sigma') = (m, \sigma)$$

by definition of Σ'_{SPSEQ} which concludes $\Pr[b' = 0 \mid b = 0] = 1$. Now let $b = 1$. Then, for the call (m, σ) to the oracle $\mathcal{O}_{\mathsf{ror}}$, we have

$$\mathsf{SPSEQ.ChgRep}(m, \sigma, \cdot, \cdot) \rightarrow (r, \sigma')$$

with $r \leftarrow\!\!\$\ \mathcal{M}$, $\sigma' \leftarrow\!\!\$\ \mathsf{SPSEQ.Sign}(\mathsf{sk}, r)$. We obviously have

$$\Pr[b' = 1 \mid b = 1] = 1 - \Pr[r = m] = 1 - \frac{1}{p^l}$$

since m is fixed at the time that r is drawn.

All in all, we get

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{spseq\text{-}ror\text{-}ch}}_{\mathcal{A},\Sigma_{\mathsf{SPSEQ}}}(\lambda) &= |\Pr[\mathsf{Exp}^{\mathsf{spseq\text{-}ror\text{-}ch}}_{\mathcal{A},\Sigma_{\mathsf{SPSEQ}}}(\lambda) = 1] - \frac{1}{2}| \\
&= |\Pr[b' = b] - \frac{1}{2}| \\
&= |\Pr[b' = 0 \mid b = 0] \cdot \frac{1}{2} + \Pr[b' = 1 \mid b = 1] \cdot \frac{1}{2} - \frac{1}{2}| \\
&= |1 \cdot \frac{1}{2} + (1 - \frac{1}{p^l}) \cdot \frac{1}{2} - \frac{1}{2}| \\
&= (1 - \frac{1}{p^l}) \cdot \frac{1}{2} \\
&= \frac{1}{2} - \frac{1}{2} \cdot \frac{1}{p^l}
\end{aligned}$$

which is not negligible since $\frac{1}{2} \cdot \frac{1}{p^l}$ is negligible because p^l is a super-polynomial expression in λ. This proves that Σ'_{SPSEQ} is not real-or-random class-hiding according to Def. 4.22. □

We see that the above proof makes use of the fact that message class-hiding (Def. 4.23) does not make any statement about the output distribution of the SPSEQ.ChgRep algorithm while this distribution is of crucial importance for the real-or-random class-hiding game. An instantiation of a message class-hiding SPS-EQ can be found in [FHS19]. So Lem. 4.24 indeed proves that there are SPS-EQ that are message class-hiding but not real-or-random class-hiding.

Next, we prove that real-or-random class-hiding does imply message class-hiding if the SPS-EQ perfectly adapts signatures and its message relation R divides the message space $\mathcal{M}$ into classes of equal size.

Lemma 4.25 *Let $\lambda \in N$ be the security parameter, $l := l(\lambda)$ the length parameter which is a polynomial in λ. Furthermore let Σ_{SPSEQ} be a real-or-random class-hiding SPS-EQ with message space $\mathcal{M}$ over message equivalence relation R with the following properties:*

(i) *Σ_{SPSEQ} is real-or-random class-hiding (Def. 4.23).*

(ii) *For $(m', \cdot) \leftarrow\!\!\$\ \mathsf{SPSEQ.ChgRep}(m, \cdot, \cdot, \cdot)$, we have that m' is identically distributed to a uniformly random message from $[m]_R$.*

(iii) *$\forall m, m' \in \mathcal{M} : |[m]_R| = |[m']_R|$, i.e. all equivalence classes of R are of the same size.*

Then Σ_{SPSEQ} is message class-hiding.

Proof. We use a standard reduction approach to prove this, i.e. assume that there exists a ppt adversary $\mathcal{A}$ with non-negligible advantage in the message class-hiding game from Def. 4.23. We construct an adversary $\mathcal{R}$ from it that has non-negligible advantage in the real-or-random class-hiding game from Def. 4.22.

On input $\mathsf{pp} \leftarrow\!\!\$\ \mathsf{SPSEQ.PGen}(\lambda), l \in \mathbb{N}$, $\mathcal{R}$ acts as follows:

1. $\mathcal{R}$ generates and outputs $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{SPSEQ.KGen}(\mathsf{pp}, l)$ together with state information state.
2. When eventually called on state information state and key pair $(\mathsf{pk}, \mathsf{sk})$, $\mathcal{R}$ queries the message oracle $\mathcal{O}_{\mathsf{msg}}(l)$ and obtains $m \leftarrow\!\!\$\ \mathcal{M}$.
3. $\mathcal{R}$ signs m by computing $\sigma \leftarrow\!\!\$\ \mathsf{SPSEQ.Sign}(\mathsf{sk}, m)$.
4. $\mathcal{R}$ queries $\mathcal{O}_{\mathsf{ror}}(\mathsf{pk}, \mathsf{sk}, b, \cdot, \cdot)$ for (m, σ), which triggers the recording of $M[b]$ on the oracle side.
5. $\mathcal{R}$ queries $\mathcal{O}_{\mathsf{ror}}(\mathsf{pk}, \mathsf{sk}, b, \cdot, \cdot)$ for (m, σ), this time obtaining some (m', σ') eventually.
6. $\mathcal{R}$ computes $b' \leftarrow\!\!\$\ \mathcal{A}(\mathsf{pp}, m, m')$ and outputs b'.

$\mathcal{R}$ obviously is a ppt since $\mathcal{A}$ is a ppt. Let $b_{\mathcal{R}}, b_{\mathcal{A}}$ be the hidden bits in the game that the respective adversary takes part in. First, let $b_{\mathcal{R}} = 0$. We see that, in this case, we have $(m', \sigma') \leftarrow\!\!\$\ \mathsf{SPSEQ.ChgRep}(m, \cdot, \cdot, \cdot)$, so by prerequisite, we have $m' \leftarrow\!\!\$\ [m]_R$. So in case $b_{\mathcal{R}} = 0$, $\mathcal{R}$ simulates the message class-hiding game for Σ_{SPSEQ} from Def. 4.23 with $b_{\mathcal{A}} = 0$ for $\mathcal{A}$. Now, let $b_{\mathcal{R}} = 1$. In this case, we have

$$(m', \sigma') \leftarrow\!\!\$\ \mathsf{SPSEQ.ChgRep}(r, \sigma, \alpha, \mathsf{pk})$$

with $r \leftarrow\!\!\$\ \mathcal{M}$, $\alpha \leftarrow\!\!\$\ \mathsf{SPSEQ.ConvSigGen}(\mathsf{pp})$, $\sigma \leftarrow\!\!\$\ \mathsf{SPSEQ.Sign}(\mathsf{sk}, r)$.

Let $\tilde{m} \in \mathcal{M}$ be some arbitrary fixed message. We see that

$$\begin{aligned}
\Pr[m' = \tilde{m}] &= \Pr[m' = \tilde{m} \mid r \in [\tilde{m}]_R] \cdot \Pr[r \in [\tilde{m}]_R] \\
&= \frac{1}{\frac{|\mathcal{M}|}{|\mathcal{M}/R|}} \cdot \frac{\frac{|\mathcal{M}|}{|\mathcal{M}/R|}}{|\mathcal{M}|} \\
&= \frac{1}{\frac{|\mathcal{M}|}{|\mathcal{M}/R|}} \cdot \frac{1}{|\mathcal{M}/R|} \\
&= \frac{1}{|\mathcal{M}|}
\end{aligned}$$

where we exploit that the size of each equivalence class of R is exactly

$$\frac{|\mathcal{M}|}{|\mathcal{M}/R|}$$

since all classes have the same size by prerequisite. Because $\tilde{m}$ was arbitrary, we thus get that in the case $b_{\mathcal{R}} = 1$, $m' \in \mathcal{M}$ is an uniformly random message. So, in case

that $b_{\mathcal{R}} = 1$, $\mathcal{R}$ simulates the message class-hiding game for Σ_{SPSEQ} from Def. 4.23 with $b_{\mathcal{A}} = 1$ for $\mathcal{A}$.

Let $b'_{\mathcal{R}}, b'_{\mathcal{A}}$ be the bits output by the respective adversaries. We get

$$\begin{aligned}\Pr[b'_{\mathcal{R}} = b_{\mathcal{R}}] &= \Pr[b'_{\mathcal{R}} = 0 \mid b_{\mathcal{R}} = 0] \cdot \Pr[b_{\mathcal{R}} = 0] + \Pr[b'_{\mathcal{R}} = 1 \mid b_{\mathcal{R}} = 1] \cdot \Pr[b_{\mathcal{R}} = 1] \\ &= \Pr[b'_{\mathcal{A}} = 0 \mid b_{\mathcal{R}} = 0] \cdot \Pr[b_{\mathcal{R}} = 0] + \Pr[b'_{\mathcal{A}} = 1 \mid b_{\mathcal{R}} = 1] \cdot \Pr[b_{\mathcal{R}} = 1] \\ &= \Pr[b'_{\mathcal{A}} = 0 \mid b_{\mathcal{A}} = 0] \cdot \Pr[b_{\mathcal{A}} = 0] + \Pr[b'_{\mathcal{A}} = 1 \mid b_{\mathcal{A}} = 1] \cdot \Pr[b_{\mathcal{A}} = 1] \\ &= \Pr[b'_{\mathcal{A}} = b_{\mathcal{A}}]\end{aligned}$$

which yields

$$\mathsf{Adv}^{\mathsf{spseq\text{-}ror\text{-}ch}}_{\mathcal{R},\Sigma_{\mathsf{SPSEQ}}}(\lambda) = \mathsf{Adv}^{\mathsf{spseq\text{-}ch}}_{\mathcal{A},\Sigma_{\mathsf{SPSEQ}}}(\lambda)$$

So since Σ_{SPSEQ} is message class-hiding according to Def. 4.23, Σ_{SPSEQ} is real-or-random class-hiding according to Def. 4.22. □

Perfect adaptation of signatures The difference between the distribution of fresh signatures generated using SPSEQ.Sign and randomized signatures generated using SPSEQ.ChgRep is yet an open question. This motivates the definition of the following additional security property for SPS-EQ:

Definition 4.26 (perfect adaptation of signatures for SPS-EQ) *Let $\lambda \in \mathbb{N}$ be the security parameter, $l > 1$ the vector length, G a group, Σ_{SPSEQ} an SPS-EQ with message space $\mathcal{M} := G^l$ over some equivalence relation R.* Σ_{SPSEQ} perfectly adapts signatures (w.r.t. the message space) *if for all*

$$(\mathsf{pk}, \mathsf{sk}, m, \sigma, \alpha) \in E \times H \times \mathcal{M} \times \mathcal{S} \times [\mathsf{SPSEQ.ConvSigGen}(\mathsf{pp})]$$

with $\mathsf{SPSEQ.VKey}(\mathsf{pk}, \mathsf{sk}) = 1$, $\mathsf{SPSEQ.Vfy}(\mathsf{pk}, m, \sigma) = 1$, $\mathsf{pp} \leftarrow\!\!\$\ \mathsf{SPSEQ.PGen}(\lambda)$, $\alpha \leftarrow\!\!\$\ \mathsf{SPSEQ.ConvSigGen}(\mathsf{pp})$ *it holds that*

$$v_1 := (m', \mathsf{SPSEQ.Sign}(\mathsf{sk}, m')), v_2 := \mathsf{SPSEQ.ChgRep}(m, \sigma, \alpha, \mathsf{pk})$$

are identically distributed with $(m', \cdot) \leftarrow\!\!\$\ \mathsf{SPSEQ.ChgRep}(m, \sigma, \alpha, \mathsf{pk})$.

So if an SPS-EQ perfectly adapts signatures, this means that fresh and randomized signatures on any given message are indistinguishable, even for an adversary with unlimited computational power. Note that this is fundamentally different from class-hiding which states that the uniform distribution in $\mathcal{M}$ is notably different from all uniform distributions in some class $[m]_R \in \mathcal{M}/R$, not making any statement about signatures but considering messages from more than one class. The above definition is a more general version of the one found in [BHKS18] since it does not prescribe that the output message m' of SPSEQ.ChgRep is computed deterministically from the input randomness α but is just any random representative from the input class $[m]_R$. We made this change since in this work, we cover general SPS-EQ over arbitrary equivalence relations while [FHS19] and [BHKS18] focused on SPS-EQ over a specific equivalence relation.

In [FHS19], Fuchsbauer et al. provide an alternative variant of perfect signature

adaptation that takes arbitrary maliciously generated public keys into account. We adapt it in the same way we did for obtaining Def. 4.26 from [BHKS18], resulting in the following definition which is more general than Def. 4.26:

Definition 4.27 (**perfect adaptation of signatures under malicious public keys**) *Let $\lambda \in \mathbb{N}$ be the security parameter, $l > 1$ the vector length, G a group, Σ_{SPSEQ} an SPS-EQ with message space $\mathcal{M} := G^l$ over some equivalence relation R.* Σ_{SPSEQ} perfectly adapts signatures (w.r.t. the message space) under malicious public keys *if for all*

$$(\mathsf{pk}, m, \sigma, \alpha) \in E \times \mathcal{M} \times \mathcal{S} \times [\mathsf{SPSEQ.ConvSigGen}(\mathsf{pp})]$$

with $\mathsf{SPSEQ.Vfy}(\mathsf{pk}, m, \sigma) = 1$, $\mathsf{pp} \leftarrow\!\!\$\ \mathsf{SPSEQ.PGen}(\lambda)$, $\alpha \leftarrow\!\!\$\ \mathsf{SPSEQ.ConvSigGen}(\mathsf{pp})$ *it holds that*

$$v_1 := (m', \sigma'), v_2 := \mathsf{SPSEQ.ChgRep}(m, \sigma, \alpha, \mathsf{pk})$$

are identically distributed with $(m', \cdot) \leftarrow\!\!\$\ \mathsf{SPSEQ.ChgRep}(m, \sigma, \alpha, \mathsf{pk})$ *and σ' being identically distributed to a uniformly random signature in the set of valid signatures for m', i.e.*

$$\{\hat{\sigma} \mid \mathsf{SPSEQ.Vfy}(\mathsf{pk}, m', \hat{\sigma}) = 1\}$$

In the above definition, we consider arbitrary public keys, including those that no corresponding secret key is known for. So we require the distribution of the output signature σ' of $\mathsf{SPSEQ.ChgRep}$ must be indistinguishable from the uniformly random distribution on the set of valid signatures for m' instead of the distribution of $\mathsf{SPSEQ.Sign}(\mathsf{sk}, m')$. This is because the random variable $\mathsf{SPSEQ.Sign}(\mathsf{sk}, m')$ is not defined for public keys pk for which no corresponding secret key sk exists.

4.3 Signatures with flexible public keys

In this chapter, we will deal with *flexible public key signatures* (alternatively called *signatures with flexible public key*, in short *SFPK*) which were originally introduced by Backes et al. in [BHKS18]. An SFPK has a public key space that is divided into the equivalence classes of an equivalence relation R and allows the owner of a key pair $(\mathsf{pk}, \mathsf{sk})$ to efficiently randomize it to a new pair $(\mathsf{pk}', \mathsf{sk}')$ where pk and pk' are related via R. Thus, SFPK can be seen as the complementary primitive to SPS-EQ (see Sect. 4.2), which have an analogous equivalence relation on the message space. The possibility to randomize a public key within its equivalence class requires a relaxation of the winning condition of the signature unforgeability game from Def. 3.19, more precisely the adversary also wins if it forges a signature valid under any key pk' from the same class as the challenge key pk. Furthermore, without access to a *class trapdoor* τ_{pk} it should not be possible to efficiently tell whether some public key pk' is in the same class as a given key pk. This property is called *class-hiding*. In the following, we will formally introduce SFPK and the security requirements sketched above. If not otherwise stated, our definitional framework hereby is based on [BHKS18] where SFPK were first introduced. We start with the basic syntax of a flexible public key signature.

Definition 4.28 (**SFPK syntax**) *A* flexible public key signature (SFPK) *over equivalence relation R with public key space E, secret key space H, message space $\mathcal{M}$, signature space $\mathcal{S}$ and trapdoor space T is a tuple Σ_{SFPK} of ppt algorithms as follows:*

$\mathsf{SFPK.PGen}(\lambda)$: *probablistic parameter generation algorithm, takes as input the security parameter $\lambda \in \mathbb{N}$ and outputs public parameters* pp

$\mathsf{SFPK.KGen}(\mathsf{pp}, \omega)$: *deterministic key generation algorithm, takes as input public parameters* pp *and* key generation randomness $\omega \in \{0,1\}^{\mathsf{poly}(\lambda)}$ *and outputs a key pair* $(\mathsf{pk}, \mathsf{sk})$*, consisting of a public key* $\mathsf{pk} \in E$ *and a secret key* $\mathsf{sk} \in H$

$\mathsf{SFPK.TKGen}(\mathsf{pp}, \omega)$: *deterministic trapdoor key generation algorithm, takes as input public parameters* pp *and* key generation randomness $\omega \in \{0,1\}^{\mathsf{poly}(\lambda)}$ *and outputs a key pair* $(\mathsf{pk}, \mathsf{sk})$ *(consisting of a public key* $\mathsf{pk} \in E$ *and a secret key* $\mathsf{sk} \in H$*) together with a* class trapdoor $\tau \in T$

$\mathsf{SFPK.Sign}(\mathsf{sk}, m)$: *probablistic signing algorithm, takes as input a secret key* $\mathsf{sk} \in H$ *and a message $m \in \mathcal{M}$ and outputs a signature $\sigma \in \mathcal{S}$*

$\mathsf{SFPK.ChkRep}(\tau, \mathsf{pk})$: *deterministic representative-check algorithm, takes as input a class trapdoor $\tau \in T$ and a public key* $\mathsf{pk} \in E$ *and outputs a bit $b \in \{0,1\}$*

$\mathsf{SFPK.KeyConvGen}(\mathsf{pp})$ *probablistic key change randomness generation algorithm, takes as input public parameters* pp *and outputs key conversion randomness r*

$\mathsf{SFPK.ChgPK}(\mathsf{pk}, r)$: *deterministic public key adaption algorithm, takes as input a public key* $\mathsf{pk} \in E$ *and randomness r and outputs a public key* $\mathsf{pk}' \in E$

$\mathsf{SFPK.ChgSK}(\mathsf{sk}, r)$: *deterministic secret key adaption algorithm, takes as input a secret key* $\mathsf{sk} \in H$ *and randomness r and outputs a secret key* $\mathsf{sk}' \in H$

$\mathsf{SFPK.Vfy}(\mathsf{pk}, m, \sigma)$: *deterministic verification algorithm, takes as input a public key* $\mathsf{pk} \in E$*, a message $m \in \mathcal{M}$ and a signature $\sigma \in \mathcal{S}$ and outputs a bit $b \in \{0,1\}$*

$\mathsf{SFPK.VKey}(\mathsf{pk}, \mathsf{sk})$: *deterministic key pair verification algorithm, takes as input a public key* $\mathsf{pk} \in E$ *and a secret key* $\mathsf{sk} \in H$ *and outputs a bit $b \in \{0,1\}$*

If not explicitly stated otherwise, we always assume that the algorithms of an SFPK are named as in Def. 4.28. The set of values r that can be input to the $\mathsf{SFPK.ChgPK}$ and $\mathsf{SFPK.ChgSK}$ algorithms as randomness is implicitly defined by the particular SFPK and contains the support of $\mathsf{SFPK.KeyConvGen}$ as a subset. Note that the key generation algorithms $\mathsf{SFPK.KGen}, \mathsf{SFPK.TKGen}$ are deterministic, with random coins that are passed as parameters. This allows to hand the adversary in SFPK security games the randomness that was used to generate keys. For example, in constructions where there are elements of some cyclic group contained in a key, this models the case that the adversary knows the respective discrete logarithms of these elements. Next, we define the correctness constraints that an SFPK has to fulfill.

Definition 4.29 (**correct SFPK**) *Let Σ_{SFPK} be an SFPK over equivalence relation R with public key space E, secret key space H, message space $\mathcal{M}$, signature space $\mathcal{S}$ and trapdoor space T. Σ_{SFPK} is* correct *if the following conditions are fulfilled:*

(i) $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{SFPK.KGen}(\lambda, \omega)$ *and* $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{SFPK.TKGen}(\lambda, \omega)$ *(which are random variables in the key generation randomness ω) are identically distributed for all $\lambda \in \mathbb{N}$*

(ii) For all key pairs $(\mathsf{sk}, \mathsf{pk}) \leftarrow\!\!\$\ \mathsf{KGen}(\lambda, \omega)$, $\omega \leftarrow\!\!\$\ \{0,1\}^{\mathsf{poly}(\lambda)}$, *messages $m \in \mathcal{M}$, we have*

a) $\Pr[\mathsf{SFPK.Vfy}(\mathsf{pk}, m, \mathsf{SFPK.Sign}(\mathsf{sk}, m)) = 1] = 1$

b) For $r \leftarrow\!\!\$\ \mathsf{SFPK.KeyConvGen}(\mathsf{pp})$, $\mathsf{pk}' \leftarrow \mathsf{SFPK.ChgPK}(\mathsf{pk}, r)$ *and* $\mathsf{sk}' \leftarrow \mathsf{SFPK.ChgSK}(\mathsf{sk}, r)$, *we have*

i. $\Pr[\mathsf{SFPK.Vfy}(\mathsf{pk}', m, \mathsf{SFPK.Sign}(\mathsf{sk}', m)) = 1] = 1$

ii. $\Pr[\mathsf{SFPK.VKey}(\mathsf{pk}', \mathsf{sk}') = 1] = 1$

(iii) For all $\lambda \in \mathbb{N}$, $\omega \in \{0,1\}^{\mathsf{poly}(\lambda)}$, $(\mathsf{pk}, \mathsf{sk}, \tau) \leftarrow\!\!\$\ \mathsf{SFPK.TKGen}(\lambda, \omega)$ *and all public keys* $\mathsf{pk}' \in E$, *we have*

$$\mathsf{SFPK.ChkRep}(\tau, \mathsf{pk}') = 1 \Leftrightarrow \mathsf{pk}' \in [\mathsf{pk}]_R$$

(iv) For all public parameters $\mathsf{pp} \leftarrow\!\!\$\ \mathsf{SFPK.PGen}(\lambda)$, *public keys* $\mathsf{pk} \in E$ *and random coins* $r \leftarrow\!\!\$\ \mathsf{SFPK.KeyConvGen}(\mathsf{pp})$, *we have*

$$\mathsf{SFPK.ChgPK}(\mathsf{pk}, r) =: \mathsf{pk}' \in [\mathsf{pk}]_R$$

So for a correct SFPK, key pairs output by SFPK.KGen and SFPK.TKGen are perfectly indistinguishable, due to them being identically distributed. This yields that the second requirement also holds for key pairs generated using SFPK.TKGen. It requires that signatures generated in an honest way using a valid key pair are always valid, furthermore a key pair whose keys are both adapted using the same randomness r stays valid. The third requirement states that SFPK.ChkRep perfectly tells whether two keys are related while the fourth one requires that SFPK.ChgPK outputs another representative of the input class $[\mathsf{pk}]_R$.

Unforgeability With syntax and correctness defined, we next move to the security definitions for SFPK which comprise a modified version of standard signature unforgeability (from Def. 3.19) as well as the class hiding notion already mentioned in the introduction of this subsection. We start with unforgeability.

Definition 4.30 (**SFPK unforgeability**) *Let $\lambda \in \mathbb{N}$ be the security parameter, Σ_{SFPK} be an SFPK over equivalence relation R. We define the following security game for and adversary $\mathcal{A}$ and a challenger $\mathcal{C}$:*

$\mathsf{Exp}^{\mathsf{sfpk\text{-}euf}}_{\mathcal{A},\Sigma_{\mathsf{SFPK}}}(\lambda)$

1 : $\mathsf{pp} \leftarrow\!\$\ \mathsf{SFPK.PGen}(\lambda)$
2 : $\omega \leftarrow\!\$\ \{0,1\}^{\mathsf{poly}(\lambda)}$
3 : $(\mathsf{pk}, \mathsf{sk}, \tau) \leftarrow\!\$\ \mathsf{SFPK.TKGen}(\mathsf{pp}, \omega)$
4 : $Q := \emptyset$
5 : $(\mathsf{pk}^*, m^*, \sigma^*) \leftarrow\!\$\ \mathcal{A}^{\mathsf{Sign}(\mathsf{sk},\cdot),\mathsf{Sign}(\mathsf{sk},\cdot,\cdot)}(\mathsf{pp}, \mathsf{pk}, \tau)$
6 : **return** 1 **if** $(m^*, \cdot) \notin Q$
7 : $\quad \wedge\ \mathsf{SFPK.ChkRep}(\tau, \mathsf{pk}^*) = 1$
8 : $\quad \wedge\ \mathsf{SFPK.Vfy}(\mathsf{pk}^*, m^*, \sigma^*) = 1$
9 : **else return** 0

with oracles

$\mathsf{Sign}(\mathsf{sk}, m)$

1 : $\sigma \leftarrow\!\$\ \mathsf{SFPK.Sign}(\mathsf{sk}, m)$
2 : $Q := Q \cup \{(m, \sigma)\}$
3 : **return** σ

$\mathsf{Sign}(\mathsf{sk}, m, r)$

1 : $\mathsf{sk}' := \mathsf{SFPK.ChgSK}(\mathsf{sk}, r)$
2 : $\sigma \leftarrow\!\$\ \mathsf{SFPK.Sign}(\mathsf{sk}', m)$
3 : $Q := Q \cup \{(m, \sigma)\}$
4 : **return** σ

We define the advantage of $\mathcal{A}$ in the above security game as

$$\mathsf{Adv}^{\mathsf{sfpk\text{-}euf}}_{\mathcal{A},\Sigma_{\mathsf{SFPK}}}(\lambda) = \Pr[\mathsf{Exp}^{\mathsf{sfpk\text{-}euf}}_{\mathcal{A},\Sigma_{\mathsf{SFPK}}}(\lambda) = 1]$$

Σ_{SFPK} *is* existentially unforgeable under chosen-message attacks (EUF-CMA) *if for all ppt adversaries $\mathcal{A}$ $\mathsf{Adv}^{\mathsf{sfpk\text{-}euf}}_{\mathcal{A},\Sigma_{\mathsf{SFPK}}}(\lambda)$ is negligible.*

So intuitively, it should be hard to efficiently forge a valid signature for a new message under a public key that is related to the challenge public key when being able to see signatures created using any key related to the challenge secret key. Analogously to regular digital signature schemes, we can define a stronger unforgeability notion by weakening the winning condition, more precisely we also allow the adversary to output a distinct signature for an already-signed message.

Definition 4.31 (SFPK sEUF-CMA) *Let $\lambda \in \mathbb{N}$ be the security parameter, Σ_{SFPK} be an SFPK over equivalence relation R. We define the security game $\mathsf{Exp}^{\mathsf{sfpk\text{-}seuf}}_{\mathcal{A},\Sigma_{\mathsf{SFPK}}}(\lambda)$ between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$ by replacing line 5 in the pseudocode of Def. 4.30 by*

$$\textbf{\textit{return}}\ 1\ \textbf{\textit{if}}\ (m^*, \sigma^*) \notin Q$$

We define the advantage of $\mathcal{A}$ in the above security game as

$$\mathsf{Adv}^{\mathsf{sfpk\text{-}seuf}}_{\mathcal{A},\Sigma_{\mathsf{SFPK}}}(\lambda) = \Pr[\mathsf{Exp}^{\mathsf{sfpk\text{-}seuf}}_{\mathcal{A},\Sigma_{\mathsf{SFPK}}}(\lambda) = 1]$$

Σ_{SFPK} *is* strongly existentially unforgeable under chosen-message attacks (sEUF-CMA) *if for all ppt adversaries $\mathcal{A}$ $\mathsf{Adv}^{\mathsf{sfpk\text{-}seuf}}_{\mathcal{A},\Sigma_{\mathsf{SFPK}}}(\lambda)$ is negligible.*

We next formulate a necessary condition for unforgeability of SFPK whose key generation algorithms output uniformly random public keys. In particular, such SFPK need to have "small" equivalence classes.

Lemma 4.32 (**small classes lemma**) *Let Σ_{SFPK} be a SFPK with public key space E which has the property that for all $\lambda \in \mathbb{N}$, pk in $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{SFPK.KGen}(\lambda, \omega)$ is a uniformly random element from E if $\omega \leftarrow\!\!\$\ \{0,1\}^{\mathsf{poly}(\lambda)}$ is drawn uniformly at random. Then it holds that*

$$\begin{aligned}&\Sigma_{\mathsf{SFPK}} \textit{ EUF-CMA secure}\\ \Rightarrow&(\forall [\mathsf{pk}]_R \in E/R : \frac{|[\mathsf{pk}]_R|}{|E|} = \mu_{\mathsf{pk}}(\lambda) \textit{ for a negligible } \mu_{\mathsf{pk}} : \mathbb{N} \to \mathbb{R})\end{aligned}$$

Proof. We prove the theorem by contraposition, i.e. we assume that

$$\exists [\mathsf{pk}^*]_R \in E/R : \frac{|[\mathsf{pk}^*]_R|}{|E|} = \eta_{\mathsf{pk}^*}(\lambda) \text{ for a non-negligible } \eta_{\mathsf{pk}^*} : \mathbb{N} \to \mathbb{R}$$

and prove that Σ_{SFPK} is not unforgeable by constructing an adversary $\mathcal{A}$ that has non-negligible advantage $\mathsf{Adv}^{\mathsf{sfpk\text{-}euf}}_{\mathcal{A},\Sigma_{\mathsf{SFPK}}}(\lambda)$. Upon input a public key pk and a trapdoor $\tau \in T$ (where T is the trapdoor space of Σ_{SFPK}), $\mathcal{A}$ behaves as follows:

1. $\mathcal{A}$ draws random coins $\omega \leftarrow\!\!\$\ \{0,1\}^{\mathsf{poly}(\lambda)}$ uniformly at random
2. $\mathcal{A}$ generates a fresh key pair $(\mathsf{pk}', \mathsf{sk}') \leftarrow\!\!\$\ \mathsf{SFPK.KGen}(\lambda, \omega)$
3. $\mathcal{A}$ chooses any message $m \in \mathcal{M}$
4. $\mathcal{A}$ computes and outputs $\sigma \leftarrow\!\!\$\ \mathsf{SFPK.Sign}(\mathsf{sk}', m)$

$\mathcal{A}$ obviously is a ppt since $\mathsf{SFPK.KGen}$ and $\mathsf{SFPK.Sign}$ are (because Σ_{SFPK} is an SFPK). σ obviously is a valid signature for m since Σ_{SFPK} is a correct SFPK according to Def. 4.29, furthermore $\mathcal{A}$ does not query the oracles for any messages. So we get

$$\begin{aligned}\mathsf{Adv}^{\mathsf{sfpk\text{-}euf}}_{\mathcal{A},\Sigma_{\mathsf{SFPK}}}(\lambda) =& \Pr[\mathsf{SFPK.ChkRep}(\tau, \mathsf{pk}') = 1]\\ \geq& \Pr[\mathsf{pk} \in [\mathsf{pk}^*]_R \wedge \mathsf{pk}' \in [\mathsf{pk}^*]_R]\\ =& \Pr[\mathsf{pk} \in [\mathsf{pk}^*]_R] \cdot \Pr[\mathsf{pk}' \in [\mathsf{pk}^*]_R]\\ =& \eta_{\mathsf{pk}^*}(\lambda) \cdot \eta_{\mathsf{pk}^*}(\lambda)\end{aligned}$$

(which proves the lemma), where the second to last inequality holds because pk and pk' where independently generated and the last equality holds because of the requirement that $\mathsf{SFPK.KGen}$ outputs uniformly random public keys. □

Class-hiding The second important security definition for SFPK is class-hiding. A variety of class-hiding notions has been introduced over the years, starting with what we call *find-original class-hiding* by Backes et al. [BHKS18]. Informally, this security notion

states that no adversary should be able to efficiently tell to which of two public keys pk_0 and pk_1 a given public key pk' is related, even when given the randomness that was used to generate the initial keys pk_0 and pk_1. The adversary can furthermore observe signatures under the secret key sk' corresponding to pk'. Note that the randomness gives the adversary more information about the public keys pk_0 and pk_1 than just their value itself since it also learns the corresponding secret keys sk_0 and sk_1 as well as some additional information about $\mathsf{pk}_0, \mathsf{pk}_1$. For example, this additional information could be discrete logarithms in a scenario where the public keys contain elements of a cyclic group.

Definition 4.33 (**SPFK find-original class-hiding**) *Let $\lambda \in \mathbb{N}$ be the security parameter, Σ_{SFPK} be an SFPK over equivalence relation R. We define the following security game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$:*

$\mathsf{Exp}^{\mathsf{sfpk\text{-}fo\text{-}ch}}_{\mathcal{A},\Sigma_{\mathsf{SFPK}}}(\lambda)$

1 : $\mathsf{pp} \leftarrow\!\$\ \mathsf{SFPK.PGen}(\lambda)$
2 : $\omega_0, \omega_1 \leftarrow\!\$\ \{0,1\}^{\mathsf{poly}(\lambda)}$
3 : **for** $i \in \{0,1\} : (\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow\!\$\ \mathsf{SFPK.KGen}(\mathsf{pp}, \omega_i)$
4 : $b \leftarrow\!\$\ \{0,1\}$
5 : $r \leftarrow\!\$\ \mathsf{SFPK.KeyConvGen}(\mathsf{pp})$
6 : $\mathsf{sk}' := \mathsf{SFPK.ChgSK}(\mathsf{sk}_b, r), \mathsf{pk}' := \mathsf{SFPK.ChgPK}(\mathsf{pk}_b, r)$
7 : $b' \leftarrow\!\$\ \mathcal{A}^{\mathsf{SFPK.Sign}(\mathsf{sk}', \cdot)}(\omega_0, \omega_1, \mathsf{pk}')$
8 : **return** 1 **if** $b = b'$**else return** 0

We define the advantage of $\mathcal{A}$ in the above security game as

$$\mathsf{Adv}^{\mathsf{sfpk\text{-}fo\text{-}ch}}_{\mathcal{A},\Sigma_{\mathsf{SFPK}}}(\lambda) := |\Pr[\mathsf{Exp}^{\mathsf{sfpk\text{-}fo\text{-}ch}}_{\mathcal{A},\Sigma_{\mathsf{SFPK}}}(\lambda) = 1] - \frac{1}{2}|$$

Σ_{SFPK} *is called* class-hiding *if for all ppt adversary $\mathcal{A}$* $\mathsf{Adv}^{\mathsf{sfpk\text{-}fo\text{-}ch}}_{\mathcal{A},\Sigma_{\mathsf{SFPK}}}(\lambda)$.

If not explicitly stated otherwise, we will use this find-original class-hiding definition for SFPK since Backes et al. used it when proving their SFPK constructions from [BHKS18] to be secure. If we would deviate to a different class-hiding notion, it might be unclear whether it is possible to instantiate it, i.e. how to build an SFPK that fulfills it.

Class-hiding discussion In the following, we introduce several other class-hiding notions and relate them to the above one. We begin with *real-or-random class-hiding* which is inspired by the public-key class-hiding notion for mercurial signatures which Crites et al. introduced in [CL19]. Intuitively, an adversary should decide whether a given public key was adapted from a known one or was freshly and independently generated.

Definition 4.34 (**SFPK real-or-random class-hiding**) *Let $\lambda \in \mathbb{N}$ be the security parameter, Σ_{SFPK} be an SFPK. We define the following security game for Σ_{SFPK} between*

an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$:

$\mathsf{Exp}^{\mathsf{sfpk\text{-}ror\text{-}ch}}_{\mathcal{A},\Sigma_{\mathsf{SFPK}}}(\lambda)$

1 : $\mathsf{pp} \leftarrow\!\$\ \mathsf{SFPK.PGen}(\lambda)$
2 : $\omega_1, \omega_2 \leftarrow\!\$\ \{0,1\}^{\mathsf{poly}(\lambda)}$
3 : $(\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow\!\$\ \mathsf{SFPK.KGen}(\mathsf{pp}, \omega_1)$
4 : $(\mathsf{pk}_2^0, \mathsf{sk}_2^0) \leftarrow\!\$\ \mathsf{SFPK.KGen}(\mathsf{pp}, \omega_2)$
5 : $r \leftarrow\!\$\ \mathsf{SFPK.KeyConvGen}(\mathsf{pp})$
6 : $(\mathsf{pk}_2^1, \mathsf{sk}_2^1) \leftarrow\!\$\ (\mathsf{SFPK.ChgPK}(\mathsf{pk}_1, r), \mathsf{SFPK.ChgSK}(\mathsf{sk}_1, r))$
7 : $b \leftarrow\!\$\ \{0,1\}$
8 : $b' \leftarrow\!\$\ \mathcal{A}^{\mathsf{Sign}(\mathsf{sk}_2^b, \cdot)}(\omega_1, \mathsf{pk}_2^b)$
9 : **return** $b = b'$

We define the advantage $\mathsf{Adv}^{\mathsf{sfpk\text{-}ror\text{-}ch}}_{\mathcal{A},\Sigma_{\mathsf{SFPK}}}(\lambda)$ *of* $\mathcal{A}$ *in the above security game as*

$$\mathsf{Adv}^{\mathsf{sfpk\text{-}ror\text{-}ch}}_{\mathcal{A},\Sigma_{\mathsf{SFPK}}}(\lambda) := |\Pr[\mathsf{Exp}^{\mathsf{sfpk\text{-}ror\text{-}ch}}_{\mathcal{A},\Sigma_{\mathsf{SFPK}}}(\lambda) = 1] - \frac{1}{2}|$$

Σ_{SFPK} *is* real-or-random class-hiding *if for all ppt adversaries* $\mathcal{A}$ $\mathsf{Adv}^{\mathsf{sfpk\text{-}ror\text{-}ch}}_{\mathcal{A},\Sigma_{\mathsf{SFPK}}}(\lambda)$ *is negligible.*

Next, we formally prove that find-original class-hiding is implied by real-or-random class-hiding.

Lemma 4.35 *Let $\lambda \in \mathbb{N}$ be the security parameter, Σ_{SFPK} be a real-or-random class-hiding SFPK with message space $\mathcal{M}$. Then Σ_{SFPK} is find-original class-hiding.*

Proof. Let $\mathcal{A}$ be a ppt adversary. The proof uses a sequence of games. We need to bridge the gap between the case $b = 0$ in the find-original class-hiding game $\mathsf{Exp}^{\mathsf{sfpk\text{-}fo\text{-}ch}}_{\mathcal{A},\Sigma_{\mathsf{SFPK}}}(\lambda)$ from Def. 4.33 and the case $b = 1$. We define a variant of the find-original class-hiding game from Def. 4.33 where the adversary $\mathcal{A}$ is given a third, independently generated public key pk_2 instead of an adapted version of either pk_0 or pk_1. An adversary noticing this change could distinguish fresh and adapted public keys and thus break real-or-random class-hiding.

To make this proof more convenient, we define the following distinguishing variant of the find-original class-hiding game for SFPK from Def. 4.33 (between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$):

$\mathsf{Exp}^{\mathsf{sfpk\text{-}fo\text{-}ch}\text{-}b}_{\mathcal{A},\Sigma_{\mathsf{SFPK}}}(\lambda)$

1 : $\mathsf{pp} \leftarrow\!\$\ \mathsf{SFPK.PGen}(\lambda)$
2 : $\omega_0, \omega_1 \leftarrow\!\$\ \{0,1\}^{\mathsf{poly}(\lambda)}$
3 : **for** $i \in \{0,1\}$: $(\mathsf{sk}_i, \mathsf{pk}_i) \leftarrow\!\$\ \mathsf{SFPK.KGen}(\mathsf{pp}, \omega_i)$
4 : $r \leftarrow\!\$\ \mathsf{SFPK.KeyConvGen}(\mathsf{pp})$
5 : $\mathsf{sk}' := \mathsf{SFPK.ChgSK}(\mathsf{sk}_b, r), \mathsf{pk}' := \mathsf{SFPK.ChgPK}(\mathsf{pk}_b, r)$
6 : **return** $b' \leftarrow\!\$\ \mathcal{A}^{\mathsf{SFPK.Sign}(\mathsf{sk}', m)}(\omega_0, \omega_1, \mathsf{pk}')$

The advantage of $\mathcal{A}$ in the above security game is defined as

$$\mathsf{Adv}^{\mathsf{sfpk\text{-}fo\text{-}ch}}_{\mathcal{A},\Sigma_{\mathsf{SFPK}}}(\lambda)^* := |\Pr[\mathsf{Exp}^{\mathsf{sfpk\text{-}fo\text{-}ch}\text{-}0}_{\mathcal{A},\Sigma_{\mathsf{SFPK}}}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{sfpk\text{-}fo\text{-}ch}\text{-}1}_{\mathcal{A},\Sigma_{\mathsf{SFPK}}}(\lambda) = 1]|$$

and say that Σ_{SFPK} is find-original class-hiding if for all ppt adversaries $\mathcal{A}$ the advantage $\mathsf{Adv}^{\mathsf{sfpk\text{-}fo\text{-}ch}}_{\mathcal{A},\Sigma_{\mathsf{SFPK}}}(\lambda)^*$ is negligible in λ. This find-original class-hiding definition is equivalent to the one from Def. 4.33 (with analogous argumentation to the proof of Theorem 2.10 in [BS20]). We also analogously define a distinguishing variant of the real-or-random class-hiding game for SFPK as follows:

$\mathsf{Exp}^{\mathsf{sfpk\text{-}ror\text{-}ch}\text{-}b}_{\mathcal{A},\Sigma_{\mathsf{SFPK}}}(\lambda)$

1 : $\mathsf{pp} \leftarrow\!\$\ \mathsf{SFPK.PGen}(\lambda)$
2 : $\omega_1, \omega_2 \leftarrow\!\$\ \{0,1\}^{\mathsf{poly}(\lambda)}$
3 : $(\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow\!\$\ \mathsf{SFPK.KGen}(\mathsf{pp}, \omega_1)$
4 : $(\mathsf{pk}_2^0, \mathsf{sk}_2^0) \leftarrow\!\$\ \mathsf{SFPK.KGen}(\mathsf{pp}, \omega_2)$
5 : $r \leftarrow\!\$\ \mathsf{SFPK.KeyConvGen}(\mathsf{pp})$
6 : $(\mathsf{pk}_2^1, \mathsf{sk}_2^1) \leftarrow\!\$\ (\mathsf{SFPK.ChgPK}(\mathsf{pk}_1, r), \mathsf{SFPK.ChgSK}(\mathsf{sk}_1, r))$
7 : $b' \leftarrow\!\$\ \mathcal{A}^{\mathsf{Sign}(\mathsf{sk}_2^b, \cdot)}(\omega_1, \mathsf{pk}_2^b)$
8 : **return** b'

We analogously define the advantage of the adversary in the above security game as

$$\mathsf{Adv}^{\mathsf{sfpk\text{-}ror\text{-}ch}}_{\mathcal{A},\Sigma_{\mathsf{SFPK}}}(\lambda)^* := |\Pr[\mathsf{Exp}^{\mathsf{sfpk\text{-}ror\text{-}ch}\text{-}0}_{\mathcal{A},\Sigma_{\mathsf{SFPK}}}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{sfpk\text{-}ror\text{-}ch}\text{-}1}_{\mathcal{A},\Sigma_{\mathsf{SFPK}}}(\lambda) = 1]|$$

and say that Σ_{SFPK} is real-or-random class-hiding if for all ppt adversaries $\mathcal{A}$ the advantage $\mathsf{Adv}^{\mathsf{sfpk\text{-}ror\text{-}ch}}_{\mathcal{A},\Sigma_{\mathsf{SFPK}}}(\lambda)^*$ is negligible in λ. We remark that the real-or-random class-hiding definition based on the above distingushing game is equivalent to the one from Def. 4.34 (this also uses an analogous argument to the one from Theorem 2.10 from [BS20]). We continue with the actual proof of Lem. 4.35 where we use the above distinguishing variants of find-original and real-or-random class-hiding for convenience. Let λ be the security parameter, $\mathcal{A}$ be a ppt adversary. We prove the lemma using a sequence of games as follows:

G_0 find-original class-hiding game $\mathsf{Exp}^{\mathsf{sfpk\text{-}fo\text{-}ch}\text{-}0}_{\mathcal{A},\Sigma_{\mathsf{SFPK}}}(\lambda)$

G_1 find-original class-hiding game $\mathsf{Exp}^{\mathsf{sfpk\text{-}fo\text{-}ch\text{-}0}}_{\mathcal{A},\Sigma_{\mathsf{SFPK}}}(\lambda)$, but we draw fresh randomness $\omega_2 \leftarrow\!\!\$\ \{0,1\}^{\mathsf{poly}(\lambda)}$ and generate a public key pk_2 as $(\mathsf{pk}_2, \mathsf{sk}_2) \leftarrow\!\!\$\ \mathsf{SFPK.KGen}(\mathsf{pp}, \omega_2)$ before line 6. In line 6, $\mathcal{A}$ is then called on input $\omega_0, \omega_1, \mathsf{pk}_2$. The signing oracle that the adversary can access during the game signs submitted messages with sk_2.

G_2 find-original class-hiding game $\mathsf{Exp}^{\mathsf{sfpk\text{-}fo\text{-}ch\text{-}1}}_{\mathcal{A},\Sigma_{\mathsf{SFPK}}}(\lambda)$

Let S_i denote the event that $\mathcal{A}$ outputs 1 in experiment i. So to prove that Σ_{SFPK} is find-original class-hiding according to Def. 4.33, we need to prove that $|\Pr[S_0] - \Pr[S_2]|$ is negligible. We get that there exist negligible functions $\epsilon_1, \epsilon_2, \epsilon_3$ such that

$$\begin{aligned}
|\Pr[S_0] - \Pr[S_2]| &= |\Pr[S_0] - \Pr[S_1] + \Pr[S_1] - \Pr[S_2]| \\
&\leq |\Pr[S_0] - \Pr[S_1]| + |\Pr[S_1] - \Pr[S_2]| \\
&\leq \epsilon_1(\lambda) + \epsilon_2(\lambda) \\
&\leq \epsilon_3(\lambda)
\end{aligned}$$

which proves that Σ_{SFPK} is find-original class-hiding according to Def. 4.33. We will explain the second upper bound in the following. We construct a ppt distinguisher $\mathcal{D}$ for the real-or-random class-hiding game from an adversary $\mathcal{A}$ distinguishing G_0 and G_1. On input $(\omega_1, \mathsf{pk}_2^b)$ distributed as in $\mathsf{Exp}^{\mathsf{sfpk\text{-}ror\text{-}ch\text{-}}b}_{\mathcal{A},\Sigma_{\mathsf{SFPK}}}(\lambda)$ for some $b \in \{0,1\}$, $\mathcal{D}$ acts as follows

1. $\mathcal{D}$ samples $\omega^* \leftarrow\!\!\$\ \{0,1\}^{\mathsf{poly}(\lambda)}$.
2. $\mathcal{D}$ runs $\mathcal{A}(\omega_1, \omega^*, \mathsf{pk}_2^b)$.
   - Whenever $\mathcal{A}$ queries for a signature on some message $m \in \mathcal{M}$, $\mathcal{D}$ queries its own oracle $\mathsf{SFPK.Sign}(\mathsf{sk}_2^b, \cdot)$ for m, eventually obtaining $\sigma \leftarrow\!\!\$\ \mathsf{SFPK.Sign}(\mathsf{sk}_2^b, m)$. $\mathcal{D}$ then relays σ to $\mathcal{A}$.
3. Eventually, $\mathcal{A}(\omega_1, \omega^*, \mathsf{pk}_2^b)$ returns b'. $\mathcal{D}$ then outputs b'.

In the following we call the input of an adversary together with the answers to its oracle queries the *view* of an adversary. We now argue that if $b = 0$, then the view of $\mathcal{A}$ is distributed as in G_1, if on the other hand we have $b = 1$, then the view of $\mathcal{A}$ is distributed as in G_0.

- Let $b = 0$. We have $\omega_1 \leftarrow\!\!\$\ \{0,1\}^{\mathsf{poly}(\lambda)}$ and $(\mathsf{pk}_2^b, \mathsf{sk}_2^b) := \mathsf{SFPK.KGen}(\lambda, \omega_2)$ with $\omega_2 \leftarrow\!\!\$\ \{0,1\}^{\mathsf{poly}(\lambda)}$. Furthermore $\omega^* \leftarrow\!\!\$\ \{0,1\}^{\mathsf{poly}(\lambda)}$. So overall, $\mathcal{A}$ is called with two uniformly random bitstrings ω_1, ω^* and a third input which is a fresh public key generated using $\mathsf{SFPK.KGen}$ and freshly drawn random coins, which means that the input to $\mathcal{A}$ has the same distribution as in G_1. Furthermore, $\mathcal{A}$ is given access to a signing oracle that uses the signing key corresponding to the public key which is $\mathcal{A}$'s third input to answer queries. All in all, the view of $\mathcal{A}$ (consisting of input to $\mathcal{A}$ and answers to $\mathcal{A}$'s oracle queries) is identically distributed to $\mathcal{A}$'s view in G_1.

- Let $b = 1$. We have $\omega_1 \leftarrow\!\!\$\ \{0,1\}^{\mathsf{poly}(\lambda)}$, $\mathsf{pk}_2^b := \mathsf{SFPK.ChgPK}(\mathsf{pk}_1, r)$ with $(\mathsf{pk}_1, \cdot) \leftarrow\!\!\$\ \mathsf{SFPK.KGen}(\lambda, \omega_1)$, $r \leftarrow\!\!\$\ \mathsf{SFPK.KeyConvGen}(\mathsf{pp})$, $\mathsf{pp} \leftarrow\!\!\$\ \mathsf{SFPK.PGen}(\lambda)$. Furthermore $\omega^* \leftarrow\!\!\$\ \{0,1\}^{\mathsf{poly}(\lambda)}$. So overall, $\mathcal{A}$ is called with two uniformly random bitstrings and a third input that is an adapted version $\mathsf{pk}' := \mathsf{pk}_2^b$ of the public key pk_1 that can be generated using the random coins from the first component, which means that the input to $\mathcal{A}$ has the same distribution as in G_0. Furthermore, $\mathcal{A}$ is given access to a signing oracle that uses the signing key corresponding to the public key which is $\mathcal{A}$'s third input to answer queries. All in all, the view of $\mathcal{A}$ (consisting of input to $\mathcal{A}$ and answers to $\mathcal{A}$'s oracle queries) is identically distributed to $\mathcal{A}$'s view in G_0.

With the above distribution argument and the fact that $\mathcal{D}$ outputs 1 if and only if $\mathcal{A}$ outputs 1, we get that

$$|\Pr[S_0] - \Pr[S_1]| \leq \mathsf{Adv}^{\mathsf{sfpk\text{-}ror\text{-}ch}}_{\mathcal{D},\Sigma_{\mathsf{SFPK}}}(\lambda)^* \leq \epsilon_1(\lambda)$$

for some negligible function ϵ_1 since $\mathcal{D}$ is a ppt (because $\mathcal{A}$ is a ppt) and Σ_{SFPK} is real-or-random class-hiding. We can analogously prove

$$|\Pr[S_1] - \Pr[S_2]| \leq \epsilon_2(\lambda)$$

for some negligible ϵ_2 which concludes the proof. □

We next prove that the reverse implication of Lem. 4.35 does not hold, i.e. that there are find-original class-hiding SFPK that are not real-or-random class-hiding. Intuitively, this holds since in the find-original class-hiding game, the adversary is always given an adapted key (no matter what bit b is drawn) while in the real-or-random class-hiding game he is given an adapted key if and only if the hidden bit is $b = 1$. So a find-original class-hiding SFPK that marks its adapted keys so that they can efficiently be distinguished from freshly generated keys output by $\mathsf{SFPK.KGen}$ is not real-or-random class-hiding. This is formalized in the following lemma.

Lemma 4.36 *Let Σ_{SFPK} be a find-original class-hiding SFPK over public key space E, secret key space H, message space $\mathcal{M}$, signature space $\mathcal{S}$, trapdoor space T and public key relation $R \subset E \times E$. Define the following SFPK $\Sigma^{\mathsf{mark}}_{\mathsf{SFPK}}$ over public key space $E' := E \times \{0,1\}$, secret key space H, message space $\mathcal{M}$, signature space $\mathcal{S}$, trapdoor space T and public key relation $R_{\mathsf{mark}} \subset E' \times E'$ with*

$$R' = \{((e,b),(e',b')) \mid e \sim_R e'\}$$

with $\Sigma^{\mathsf{mark}}_{\mathsf{SFPK}}$ consisting of the following ppt algorithms:

$\mathsf{SFPK.PGen}_{\mathsf{mark}}(\lambda)$ *Computes and returns* $\mathsf{pp} \leftarrow\!\!\$\ \mathsf{SFPK.PGen}(\lambda)$.

$\mathsf{SFPK.KGen}_{\mathsf{mark}}(\mathsf{pp}, \omega)$ *Computes* $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{SFPK.KGen}(\mathsf{pp}, \omega)$ *and returns* $((\mathsf{pk}, 0), \mathsf{sk})$.

$\mathsf{SFPK.TKGen}_{\mathsf{mark}}(\mathsf{pp}, \omega)$ *Computes* $(\mathsf{pk}, \mathsf{sk}, \tau) \leftarrow\!\!\$\ \mathsf{SFPK.TKGen}(\mathsf{pp}, \omega)$ *and returns* $((\mathsf{pk}, 0), \mathsf{sk}, \tau)$.

$\mathsf{SFPK.Sign}_{\mathsf{mark}}(\mathsf{sk}, m)$ *Computes and returns* $\sigma \leftarrow\!\!\$\ \mathsf{SFPK.Sign}(\mathsf{sk}, m)$.

$\mathsf{SFPK.KeyConvGen}_{\mathsf{mark}}(\mathsf{pp})$ *Computes and returns* $r \leftarrow\!\!\$\ \mathsf{SFPK.KeyConvGen}(\mathsf{pp})$.

$\mathsf{SFPK.ChkRep}_{\mathsf{mark}}((\mathsf{pk}, b), \tau)$ *Computes and returns* $b := \mathsf{SFPK.ChkRep}(\mathsf{pk}, \tau)$.

$\mathsf{SFPK.ChgPK}_{\mathsf{mark}}((\mathsf{pk}, b), r)$ *Computes* $\mathsf{pk}' \leftarrow\!\!\$\ \mathsf{SFPK.ChgPK}(\mathsf{pk}, r)$ *and returns* $(\mathsf{pk}', 1)$.

$\mathsf{SFPK.ChgSK}_{\mathsf{mark}}(\mathsf{sk}, r)$ *Computes and returns* $\mathsf{sk}' \leftarrow\!\!\$\ \mathsf{SFPK.ChgSK}(\mathsf{sk}, r)$.

$\mathsf{SFPK.Vfy}_{\mathsf{mark}}((\mathsf{pk}, b), m, \sigma)$ *Computes and returns* $b := \mathsf{SFPK.Vfy}(\mathsf{pk}, m, \sigma)$.

$\mathsf{SFPK.VKey}_{\mathsf{mark}}((\mathsf{pk}, b), \mathsf{sk})$ *Computes and returns* $\mathsf{SFPK.VKey}(\mathsf{pk}, \mathsf{sk})$.

We have

(i) Σ_{SFPK} *correct (Def. 4.29)* $\Rightarrow$ $\Sigma^{\mathsf{mark}}_{\mathsf{SFPK}}$ *correct*

(ii) Σ_{SFPK} *find-original class-hiding (Def. 4.33)* $\Rightarrow$ $\Sigma^{\mathsf{mark}}_{\mathsf{SFPK}}$ *find-original class-hiding*

(iii) $\Sigma^{\mathsf{mark}}_{\mathsf{SFPK}}$ *is not real-or-random class-hiding (Def. 4.34).*

Proof. **ad (i)** This follows from inspection.

ad (ii) Let $\mathcal{A}$ be a ppt adversary. Consider the find-original class-hiding experiments $\mathsf{Exp}^{\mathsf{sfpk\text{-}fo\text{-}ch}}_{\mathcal{A},\Sigma_{\mathsf{SFPK}}}(\lambda)$ and $\mathsf{Exp}^{\mathsf{sfpk\text{-}fo\text{-}ch}}_{\mathcal{A},\Sigma^{\mathsf{mark}}_{\mathsf{SFPK}}}(\lambda)$ from Def. 4.33. In both games, $\mathcal{A}$ wins (i.e. the output of the game is 1) if $\mathcal{A}$ correctly determines whether $\mathsf{pk}_0 \sim_R \mathsf{pk}'$ or $\mathsf{pk}_1 \sim_R \mathsf{pk}'$ holds.

Fix $\omega_0, \omega_1 \leftarrow\!\!\$\ \{0,1\}^{\mathsf{poly}(\lambda)}$. When given $\omega_0, \omega_1, \mathsf{pk}'$ in $\mathsf{Exp}^{\mathsf{sfpk\text{-}fo\text{-}ch}}_{\mathcal{A},\Sigma_{\mathsf{SFPK}}}(\lambda)$, $\mathcal{A}$ can obviously compute $(\omega_0, \omega_1, (\mathsf{pk}', 1))$, furthermore for any $\omega \in \{0,1\}^{\mathsf{poly}(\lambda)}$ it is obviously possible to compute $\mathsf{SFPK.KGen}_{\mathsf{mark}}(\lambda, \omega)$ from $\mathsf{SFPK.KGen}(\lambda, \omega)$ by just appending a 0.

Informally speaking, $\mathcal{A}$ does not get any new information from the bit appended to the Σ_{SFPK} keys in the $\Sigma^{\mathsf{mark}}_{\mathsf{SFPK}}$ game. Furthermore, the winning condition of $\mathsf{Exp}^{\mathsf{sfpk\text{-}fo\text{-}ch}}_{\mathcal{A},\Sigma^{\mathsf{mark}}_{\mathsf{SFPK}}}(\lambda)$ is the same as the one for $\mathsf{Exp}^{\mathsf{sfpk\text{-}fo\text{-}ch}}_{\mathcal{A},\Sigma_{\mathsf{SFPK}}}(\lambda)$. So if Σ_{SFPK} is find-original class-hiding, $\Sigma^{\mathsf{mark}}_{\mathsf{SFPK}}$ also is find-original class-hiding. A detailed formal reduction is omitted here for brevity.

ad (iii) Let $\mathcal{A}$ be an adversary that, receiving input $(\omega_1, (\mathsf{pk}', b'))$ in the real-or-random class-hiding game $\mathsf{Exp}^{\mathsf{sfpk\text{-}ror\text{-}ch}}_{\mathcal{A},\Sigma^{\mathsf{mark}}_{\mathsf{SFPK}}}(\lambda)$ for $\Sigma^{\mathsf{mark}}_{\mathsf{SFPK}}$, outputs 1 if and only if $b' = 1$. It follows from inspection that $\mathsf{Adv}^{\mathsf{sfpk\text{-}ror\text{-}ch}}_{\mathcal{A},\Sigma^{\mathsf{mark}}_{\mathsf{SFPK}}}(\lambda) = 1$ is not negligible which proves that $\Sigma^{\mathsf{mark}}_{\mathsf{SFPK}}$ is not real-or-random class-hiding according to Def. 4.34.

□

So with Lem. 4.35 and Lem. 4.36, we have proven formally that real-or-random class-hiding for SFPK is indeed a stronger security notion than find-original class-hiding. As mentioned above, the adversary obtains a lot of information about the public keys by seeing the randomness that was used to generate them. So it is natural to define weaker class-hiding notions where the adversary is given less information. An example of this is *adaptive class-hiding (with key corruption)* which was defined by Backes et al. in [BHSB19]. Here, the adversary just sees the secret keys $(\mathsf{sk}_0, \mathsf{sk}_1)$ corresponding to the public keys $(\mathsf{pk}_0, \mathsf{pk}_1)$, respectively.

Definition 4.37 (**SFPK adaptive class-hiding with key corruption**) *Let $\lambda \in \mathbb{N}$ be the security parameter, Σ_{SFPK} be an SFPK.*

(i) We define the security game $\mathsf{Exp}^{\mathsf{adap\text{-}fo\text{-}ch}}_{\mathcal{A},\Sigma_{\mathsf{SFPK}}}(\lambda)$ between adversary $\mathcal{A}$ and challenger $\mathcal{C}$ as the find-original class-hiding game $\mathsf{Exp}^{\mathsf{sfpk\text{-}fo\text{-}ch}}_{\mathcal{A},\Sigma_{\mathsf{SFPK}}}(\lambda)$ from Def. 4.33 with the only change being that line 7 is replaced by

$$b' \leftarrow\!\!\$\ \mathcal{A}^{\mathsf{Sign}(\mathsf{sk}',\cdot)}(\mathsf{pk}_0, \mathsf{sk}_0, \mathsf{pk}_1, \mathsf{sk}_1, \mathsf{pk}')$$

The advantage of $\mathcal{A}$ in the modified security game is defined as

$$\mathsf{Adv}^{\mathsf{adap\text{-}fo\text{-}ch}}_{\mathcal{A},\Sigma_{\mathsf{SFPK}}}(\lambda) := |\Pr[\mathsf{Exp}^{\mathsf{adap\text{-}fo\text{-}ch}}_{\mathcal{A},\Sigma_{\mathsf{SFPK}}}(\lambda) = 1] - \frac{1}{2}|$$

and we say that Σ_{SFPK} is adaptively find-original class-hiding under key corruption *if for all ppt adversaries $\mathcal{A}$ $\mathsf{Adv}^{\mathsf{adap\text{-}fo\text{-}ch}}_{\mathcal{A},\Sigma_{\mathsf{SFPK}}}(\lambda)$ is negligible.*

(ii) We define the security game $\mathsf{Exp}^{\mathsf{adap\text{-}ror\text{-}ch}}_{\mathcal{A},\Sigma_{\mathsf{SFPK}}}(\lambda)$ between adversary $\mathcal{A}$ and challenger $\mathcal{C}$ as the real-or-random class-hiding game $\mathsf{Exp}^{\mathsf{sfpk\text{-}ror\text{-}ch}}_{\mathcal{A},\Sigma_{\mathsf{SFPK}}}(\lambda)$ from Def. 4.34 with the only change being that line 8 is replaced by

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

The advantage of $\mathcal{A}$ in the modified security game is defined as

$$\mathsf{Adv}^{\mathsf{adap\text{-}ror\text{-}ch}}_{\mathcal{A},\Sigma_{\mathsf{SFPK}}}(\lambda) := |\Pr[\mathsf{Exp}^{\mathsf{adap\text{-}ror\text{-}ch}}_{\mathcal{A},\Sigma_{\mathsf{SFPK}}}(\lambda) = 1] - \frac{1}{2}|$$

and we say that Σ_{SFPK} is adaptively real-or-random class-hiding under key corruption *if for all ppt adversaries $\mathcal{A}$ $\mathsf{Adv}^{\mathsf{adap\text{-}ror\text{-}ch}}_{\mathcal{A},\Sigma_{\mathsf{SFPK}}}(\lambda)$ is negligible.*

Obviously, an even weaker class-hiding notion (which we call *adaptive class-hiding without key corruption*) could be defined by not even giving the adversary the secret keys in the above security games.

Definition 4.38 (**SFPK adaptive class-hiding without key corruption**) *Let $\lambda \in \mathbb{N}$ be the security parameter, Σ_{SFPK} be an SFPK.*

(i) We define the security game $\mathsf{Exp}^{\mathsf{adap\text{-}fo\text{-}ch\text{-}nkc}}_{\mathcal{A},\Sigma_{\mathsf{SFPK}}}(\lambda)$ between adversary $\mathcal{A}$ and challenger $\mathcal{C}$ as the find-original class-hiding game $\mathsf{Exp}^{\mathsf{sfpk\text{-}fo\text{-}ch}}_{\mathcal{A},\Sigma_{\mathsf{SFPK}}}(\lambda)$ from Def. 4.33 with the only

change being that line 7 is replaced by

$$b' \leftarrow\!\!\$\ \mathcal{A}^{\mathsf{Sign}(\mathsf{sk}_0,\cdot),\mathsf{Sign}(\mathsf{sk}_1,\cdot),\mathsf{Sign}(\mathsf{sk}',\cdot)}(\mathsf{pk}_0,\mathsf{pk}_1,\mathsf{pk}')$$

The advantage of $\mathcal{A}$ in the modified security game is defined as

$$\mathsf{Adv}^{\text{adap-fo-ch-nkc}}_{\mathcal{A},\Sigma_{\mathsf{SFPK}}}(\lambda) := |\Pr[\mathsf{Exp}^{\text{adap-fo-ch-nkc}}_{\mathcal{A},\Sigma_{\mathsf{SFPK}}}(\lambda)=1]-\frac{1}{2}|$$

and we say that Σ_{SFPK} is adaptively find-original class-hiding without key corruption *if for all ppt adversaries $\mathcal{A}$ $\mathsf{Adv}^{\text{adap-fo-ch-nkc}}_{\mathcal{A},\Sigma_{\mathsf{SFPK}}}(\lambda)$ is negligible.*

(ii) We define the security game $\mathsf{Exp}^{\text{adap-ror-ch-nkc}}_{\mathcal{A},\Sigma_{\mathsf{SFPK}}}(\lambda)$ between adversary $\mathcal{A}$ and challenger $\mathcal{C}$ as the real-or-random class-hiding game $\mathsf{Exp}^{\text{sfpk-ror-ch}}_{\mathcal{A},\Sigma_{\mathsf{SFPK}}}(\lambda)$ from Def. 4.34 with the only change being that line 8 is replaced by

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

The advantage of $\mathcal{A}$ in the modified security game is defined as

$$\mathsf{Adv}^{\text{adap-ror-ch-nkc}}_{\mathcal{A},\Sigma_{\mathsf{SFPK}}}(\lambda) := |\Pr[\mathsf{Exp}^{\text{adap-ror-ch-nkc}}_{\mathcal{A},\Sigma_{\mathsf{SFPK}}}(\lambda)=1]-\frac{1}{2}|$$

and we say that Σ_{SFPK} is adaptively real-or-random class-hiding without key corruption *if for all ppt adversaries $\mathcal{A}$ $\mathsf{Adv}^{\text{adap-ror-ch-nkc}}_{\mathcal{A},\Sigma_{\mathsf{SFPK}}}(\lambda)$ is negligible.*

Note that in the above Def. 4.38, the adversary is given access to signing oracles for the secret keys corresponding to the original public keys. In the class-hiding notions with key corruption (Def. 4.37) and with randomness corruption (Def. 4.33, Def. 4.34), the adversary did not need such oracles since it had direct access to those secret keys. The following lemma summarizes the relations between all the class-hiding notions that were defined above.

Lemma 4.39 *We have the following relations between the class-hiding notions for SFPK:*

(i) Real-or-random class-hiding (Def. 4.34) implies find-original class-hiding (Def. 4.33). Furthermore adaptive real-or-random class-hiding with/without key corruption implies find-original class-hiding with/without key corruption (see Def. 4.37, Def. 4.38).

(ii) Real-or-random class-hiding (Def. 4.34) implies adaptive real-or-random class-hiding with key corruption (Def. 4.37) which implies adaptive real-or-random class-hiding without key corruption (Def. 4.38).

(iii) Find-original class-hiding (Def. 4.33) implies adaptive find-original class-hiding with key corruption (Def. 4.37) which implies adaptive find-original class-hiding without key corruption (Def. 4.38).

where "A implies B" means that an SFPK fulfilling notion A also fulfills notion B.

Proof. **ad (i)** This follows from Lem. 4.35.

ad (ii) We will sketch a proof for the claim that real-or-random class-hiding implies adaptive real-or-random class-hiding with key corruption. The other statement is proven analogously.

We use a simple reduction argument. Let $\mathcal{A}$ be a ppt adversary with $\mathsf{Adv}^{\mathsf{adap\text{-}ror\text{-}ch}}_{\mathcal{A},\Sigma_{\mathsf{SFPK}}}(\lambda)$ not negligible. A ppt adversary $\mathcal{B}$ with non-negligible $\mathsf{Adv}^{\mathsf{sfpk\text{-}ror\text{-}ch}}_{\mathcal{B},\Sigma_{\mathsf{SFPK}}}(\lambda)$ can easily be constructed as follows: on input $\omega_1, \mathsf{pk}_2^b$ as in $\mathsf{Exp}^{\mathsf{sfpk\text{-}ror\text{-}ch}}_{\mathcal{B},\Sigma_{\mathsf{SFPK}}}(\lambda)$, $\mathcal{B}$ computes $(\mathsf{pk}_1, \mathsf{sk}_1) := \mathsf{SFPK.KGen}(\lambda, \omega_1)$ and calls $\mathcal{A}(\mathsf{pk}_1, \mathsf{sk}_1, \mathsf{pk}_2^b)$. $\mathcal{B}$ relays all of $\mathcal{A}$'s signing oracle queries to its own signing oracle and outputs the same bit that $\mathcal{A}$ eventually outputs. It is clear to see that

$$\mathsf{Adv}^{\mathsf{sfpk\text{-}ror\text{-}ch}}_{\mathcal{B},\Sigma_{\mathsf{SFPK}}}(\lambda) = \mathsf{Adv}^{\mathsf{adap\text{-}ror\text{-}ch}}_{\mathcal{A},\Sigma_{\mathsf{SFPK}}}(\lambda)$$

is not negligible which proves the claim.

ad (iii) This can be proven with simple reduction arguments, as seen in (ii).

□

Recoverable signing keys To conclude the SFPK subsection, we introduce a specific class of SFPK which is called *SFPK with recoverable signing keys* and was introduced in [BHKS18]. In a standard application, the public and secret key of a key pair are jointly randomized by the signer. However, there are scenarios, e.g. *stealth addresses* in cryptocurrency systems, where a sender randomizes the public key pk of a receiver of cryptocurrency funds and then publishes the randomized public key $\mathsf{pk}' \in [\mathsf{pk}]_R$ so the receiver can retrieve it. It then sends the funds to this public key and the receiver needs to retrieve the corresponding secret key sk' to pk' in order to be able to spend the money. Note that if the receiver is able to do so, the sender can transfer money to her without ever interacting with her. However, the standard SFPK syntax definition does not include such a key recovery which is why the following specific type of SFPK scheme was introduced:

Definition 4.40 (**recoverable signing keys**) *An SFPK* Σ_{SFPK} has recoverable signing keys *if there exists a ppt algorithm*

$\mathsf{SFPK.Recover}(\mathsf{sk}, \tau, \mathsf{pk}')$: *takes as input an original secret key* sk*, a recovery trapdoor* τ *and a target public key* pk' *and outputs a secret key* sk'

such that for all $\lambda \in \mathbb{N}$, $\omega \leftarrow\!\!\$\ \{0,1\}^{\mathsf{poly}(\lambda)}, r \leftarrow\!\!\$\ \mathsf{SFPK.KeyConvGen}(\mathsf{pp})$, $(\mathsf{pk}, \mathsf{sk}, \tau) \leftarrow\!\!\$\ \mathsf{SFPK.TKGen}(\lambda, \omega)$, $\mathsf{pk}' := \mathsf{SFPK.ChgPK}(\mathsf{pk}, r)$ *it holds that*

$$\mathsf{SFPK.ChgSK}(\mathsf{sk}, r) = \mathsf{SFPK.Recover}(\mathsf{sk}, \tau, \mathsf{pk}')$$

So an SFPK has recoverable signing keys if the trapdoor τ for a key pair $(\mathsf{pk}, \mathsf{sk})$ can be used to recover the secret key sk' corresponding to an adapted public key pk'

if one possesses the secret key sk corresponding to the original public key pk. Note that if the recovery algorithm would only take the original key pair as inputs, no SFPK with recoverable signing keys could be class-hiding (since the class-hiding adversary from Def. 4.33 is given the secret keys corresponding to the initial public keys $\mathsf{pk}_0, \mathsf{pk}_1$ and thus has access to the corresponding secret keys $\mathsf{sk}_0, \mathsf{sk}_1$).

An example of an SFPK with recoverable signing keys is the warm-up SFPK scheme by Backes et al. [BHKS18] which we will cover in more detail in Sect. 5.3. Its recoverable signing keys make this scheme suitable to instantiate stealth address schemes as introduced in [Tod].

4.4 Mercurial signatures

In this section, we formally define mercurial signatures which were introduced by Crites and Lysyanskaya in [CL19]. A mercurial signature scheme allows to randomize a triple (pk, m, σ) where σ is a valid signature for message m under public key pk to another valid triple $(\mathsf{pk}', m', \sigma')$ where m' and m as well as pk' and pk are related via equivalence relations that are defined on the message and public key spaces, respectively. The following syntax definition is based on [CL19].

Definition 4.41 (**mercurial signature syntax**) *Let $l \in \mathbb{N}$, $\mathcal{M}, E, H$ be sets, $R_m \subseteq \mathcal{M} \times \mathcal{M}$, $R_{\mathsf{pk}} \subseteq E \times E$ and $R_{\mathsf{sk}} \subseteq H \times H$ be equivalence relations. A* mercurial signature scheme Σ_{Merc} *over message relation R_m, public key relation R_{pk} and secret key relation R_{sk} with message space $\mathcal{M}$, public key space E, secret key space H and signature space $\mathcal{S}$ is a tuple of ppt algorithms as follows:*

$\mathsf{Merc.PGen}(\lambda)$ *probablistic parameter generation algorithm, takes as input the security parameter $\lambda \in \mathbb{N}$ and outputs public parameters* pp

$\mathsf{Merc.KGen}(\mathsf{pp}, l)$ *probablistic key generation algorithm, takes as input public parameters* pp *and the length parameter l and outputs a key pair $(\mathsf{pk}, \mathsf{sk}) \in E \times H$, consisting of a public key* pk *and a secret key* sk

$\mathsf{Merc.Sign}(\mathsf{sk}, m)$ *probablistic signing algorithm, takes as input a secret key $\mathsf{sk} \in H$ and a message $m \in \mathcal{M}$ and outputs a signature $\sigma \in \mathcal{S}$*

$\mathsf{Merc.Vfy}(\mathsf{pk}, m, \sigma)$ *deterministic verification algorithm, takes as input a public key $\mathsf{pk} \in E$, a message $m \in \mathcal{M}$ and a signature $\sigma \in \mathcal{S}$ and outputs a bit $b \in \{0, 1\}$*

$\mathsf{Merc.KeyConvGen}(\mathsf{pp})$ *probablistic key conversion randomness generation algorithm, takes as input public parameters* pp *and outputs randomness r_k*

$\mathsf{Merc.ConvertPK}(\mathsf{pk}, r_k)$ *deterministic public key conversion algorithm, takes as input a public key $\mathsf{pk} \in E$ and randomness r_k for public parameters* pp *and outputs a public key $\mathsf{pk}' \in E$*

$\mathsf{Merc.ConvertSK}(\mathsf{sk}, r_k)$ *deterministic secret key conversion algorithm, takes as input a secret key* sk *and randomness* r_k *for public parameters* pp *and outputs a secret key* sk'

$\mathsf{Merc.AdaptSig}(\mathsf{pk}, m, \sigma, r_k)$ *probablistic signature adaption algorithm, takes as input a public key* $\mathsf{pk} \in E$*, a message* $m \in \mathcal{M}$*, a signature* $\sigma \in \mathcal{S}$ *and key conversion randomness* r_k *and outputs a signature* $\sigma' \in \mathcal{S}$

$\mathsf{Merc.ConvSigGen}(\mathsf{pp})$ *probablistic signature conversion randomness generation algorithm, takes as input public parameters* pp *and outputs randomness* r_σ

$\mathsf{Merc.ChgRep}(\mathsf{pk}, m, \sigma, r_\sigma)$ *probablistic signature conversion algorithm, takes as input a public key* $\mathsf{pk} \in E$*, a message* $m \in \mathcal{M}$*, a signature* $\sigma \in \mathcal{S}$ *and signature conversion randomness* r_σ *and outputs a message-signature pair* $(m', \sigma') \in \mathcal{M} \times \mathcal{S}$

$\mathsf{Merc.VKey}(\mathsf{pk}, \mathsf{sk})$ *deterministic key pair verification algorithm, takes as input a public key* $\mathsf{pk} \in E$ *and a secret key* $\mathsf{sk} \in H$ *and outputs a bit* $b \in \{0, 1\}$

If not explicitly stated otherwise, we always assume that the algorithms of a mercurial signature scheme are named as in Def. 4.41. The set of values r_k that can be input to the $\mathsf{Merc.ConvertPK}$, $\mathsf{Merc.ConvertSK}$ and $\mathsf{Merc.AdaptSig}$ algorithms is implicitly defined by the particular mercurial signature and contains the support of $\mathsf{Merc.KeyConvGen}$ as a subset. Analogously, the set of values r_σ that can be input to the $\mathsf{Merc.ChgRep}$ algorithm is also implicitly by the particular mercurial signature and contains the support of $\mathsf{Merc.ConvSigGen}$ as a subset. Note that the message and key spaces $\mathcal{M}, E, H$ of a mercurial signature Σ_{Merc} as well as the equivalence relations R_m, R_{pk} and R_{sk} on them depend on the public parameters $\mathsf{pp} \leftarrow\!\!\$\ \mathsf{Merc.PGen}(\lambda)$. With the syntax of mercurial signatures defined, we next formalize their correctness requirements. The following definition is based on [CL19].

Definition 4.42 (**mercurial signature correctness**) *Let* $l \in \mathbb{N}$*,* $\mathcal{M}, E, H$ *be sets,* $R_m \subseteq \mathcal{M} \times \mathcal{M}$*,* $R_{\mathsf{pk}} \subseteq E \times E$ *and* $R_{\mathsf{sk}} \subseteq H \times H$ *be equivalence relations,* Σ_{Merc} *a mercurial signature scheme over message relation* R_m*, public key relation* R_{pk} *and secret key relation* R_{sk} *with message space* $\mathcal{M}$*, public key space* E*, secret key space* H *and signature space* $\mathcal{S}$*.* Σ_{Merc} *is* correct *if it fulfills the following conditions for all* $\lambda \in \mathbb{N}$*,* $\mathsf{pp} \leftarrow\!\!\$\ \mathsf{Merc.PGen}(\lambda)$*,* $l > 1$*,* $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{Merc.KGen}(\mathsf{pp}, l)$*:*

(i) $\mathsf{Merc.VKey}(\mathsf{pk}, \mathsf{sk}) = 1$

(ii) *For all* $m \in \mathcal{M}$*,* $\sigma \leftarrow\!\!\$\ \mathsf{Merc.Sign}(\mathsf{sk}, m)$ *it holds that* $\mathsf{Merc.Vfy}(\mathsf{pk}, m, \sigma) = 1$*.*

(iii) *For all* $r_k \leftarrow\!\!\$\ \mathsf{Merc.KeyConvGen}(\mathsf{pp})$*,* $\mathsf{pk}' := \mathsf{Merc.ConvertPK}(\mathsf{pk}, r_k)$*,* $\mathsf{sk}' := \mathsf{Merc.ConvertSK}(\mathsf{sk}, r_k)$*, we have*

$$\mathsf{sk}' \in [\mathsf{sk}]_{R_{\mathsf{sk}}} \wedge \mathsf{pk}' \in [\mathsf{pk}]_{R_{\mathsf{pk}}} \wedge \mathsf{Merc.VKey}(\mathsf{pk}', \mathsf{sk}') = 1$$

(iv) *For all* $m \in \mathcal{M}$*,* $\sigma \in \mathcal{S}$ *with* $\mathsf{Merc.Vfy}(\mathsf{pk}, m, \sigma) = 1$*,* $r_k \leftarrow\!\!\$\ \mathsf{Merc.KeyConvGen}(\mathsf{pp})$*,* $\sigma' \leftarrow\!\!\$\ \mathsf{Merc.AdaptSig}(\mathsf{pk}, m, \sigma, r_k)$ *and* $\mathsf{pk}' := \mathsf{Merc.ConvertPK}(\mathsf{pk}, r_k)$ *we have*

$$\mathsf{Merc.Vfy}(\mathsf{pk}', m, \sigma') = 1$$

(v) For all $m \in \mathcal{M}$, $\sigma \in \mathcal{S}$ *with* $\mathsf{Merc.Vfy}(\mathsf{pk}, m, \sigma) = 1$, $r_\sigma \leftarrow\!\!\$\ \mathsf{Merc.ConvSigGen}(\mathsf{pp})$, $(m', \sigma') \leftarrow\!\!\$\ \mathsf{Merc.ChgRep}(\mathsf{pk}, m, \sigma, r_\sigma)$, *we have*

$$\mathsf{Merc.Vfy}(\mathsf{pk}, m', \sigma') = 1 \wedge m' \in [m]_{R_m}$$

The first requirement demands that all honestly generated key pairs are valid. The Merc.VKey algorithm was not part of the syntax of a mercurial signature in [CL19], however, we added it to be able to fully express the notion of (valid) keys that were not generated using the Merc.KGen algorithm. This is especially important when it comes to maliciously generated keys, e.g. keys that are generated by some adversarial party as seen in later security definitions for mercurial signatures (see for example Def. 4.51). The second requirement is the standard correctness requirement for digital signatures which demands that all honestly generated signatures are valid under the respective public key. This implies that in the above setting, $\Sigma :=$ (Merc.PGen, Merc.KGen, Merc.Sign, Merc.Vfy) is a correct regular digital signature scheme. The third requirement ensures that, when its keys are randomized consistently with the same randomness r_k, any honestly generated key pair stays valid. Furthermore, the resulting secret and public key are related to the respective old one via the respective equivalence relation. We saw a similar correctness requirement for SFPK in Def. 4.29. The fourth requirement ensures that a signature σ on a message m can be adapted to a new public key pk' so that the new triple $(\mathsf{pk}', m, \sigma')$ is still valid. The fifth requirement demands that Merc.ChgRep can be used to randomize a valid message-signature pair (m, σ) into a new pair (m', σ') valid under the same public key, where m' is another representative of the class $[m]_{R_m}$ of m. We had a similar correctness requirement for SPS-EQ in Def. 4.20.

As a final remark for the syntax of mercurial signatures, we note that while both SFPK (Def. 4.28) and mercurial signatures have an equivalence relation $R_{\mathsf{pk}} \subset E \times E$ on the public key space E, only the mercurial definition explicitly mentions an equivalence relation $R_{\mathsf{sk}} \subset H \times H$ on the secret key space H. However, the next lemma shows that this is not actually a syntactical difference since for an SFPK (Def. 4.28) over a (public key) relation R_{pk}, we can construct a secret key relation R_{sk} that fulfills the correctness requirement for mercurial signatures (Def. 4.42) in a black-box way.

Lemma 4.43 *Let* $\lambda \in \mathbb{N}$ *be the security parameter. Let* Σ_{SFPK} *be an SFPK as in Def. 4.29 over (public key) equivalence relation* $R_{\mathsf{pk}} \subset E \times E$. *Then we have a secret key equivalence relation* $R_{\mathsf{sk}} \subset H \times H$ *on the secret key space* H *as follows:*

$$\begin{aligned}\mathsf{sk} \sim_{R_{\mathsf{sk}}} \mathsf{sk}' \Leftrightarrow \exists \mathsf{pk}, \mathsf{pk}' \in E :& \mathsf{SFPK.VKey}(\mathsf{pk}, \mathsf{sk}) = 1 \\ &\wedge \mathsf{SFPK.VKey}(\mathsf{pk}', \mathsf{sk}') = 1 \\ &\wedge \mathsf{pk} \sim_{R_{\mathsf{pk}}} \mathsf{pk}'\end{aligned}$$

This relation fulfills the mercurial signature correctness requirement for secret key relations R_{sk} *(Def. 4.42) which translates to*

$$\mathsf{sk} \sim_{R_{\mathsf{sk}}} \mathsf{sk}'$$

for all $\mathsf{pp} \leftarrow\!\$\ \mathsf{SFPK.PGen}(\lambda)$, $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\$\ \mathsf{SFPK.KGen}(\mathsf{pp})$, $r_k \leftarrow\!\$\ \mathsf{KeyConvGen}(\mathsf{pp})$, $\mathsf{pk}' \leftarrow\!\$\ \mathsf{SFPK.ChgPK}(\mathsf{pk}, r_k)$, $\mathsf{sk}' \leftarrow\!\$\ \mathsf{SFPK.ChgSK}(\mathsf{sk}, r_k)$ *in the SFPK case.*

Proof. Let $\lambda \in \mathbb{N}$ be the security parameter, $\mathsf{pp} \leftarrow\!\$\ \mathsf{SFPK.PGen}(\lambda)$. We can use Rem. 3.16 to conclude without loss of generality that H is the support of $\mathsf{SFPK.KGen}(\mathsf{pp})$ and thus for every secret key $\mathsf{sk} \in H$, there is a public key $\mathsf{pk} \in E$ such that $\mathsf{SFPK.VKey}(\mathsf{pk}, \mathsf{sk}) = 1$.

With this remark, reflexivity of R_{sk} is obviously implied by the reflexivity of the equivalence relation R_{pk}. We analogously conclude symmetry and transitivity of R_{sk} from the respective properties of R_{pk} and thus have proven that R_{sk} is a (secret key) equivalence relation.

What is left to prove is the fulfillment of the correctness requirement for mercurial signatures. We see that by correctness of the SFPK (Def. 4.29), we get that $\mathsf{pk} \sim_{R_{\mathsf{pk}}} \mathsf{pk}'$ which by definition of R_{sk} immediately concludes $\mathsf{sk} \sim_{R_{\mathsf{sk}}} \mathsf{sk}'$ for honestly generated key pairs $(\mathsf{pk}, \mathsf{sk})$ and $(\mathsf{pk}', \mathsf{sk}')$, as required. □

Note that not every SFPK or mercurial signature has to use the secret key relation R_{sk} from Lem. 4.43 based on its public key relation R_{pk}. Lem. 4.43 is only used to constructively prove that if a signature scheme has a public key relation R_{pk}, then it also has a secret key relation R_{sk}.

Unforgeability After defining syntax and correctness of mercurial signatures, we now turn towards their security. We start with adapting the standard unforgeability notion for digital signatures. Analogously to SPS-EQ (Def. 4.21), an unforgeability adversary must be required to forge a signature for a message from a new class for that it has not seen any signatures yet. This is because otherwise, the $\mathsf{Merc.ChgRep}$ algorithm would yield a trivial way to forge signatures. Analogously to SFPK (Def. 4.30), an unforgeability adversary is allowed to forge a signature under an arbitrary public key pk^* that is related to the challenge key pk. The following unforgeability definition is based on [CL19].

Definition 4.44 (EUF-CMA for mercurial signatures) *Let $\lambda \in \mathbb{N}$ be the security parameter, $l : \mathbb{N} \to \mathbb{N}$ be a polynomial function. For a mercurial signature scheme Σ_{Merc} over message relation R_m, public key relation R_{pk} and secret key relation R_{sk}, we define the following security game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$:*

$\mathsf{Exp}^{\mathsf{merc\text{-}euf}}_{\mathcal{A},\Sigma_{\mathsf{Merc}}}(\lambda)$

$\mathsf{pp} \leftarrow\!\!\$\ \mathsf{Merc.PGen}(\lambda)$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{Merc.KGen}(\mathsf{pp}, l)$
$Q := \emptyset$
$(\mathsf{pk}^*, m^*, \sigma^*) \leftarrow\!\!\$\ \mathcal{A}^{\mathsf{Sign}(\mathsf{sk},\cdot)}(\mathsf{pp}, \mathsf{pk})$
if $\forall m \in Q : m^* \notin [m]_{R_m}$
  $\wedge\ \mathsf{pk}^* \in [\mathsf{pk}]_{R_{\mathsf{pk}}}$
  $\wedge\ \mathsf{Merc.Vfy}(\mathsf{pk}^*, m^*, \sigma^*) = 1$
  return 1
else return 0

with oracle

$\mathsf{Sign}(\mathsf{sk}, m)$

$\sigma \leftarrow\!\!\$\ \mathsf{Merc.Sign}(\mathsf{sk}, m)$
$Q := Q \cup \{(m, \sigma)\}$
return σ

We define the advantage of $\mathcal{A}$ in the above security game as

$$\mathsf{Adv}^{\mathsf{merc\text{-}euf}}_{\mathcal{A},\Sigma_{\mathsf{Merc}}}(\lambda) := \Pr[\mathsf{Exp}^{\mathsf{merc\text{-}euf}}_{\mathcal{A},\Sigma_{\mathsf{Merc}}}(\lambda) = 1]$$

Σ_{Merc} *is* existentially unforgeable under chosen-message attacks (EUF-CMA secure) *if for all ppt adversaries $\mathcal{A}$ $\mathsf{Adv}^{\mathsf{merc\text{-}euf}}_{\mathcal{A},\Sigma_{\mathsf{Merc}}}(\lambda)$ is negligible.*

So for an unforgeable mercurial signature scheme, it should be hard to efficiently forge a signature for a new class under a public key that is related to the challenge key. Note that in contrast to the SFPK unforgeability experiment from Def. 4.30, no equivalent for the randomize-then-sign oracle $\mathsf{Sign}(\mathsf{sk}, \cdot, \cdot)$ from $\mathsf{Exp}^{\mathsf{sfpk\text{-}euf}}_{\mathcal{A},\Sigma_{\mathsf{SFPK}}}(\lambda)$ is accessible to the adversary in the above experiment $\mathsf{Exp}^{\mathsf{merc\text{-}euf}}_{\mathcal{A},\Sigma_{\mathsf{Merc}}}(\lambda)$. Note that this clearly weakens the adversaries power since it now cannot observe any fresh signatures under randomized versions of the challenge key sk. While the $\mathsf{Merc.AdaptSig}$ algorithm can be used to obtain signatures under randomized versions of sk, it is not clear whether these signatures have the same distribution as fresh signatures that are output using $\mathsf{Merc.Sign}$. For completeness, we will define a game for a stronger unforgeability notion for mercurials in Def. 4.45, which gives the adversary access to such a randomize-then-sign oracle. It is however left as an open research question whether mercurial signatures that fulfill such a stronger unforgeability notion do exist.

Definition 4.45 (EUF-CMA game variant for mercurial signatures) *Let $\lambda \in \mathbb{N}$ be the security parameter, $l : \mathbb{N} \to \mathbb{N}$ be a polynomial function. For a mercurial signature scheme Σ_{Merc} over message relation R_m, public key relation R_{pk} and secret key relation R_{sk}, we define the security experiment $\mathsf{Exp}^{\mathsf{merc\text{-}euf}^*}_{\mathcal{A},\Sigma_{\mathsf{Merc}}}(\lambda)$ for an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$ identically to $\mathsf{Exp}^{\mathsf{merc\text{-}euf}}_{\mathcal{A},\Sigma_{\mathsf{Merc}}}(\lambda)$, with the only change being that $\mathcal{A}$ can access another oracle $\mathsf{Sign}(\mathsf{sk}, \cdot, \cdot)$ which is defined as follows:*

$\mathsf{Sign}(\mathsf{sk}, m, r)$

1 : $\mathsf{sk}' := \mathsf{Merc.ConvertSK}(\mathsf{sk}, r)$
2 : $\sigma \leftarrow\!\$\ \mathsf{Merc.Sign}(\mathsf{sk}', m)$
3 : $Q := Q \cup \{(m, \sigma)\}$
4 : **return** σ

To conclude the unforgeability discussion for mercurial signatures we want to remark that, analogously to SFPK unforgeability in Lem. 4.32, the public key relation R_{pk} of an unforgeable mercurial signature scheme needs to have "small" classes if the public keys output by Merc.KGen are uniformly random.

Class-hiding Next we define class-hiding for mercurial signatures, which is split up into two separate definitions called message class-hiding and public key class-hiding. We begin with message class hiding which basically requires that, given two random messages, it is hard to tell whether they are related via the message relation or not. So message class-hiding for mercurial signatures is analogous to message class-hiding for SPS-EQ as defined in Def. 4.23. We next formally define message class-hiding for mercurial signatures where we base our definition on [CL19].

Definition 4.46 (**message class-hiding**) *Let $\lambda \in \mathbb{N}$ be the security parameter, $l : \mathbb{N} \to \mathbb{N}$ be a polynomial function. For a mercurial signature scheme Σ_{Merc} over message relation R_m, public key relation R_{pk} and secret key relation R_{sk}, we define the following security game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$:*

$\mathsf{Exp}^{\mathsf{merc\text{-}mes\text{-}ch}}_{\mathcal{A},\Sigma_{\mathsf{Merc}}}(\lambda)$

1 : $\mathsf{pp} \leftarrow\!\$\ \mathsf{Merc.PGen}(\lambda)$
2 : $m_1 \leftarrow\!\$\ \mathcal{M}$
3 : $m_2^0 \leftarrow\!\$\ [m_1]_{R_m}, m_2^1 \leftarrow\!\$\ \mathcal{M}$
4 : $b \leftarrow\!\$\ \{0,1\}$
5 : $b' \leftarrow\!\$\ \mathcal{A}(\mathsf{pp}, m_1, m_2^b)$
6 : **return** 1 **if** $b = b'$
7 : **else return** 0

We define the advantage of $\mathcal{A}$ in the above security game as

$$\mathsf{Adv}^{\mathsf{merc\text{-}mes\text{-}ch}}_{\mathcal{A},\Sigma_{\mathsf{Merc}}}(\lambda) := |\Pr[\mathsf{Exp}^{\mathsf{merc\text{-}mes\text{-}ch}}_{\mathcal{A},\Sigma_{\mathsf{Merc}}}(\lambda) = 1] - \frac{1}{2}|$$

Σ_{Merc} *is* message class-hiding *if for all ppt adversaries $\mathcal{A}$ $\mathsf{Adv}^{\mathsf{merc\text{-}mes\text{-}ch}}_{\mathcal{A},\Sigma_{\mathsf{Merc}}}(\lambda)$ is negligible.*

Note that we could define real-or-random message class-hiding for mercurial signatures analogously to what we did for SPS-EQ in Def. 4.22. This is skipped here for space reasons since the message class-hiding definition above (Def. 4.46) is the one found in the literature [CL19].

We next define the other class-hiding notion for mercurial signatures, which is public-key class-hiding. We first state a definition based on the one from [CL19] which is analogous to the SFPK real-or-random class-hiding notion (without key corruption) from Def. 4.38.

Definition 4.47 (**(real-or-random) public-key class-hiding**) *Let $\lambda \in \mathbb{N}$ be the security parameter, $l : \mathbb{N} \to \mathbb{N}$ be a polynomial function. For a mercurial signature scheme Σ_{Merc} over message relation R_m, public key relation R_{pk} and secret key relation R_{sk}, we define the following security game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$:*

$\mathsf{Exp}^{\mathsf{merc\text{-}ror\text{-}pk\text{-}ch}}_{\mathcal{A},\Sigma_{\mathsf{Merc}}}(\lambda)$

1 : $\mathsf{pp} \leftarrow\!\$\ \mathsf{Merc.PGen}(\lambda)$
2 : $(\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow\!\$\ \mathsf{Merc.KGen}(\mathsf{pp}, l(\lambda))$
3 : $(\mathsf{pk}_2^0, \mathsf{sk}_2^0) \leftarrow\!\$\ \mathsf{Merc.KGen}(\mathsf{pp}, l(\lambda))$
4 : $r_k \leftarrow\!\$\ \mathsf{Merc.KeyConvGen}(\mathsf{pp})$
5 : $(\mathsf{pk}_2^1, \mathsf{sk}_2^1) := (\mathsf{Merc.ConvertPK}(\mathsf{pk}, r_k), \mathsf{Merc.ConvertSK}(\mathsf{sk}, r_k))$
6 : $b \leftarrow\!\$\ \{0,1\}$
7 : $b' \leftarrow\!\$\ \mathcal{A}^{\mathsf{Merc.Sign}(\mathsf{sk}_1,\cdot),\mathsf{Merc.Sign}(\mathsf{sk}_2^b,\cdot)}(\mathsf{pk}_1, \mathsf{pk}_2^b)$
8 : **return** 1 **if** $b = b'$**else return** 0

We define the advantage of $\mathcal{A}$ in the above security game as

$$\mathsf{Adv}^{\mathsf{merc\text{-}ror\text{-}pk\text{-}ch}}_{\mathcal{A},\Sigma_{\mathsf{Merc}}}(\lambda) := |\Pr[\mathsf{Exp}^{\mathsf{merc\text{-}ror\text{-}pk\text{-}ch}}_{\mathcal{A},\Sigma_{\mathsf{Merc}}}(\lambda) = 1] - \frac{1}{2}|$$

Σ_{Merc} *is* (real-or-random) public-key class-hiding *if for all ppt adversaries $\mathcal{A}$ $\mathsf{Adv}^{\mathsf{merc\text{-}ror\text{-}pk\text{-}ch}}_{\mathcal{A},\Sigma_{\mathsf{Merc}}}(\lambda)$ is negligible.*

So public-key class-hiding for mercurial signatures basically states that it is hard to tell whether two given key pairs are related or independently drawn and unrelated. Note that the game works different than the find-original class-hiding game $\mathsf{Exp}^{\mathsf{adap\text{-}fo\text{-}ch\text{-}nkc}}_{\mathcal{A},\Sigma_{\mathsf{SFPK}}}(\lambda)$ (without key corruption) from Def. 4.38. Here, the second key pair is either drawn uniformly at random or derived from the first one (depending on the hidden bit b), while in Def. 4.38, the hidden bit decides which of the two key pairs is randomized into the challenge key pair $(\mathsf{pk}', \mathsf{sk}')$. In the following, we will define find-original public-key class-hiding for mercurial signatures in an analogous way than we did for SFPK.

Definition 4.48 (**find-original public-key class-hiding**) *Let $\lambda \in \mathbb{N}$ be the security parameter, $l : \mathbb{N} \to \mathbb{N}$ be a polynomial function. For a mercurial signature scheme Σ_{Merc} over message relation R_m, public key relation R_{pk} and secret key relation R_{sk}, we define the following security game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$:*

$\mathsf{Exp}^{\mathsf{merc\text{-}fo\text{-}pk\text{-}ch}}_{\mathcal{A},\Sigma_{\mathsf{Merc}}}(\lambda)$

1 : $\mathsf{pp} \leftarrow\!\$\ \mathsf{Merc.PGen}(\lambda)$
2 : $(\mathsf{pk}_0, \mathsf{sk}_0), (\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow\!\$\ \mathsf{Merc.KGen}(\mathsf{pp}, l(\lambda))$
3 : $b \leftarrow\!\$\ \{0,1\}$
4 : $r_k \leftarrow\!\$\ \mathsf{Merc.KeyConvGen}(\mathsf{pp})$
5 : $(\mathsf{pk}', \mathsf{sk}') := (\mathsf{Merc.ConvertPK}(\mathsf{pk}_b, r_k), \mathsf{Merc.ConvertSK}(\mathsf{sk}_b, r_k))$
6 : $b' \leftarrow\!\$\ \mathcal{A}^{\mathsf{Merc.Sign}(\mathsf{sk}_0,\cdot),\mathsf{Merc.Sign}(\mathsf{sk}_1,\cdot),\mathsf{Merc.Sign}(\mathsf{sk}',\cdot)}(\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{pk}')$
7 : **return** 1 **if** $b = b'$ **else return** 0

We define the advantage of $\mathcal{A}$ in the above security game as

$$\mathsf{Adv}^{\mathsf{merc\text{-}fo\text{-}pk\text{-}ch}}_{\mathcal{A},\Sigma_{\mathsf{Merc}}}(\lambda) := |\Pr[\mathsf{Exp}^{\mathsf{merc\text{-}fo\text{-}pk\text{-}ch}}_{\mathcal{A},\Sigma_{\mathsf{Merc}}}(\lambda) = 1] - \frac{1}{2}|$$

Σ_{Merc} *is* find-original public-key class-hiding *if for all ppt adversaries* $\mathcal{A}$ $\mathsf{Adv}^{\mathsf{merc\text{-}fo\text{-}pk\text{-}ch}}_{\mathcal{A},\Sigma_{\mathsf{Merc}}}(\lambda)$ *is negligible.*

Remark 4.49 *While we skip this here for space reasons, it is important to note that one could continue the class-hiding discussion for mercurial signatures (by giving the adversary more information about the challenge key pairs) to ultimately prove a result for mercurial signature class-hiding that is analogous to the one for SFPKs from Lem. 4.39. In particular, note that, analogously to SFPK, real-or-random public-key class-hiding for mercurial signatures (Def. 4.47) is stronger than find-original public-key class-hiding for mercurial signatures (Def. 4.48) where the proof is analogous to Lem. 4.36 and Lem. 4.35.*

If not explicitly stated otherwise, we will use the real-or-random public-key class-hiding notion from Def. 4.47 for mercurial signatures since it was used by Crites et al. to prove their mercurial construction from [CL19] secure. Mainly for the reason of having a single reference number for it, we formally summarize what class-hiding for mercurial signatures means.

Definition 4.50 (**class-hiding mercurial signature**) *A mercurial signature scheme* Σ_{Merc} *is* class-hiding *if it is both message class-hiding (Def. 4.46) and public-key class-hiding (Def. 4.47).*

Origin-hiding Next, we define the last security property for mercurial signatures which is origin-hiding. This basically states that key-message-signature triples $(\mathsf{pk}', m', \sigma')$ that are obtained via the Merc.ChgRep and Merc.AdaptSig algorithms of a mercurial signature scheme do not reveal anything about the original input triple to the algorithms. More precisely, the output triple looks like a uniformly random public key pk' and a uniformly random message m' with a uniformly random signature σ' for it that is valid under pk'. This should even hold in the case that the input public key pk for Merc.ChgRep and Merc.AdaptSig was maliciously generated. The above behaviour is formalized in the following definition which is based on [CL19].

Definition 4.51 (**origin-hiding**) *Let $l \in \mathbb{N}$, $\mathcal{M}, E, H$ be sets, $R_m \subseteq \mathcal{M} \times \mathcal{M}$, $R_{\mathsf{pk}} \subseteq$*

$E \times E$ and $R_{\mathsf{sk}} \subseteq H \times H$ be equivalence relations, Σ_{Merc} a mercurial signature scheme over message relation R_m, public key relation R_{pk} and secret key relation R_{sk} with message space $\mathcal{M}$, public key space E, secret key space H and signature space $\mathcal{S}$. Furthermore let $\lambda \in \mathbb{N}$, m a message, $\mathsf{pp} \leftarrow\!\!\$\ \mathsf{Merc.PGen}(\lambda)$, $\mathsf{pk} \in E$. Σ_{Merc} is origin-hiding *if it fulfills the following two conditions:*

(i) Assume $\sigma \in \mathcal{S}$ with $\mathsf{Merc.Vfy}(\mathsf{pk}, m, \sigma) = 1$, $r_\sigma \leftarrow\!\!\$\ \mathsf{Merc.ConvSigGen}(\mathsf{pp})$, $(m', \sigma') \leftarrow\!\!\$\ \mathsf{Merc.ChgRep}(\mathsf{pk}, m, \sigma, r_\sigma)$. Then m' is identically distributed to a uniformly random message in $[m]_{R_m}$, furthermore σ' is identically distributed to a uniformly random signature in

$$\{\hat{\sigma} \in \mathcal{S} \mid \mathsf{Merc.Vfy}(\mathsf{pk}, m', \hat{\sigma}) = 1\}$$

This property is called origin-hiding *of the* Merc.ChgRep *algorithm.*

(ii) Assume $\sigma \in \mathcal{S}$ with $\mathsf{Merc.Vfy}(\mathsf{pk}, m, \sigma) = 1$, $r_k \leftarrow\!\!\$\ \mathsf{Merc.KeyConvGen}(\mathsf{pp})$, $\sigma' \leftarrow\!\!\$\ \mathsf{Merc.AdaptSig}(\mathsf{pk}, m, \sigma, r_k)$, $\mathsf{pk}' := \mathsf{Merc.ConvertPK}(\mathsf{pk}, r_k)$. Then pk' is identically distributed to a uniformly random public key in $[\mathsf{pk}]_{R_{\mathsf{pk}}}$, furthermore σ' is identically distributed to a uniformly random signature in

$$\{\hat{\sigma} \mid \mathsf{Merc.Vfy}(\mathsf{pk}', m, \hat{\sigma}) = 1\}$$

This property is called origin-hiding *of the* Merc.AdaptSig *algorithm.*

Analogously to perfect adaptation of signatures for SPS-EQ, origin-hiding for mercurials is fundamentally different from message class-hiding for mercurials. This is because message class-hiding considers messages from more than one equivalence class but does not make any statement about signatures.

Furthermore, analogously to perfect SPS-EQ adaptation of signatures under maliciously generated public keys (Def. 4.27), we consider arbitrary public keys pk for Merc.ChgRep-origin-hiding (including maliciously generated ones, for which it is possible that no secret key sk exists or is known). So we require that the distribution of the output signature σ' of Merc.ChgRep must be indistinguishable from the uniform distribution on the set of valid signatures on the output message m', instead of the distribution of $\mathsf{Merc.Sign}(\mathsf{sk}, m')$. This is because the random variable $\mathsf{Merc.Sign}(\mathsf{sk}, m')$ is not defined for public keys pk for which no corresponding secret key sk exists.

4.4.1 Mercurial signature by Crites and Lysyanskaya

In this subsection, we will give the construction by Crites and Lysyanskaya [CL19] as an example for a mercurial signature. We adapted the definition from [CL19] to our flavour of mercurial signature syntax (Def. 4.41).

Definition 4.52 (**Crites-Lysyanskaya mercurial signature**) *Consider the following*

message, secret- and public key relations:

$$R_m := \{(M, M') \in (G_1 \setminus \{1_{G_1}\})^l \times (G_1 \setminus \{1_{G_1}\})^l \mid \exists r \in \mathbb{Z}_p^* : M' = M^r\}$$
$$R_{\mathsf{sk}} := \{(\mathsf{sk}, \mathsf{sk}') \in (\mathbb{Z}_p^*)^l \times (\mathbb{Z}_p^*)^l \mid \exists r \in Z_p^* : \mathsf{sk}' = \mathsf{sk}^r\}$$
$$R_m := \{(\mathsf{pk}, \mathsf{pk}') \in (G_2 \setminus \{1_{G_2}\})^l \times (G_2 \setminus \{1_{G_2}\})^l \mid \exists r \in \mathbb{Z}_p^* : \mathsf{pk}' = \mathsf{pk}^r\}$$

The Crites-Lysyanskaya mercurial signature $\Sigma^{\mathsf{CL}}_{\mathsf{Merc}}$ *is a mercurial signature over relations* R_m, R_{sk} *and* R_{pk} *defined as follows:*

Merc.PGen(λ)

1 : $\mathrm{BG} := (G_1, G_2, G_T, p, e, g_1, g_2) \leftarrow\$ \mathsf{BGGen}(\lambda, 3)$
2 : **return** $\mathsf{pp} := \mathrm{BG}$

Merc.KGen(pp, l)

1 : **for** $1 \le i \le l : x_i \leftarrow\$ \mathbb{Z}_p^*$
2 : $\quad \mathsf{sk} := (x_1, \ldots, x_l)$
3 : **for** $1 \le i \le l : X_i := g_2^{x_i}$
4 : $\quad \mathsf{pk} := (X_1, \ldots, X_l)$
5 : **return** $(\mathsf{pk}, \mathsf{sk})$

Merc.Sign(sk, m)

1 : ***parse*** $(x_1, \ldots, x_l) := \mathsf{sk}, (m_1, \ldots, m_l) := m$
2 : $y \leftarrow\$ \mathbb{Z}_p^*$
3 : $Z := (\Pi_{i=1}^l m_i^{x_i})^y$
4 : $Y_1 := g_1^{\frac{1}{y}}, Y_2 := g_2^{\frac{1}{y}}$
5 : **return** $\sigma := (Z, Y_1, Y_2)$

Merc.Vfy(pk, m, σ)

1 : ***parse*** $(X_1, \ldots, X_l) := \mathsf{pk}$,
2 : $(m_1, \ldots, m_l) := m$,
3 : $\sigma := (Z, Y_1, Y_2)$
4 : **return** 1 **if** $\Pi_{i=1}^l e(m_i, X_i) = e(Z, Y_2)$
5 : $\quad \wedge\, e(Y_1, g_2) = e(g_1, Y_2)$
6 : $\quad$ **return** 0

Merc.KeyConvGen(pp)

1 : **return** $r_k \leftarrow\$ \mathbb{Z}_p^*$

Merc.ConvertPK(pk, r_k)

1 : **return** $\mathsf{pk}' := \mathsf{pk}^{r_k}$

Merc.ConvertSK(sk, r_k)

1 : **return** $\mathsf{sk}' := \mathsf{sk}^{r_k}$

Merc.AdaptSig($\mathsf{pk}, m, \sigma, r_k$)

1 : ***parse*** $\sigma := (Z, Y_1, Y_2)$
2 : $\alpha \leftarrow\$ \mathbb{Z}_p^*$
3 : **return** $\sigma' := (Z^{r_k \cdot \alpha}, Y_1^{\frac{1}{\alpha}}, Y_2^{\frac{1}{\alpha}})$

Merc.ConvSigGen(pp)

1 : **return** $r_\sigma \leftarrow\$ \mathbb{Z}_p^*$

Merc.ChgRep($\mathsf{pk}, m, \sigma, r_\sigma$)

1 : ***parse*** $\sigma := (Z, Y_1, Y_2)$
2 : $\alpha \leftarrow\$ \mathbb{Z}_p^*$
3 : $m' := m^{r_\sigma}$
4 : $\sigma' := (Z^{\alpha \cdot r_\sigma}, Y_1^{\frac{1}{\alpha}}, Y_2^{\frac{1}{\alpha}})$
5 : **return** (m', σ')

$\mathsf{Merc.VKey}(\mathsf{pk}, \mathsf{sk})$

1 : $\quad \boldsymbol{parse}\ (X_1, \ldots, X_l) := \mathsf{pk},$

2 : $\quad (x_1, \ldots, x_l) := \mathsf{sk}$

3 : $\quad \mathbf{return}\ \bigwedge_{i=1}^{l} g_2^{x_i} = X_i$

The above mercurial signature is set in a type-3 bilinear group with groups of prime order $p \in \mathbb{P}$. For completeness, we quickly summarize the security results for the Crites-Lysyanskaya mercurial signature. The following theorem was proven in [CL19] but we omit the proof here because of scope reasons.

Theorem 4.53 *The Crites-Lysyanskaya mercurial signature is unforgeable (Def. 4.44), class-hiding (Def. 4.50) and origin-hiding (Def. 4.51) in the generic group model for type-3 bilinear groups.*

5 Relations between the signature primitives

In this chapter, we analyze the connections between the four advanced digital signature primitives from this thesis, namely key-homomorphic signatures [DS16] (Sect. 4.1), SPS-EQ [HS14, FHS14] (Sect. 4.2), SFPK [BHKS18] (Sect. 4.3) and mercurial signatures [CL19] (Sect. 4.4).

To start off, note that SFPK and SPS-EQ can be seen as complementary primitives. They both come with an an equivalence relation that divides one of their spaces into equivalence classes where a representative of a class can efficiently be randomized to another one. While SPS-EQ have such an equivalence relation on the message space, SFPK have it on the public key space. It is notable that mercurial signatures have equivalence relations on both of their message and public key space which immediately gives rise to the question whether mercurials are SPS-EQ/SFPK and whether a combination of SPS-EQ and SFPK can be used to define a mercurial signature in a black-box way. We address this question in Sect. 5.1.

At first glance, one could assume that there is a connection between SFPK (Sect. 4.3) and key-homomorphic signatures (Sect. 4.1.1) since they both provide a mechanism to randomize valid key pairs into new ones. In Sect. 5.2, we discuss what actually sets these two primitives apart. In Sect. 5.3, we furthermore examine an example SFPK (namely the warm-up SFPK scheme by Backes et al. [BHKS18]) for key-homomorphic properties.

5.1 Connection of mercurial signatures to SFPK and SPS-EQ

In this section, we analyze the relation of mercurial signatures, SFPK and SPS-EQ. First, in Sect. 5.1.1, we prove that (with some minor syntactical tweaks), every secure mercurial is a secure SPS-EQ (Thm. 5.1). Note that in [CL19], Crites and Lysyanskaya sketch such a proof which we extend in terms of covered security notions and adapt to our definitional framework from Sect. 4.2 and Sect. 4.4 in Thm. 5.1. We next prove in Thm. 5.2 that every secure mercurial signature that provides some mechanism to check whether two given public keys from its public key space E are related via the public key relation $R_{\mathsf{pk}} \subset E \times E$ is a secure SFPK. To conclude this section, in Sect. 5.1.2, we prove that an SFPK is a mercurial signature if it fulfills the following additional requirements:

- it provides an equivalence relation on its message space as well as a change-representative algorithm ChgRep in the spirit of SPS-EQ (Def. 4.19)

- it provides an adaption algorithm $\mathsf{AdaptSig}$ in the spirit of mercurial signatures (Def. 4.41)

We also examine what assumptions on the underlying primitives are required for the above signature constructions to be secure.

5.1.1 Flexible public key signatures and structure-preserving signatures on equivalence classes from mercurial signatures

In [CL19], Crites et al. informally sketched how unforgeable mercurial signatures can be seen as unforgeable SPS-EQ, i.e. that mercurials are a more powerful signature type than SPS-EQ since they also have equivalence relations on their key spaces. In the following, we will formally adapt the proof sketch from [CL19] to our framework, extending it in terms of security notions by giving a sufficient condition for the resulting SPS-EQ to be class-hiding and perfectly adapting signatures.

Theorem 5.1 *Let $l \in N$, G be a group. Let Σ_{Merc} be a mercurial signature scheme over message space $\mathcal{M} := G^l$, public key space E, secret key space H, signature space $\mathcal{S}$, message relation $R_m \subset \mathcal{M} \times \mathcal{M}$, public key relation*

$$R_{\mathsf{pk}} := \{(\mathsf{pk}, \mathsf{pk}) \mid \mathsf{pk} \in E\} \subset E \times E$$

and secret key relation $R_{\mathsf{sk}} \subset H \times H$. Furthermore let T be a set and assume that there exists a ppt algorithm $\mathsf{Merc.TPGen}$ that takes as input a security parameter $\lambda \in \mathbb{N}$ and outputs public parameters pp that are identically distributed to $\mathsf{Merc.PGen}(\lambda)$ as well as a trapdoor $\tau \in T$. Then, if Σ_{Merc} is unforgeable (Def. 4.44), message class-hiding (Def. 4.46) and $\mathsf{Merc.ChgRep}$ is origin-hiding (Def. 4.51), we have that

(i)

$$\begin{aligned}\Sigma_{\mathsf{SPSEQ}}^{(\mathsf{Merc})} = (&\mathsf{SPSEQ.PGen} := \mathsf{Merc.PGen},\\ &\mathsf{SPSEQ.TPGen} := \mathsf{Merc.TPGen},\\ &\mathsf{SPSEQ.KGen} := \mathsf{Merc.KGen},\\ &\mathsf{SPSEQ.Sign} := \mathsf{Merc.Sign},\\ &\mathsf{SPSEQ.ConvSigGen} := \mathsf{Merc.ConvSigGen},\\ &\mathsf{SPSEQ.ChgRep} := \mathsf{Merc.ChgRep},\\ &\mathsf{SPSEQ.Vfy} := \mathsf{Merc.Vfy},\\ &\mathsf{SPSEQ.VKey} := \mathsf{Merc.VKey})\end{aligned}$$

is an SPS-EQ over R_m with message space $\mathcal{M}$, public key space E, secret key space H, signature space $\mathcal{S}$ and trapdoor space T.

(ii) $\Sigma_{\mathsf{SPSEQ}}^{(\mathsf{Merc})}$ *is unforgeable (Def. 4.21).*

(iii) $\Sigma_{\mathsf{SPSEQ}}^{(\mathsf{Merc})}$ *is class-hiding (Def. 4.23).*

(iv) $\Sigma^{(\mathsf{Merc})}_{\mathsf{SPSEQ}}$ *perfectly adapts signatures (Def. 4.26) if for all messages* $m \in \mathcal{M}$, $\lambda, l \in \mathbb{N}$, $\mathsf{pp} \leftarrow\!\$\ \mathsf{Merc.PGen}(\lambda)$, $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\$\ \mathsf{Merc.KGen}(\mathsf{pp}, l)$, *we have that* σ *in* $\sigma \leftarrow\!\$\ \mathsf{Merc.Sign}(\mathsf{sk}, m)$ *is an uniformly random valid signature for* m *under* pk, *i.e. a uniformly random signature in*

$$\{\hat{\sigma} \in \mathcal{S} \mid \mathsf{Merc.Vfy}(\mathsf{pk}, m, \hat{\sigma}) = 1\}$$

Proof. **ad (i):** It follows from inspection that $\Sigma^{(\mathsf{Merc})}_{\mathsf{SPSEQ}}$ fulfills the syntax definition of SPS-EQ (Def. 4.19). What is left to prove is that $\Sigma^{(\mathsf{Merc})}_{\mathsf{SPSEQ}}$ is a correct SPS-EQ according to Def. 4.20. Since Σ_{Merc} is a correct mercurial signature, $\mathsf{Merc.KGen}$ only generates valid key pairs and thus $\Sigma^{(\mathsf{Merc})}_{\mathsf{SPSEQ}}$ fulfills the first correctness requirement for SPS-EQ. By prerequisite, $\mathsf{Merc.PGen}(\lambda)$ and $\mathsf{Merc.TPGen}(\lambda)$ have the same output distribution, so $\Sigma^{(\mathsf{Merc})}_{\mathsf{SPSEQ}}$ also fulfills the second correctness requirement for SPS-EQ.

To prove the third correctness requirement, let $\lambda \in \mathbb{N}$, $\mathsf{pp} \leftarrow\!\$\ \mathsf{Merc.PGen}(\lambda)$, $\mathsf{Merc.VKey}(\mathsf{pk}, \mathsf{sk}) = 1$, $m \in \mathcal{M}$, $\alpha \leftarrow\!\$\ \mathsf{Merc.ConvSigGen}(\mathsf{pp})$ and $\sigma \leftarrow\!\$\ \mathsf{Merc.Sign}(\mathsf{sk}, m)$. Then

$$\mathsf{Merc.Vfy}(\mathsf{pk}, m, \sigma) = 1$$

and

$$\mathsf{Merc.Vfy}(\mathsf{pk}, m', \sigma') = 1 \wedge m' \in [m]_R$$

where $(m', \sigma') \leftarrow\!\$\ \mathsf{Merc.ChgRep}(\mathsf{pk}, m, \mathsf{Merc.Sign}(\mathsf{sk}, m), \alpha)$ since Σ_{Merc} is a correct mercurial signature scheme.

So all in all, $\Sigma^{(\mathsf{Merc})}_{\mathsf{SPSEQ}}$ is a correct SPS-EQ (Def. 4.20).

ad (ii): Let $\mathcal{A}$ be a ppt adversary, $\lambda \in \mathbb{N}$ be the security parameter. We have

$$\begin{aligned}\mathsf{Adv}^{\Sigma_{\mathsf{spseq\text{-}euf}}}_{\mathcal{A},\Sigma^{(\mathsf{Merc})}_{\mathsf{SPSEQ}}}(\lambda) &= \Pr[\mathsf{Exp}^{\Sigma_{\mathsf{spseq\text{-}euf}}}_{\mathcal{A},\Sigma^{(\mathsf{Merc})}_{\mathsf{SPSEQ}}}(\lambda) = 1] \\ &= \Pr[\mathsf{Exp}^{\Sigma_{\mathsf{merc\text{-}euf}}}_{\mathcal{A},\Sigma_{\mathsf{Merc}}}(\lambda) = 1] \\ &= \mathsf{Adv}^{\Sigma_{\mathsf{merc\text{-}euf}}}_{\mathcal{A},\Sigma_{\mathsf{Merc}}}(\lambda)\end{aligned}$$

negligible which proves that $\Sigma^{(\mathsf{Merc})}_{\mathsf{SPSEQ}}$ is an unforgeable SPS-EQ according to Def. 4.21. The second equality comes from the fact that since

$$R_{\mathsf{pk}} = \{(e, e) \mid e \in E\}$$

we have

$$\mathsf{pk}^* \in [\mathsf{pk}] \Leftrightarrow \mathsf{pk}^* = \mathsf{pk}$$

for all $(\mathsf{pk}, \mathsf{pk}^*) \in E \times E$, which yields that the winning conditions of $\mathsf{Exp}^{\mathsf{spseq\text{-}euf}}_{\mathcal{A},\Sigma^{(\mathsf{Merc})}_{\mathsf{SPSEQ}}}(\lambda)$ and $\mathsf{Exp}^{\mathsf{merc\text{-}euf}}_{\mathcal{A},\Sigma_{\mathsf{Merc}}}(\lambda)$ are equivalent.

ad (iii): Let $\mathcal{A}$ be a ppt adversary, $\lambda \in \mathbb{N}$ the security parameter. Observe that $\mathsf{Exp}^{\mathsf{merc\text{-}mes\text{-}ch}}_{\mathcal{A},\Sigma_{\mathsf{Merc}}}(\lambda)$ from Def. 4.46 and $\mathsf{Exp}^{\mathsf{spseq\text{-}ch}}_{\mathcal{A},\Sigma^{(\mathsf{Merc})}_{\mathsf{SPSEQ}}}(\lambda)$ from Def. 4.23 proceed identically. So we get that

$$\begin{aligned}\mathsf{Adv}^{\mathsf{spseq\text{-}ch}}_{\mathcal{A},\Sigma^{(\mathsf{Merc})}_{\mathsf{SPSEQ}}}(\lambda) &= \Pr[\mathsf{Exp}^{\mathsf{spseq\text{-}ch}}_{\mathcal{A},\Sigma^{(\mathsf{Merc})}_{\mathsf{SPSEQ}}}(\lambda) = 1] \\ &= \Pr[\mathsf{Exp}^{\mathsf{merc\text{-}mes\text{-}ch}}_{\mathcal{A},\Sigma_{\mathsf{Merc}}}(\lambda) = 1] = 1 \\ &= \mathsf{Adv}^{\mathsf{merc\text{-}mes\text{-}ch}}_{\mathcal{A},\Sigma_{\mathsf{Merc}}}(\lambda)\end{aligned}$$

is negligible which proves that $\Sigma^{(\mathsf{Merc})}_{\mathsf{SPSEQ}}$ is class-hiding according to Def. 4.23.

ad (iv) We need to show the identity of the two distributions from Def. 4.26 for $\Sigma^{(\mathsf{Merc})}_{\mathsf{SPSEQ}}$. Let

$$(\mathsf{pk}, \mathsf{sk}, m, \sigma, \alpha) \in E \times H \times \mathcal{M} \times \mathcal{S} \times [\mathsf{Merc.ConvSigGen}(\mathsf{pp})]$$

with $\mathsf{Merc.VKey}(\mathsf{pk}, \mathsf{sk}) = 1$, $\mathsf{Merc.Vfy}(\mathsf{pk}, m, \sigma) = 1$, $\mathsf{pp} \leftarrow\!\!\$\ \mathsf{Merc.PGen}(\lambda)$, $\alpha \leftarrow\!\!\$\ \mathsf{Merc.ConvSigGen}(\mathsf{pp})$ as in Def. 4.26.

Let $(m', \sigma') \leftarrow\!\!\$\ \mathsf{Merc.ChgRep}(\mathsf{pk}, m, \sigma, \alpha)$. By origin-hiding of Σ_{Merc} we have that m' is distributed like $m^* \leftarrow\!\!\$\ [m]_{R_m}$ and σ' is distributed like a uniformly random valid signature σ^* for m', i.e.

$$\sigma^* \leftarrow\!\!\$\ \{\hat{\sigma} \mid \mathsf{Merc.Vfy}(\mathsf{pk}, m', \hat{\sigma})\}$$

Using our assumption on the distribution of the signatures output by $\mathsf{Merc.Sign}$, we get that σ' and $\mathsf{Merc.Sign}(\mathsf{sk}, m')$ are identically distributed. Since obviously $m' \in [m]_{R_m}$, we have proven the required identity of distributions and thus that $\Sigma^{(\mathsf{Merc})}_{\mathsf{SPSEQ}}$ perfectly adapts signatures according to Def. 4.26.

□

So informally, the above theorem states that, if it has a suitable message space, an unforgeable, message class-hiding mercurial signature scheme with an origin-hiding change-representative algorithm canonically defines an unforgeable and class-hiding SPS-EQ. The resulting SPS-EQ perfectly adapts signatures if the $\mathsf{Merc.Sign}$ algorithm outputs uniformly random valid signatures on the input message. An example for a mercurial signature scheme with such a signing algorithm is the construction by Crites and Lysyanskaya from [CL19] which we examined in Sect. 4.4.1. Note that with an analogous argumentation as in Thm. 5.1, one could prove that origin-hiding of the $\mathsf{Merc.ChgRep}$ algorithm of Σ_{Merc} implies perfect adaptation of signatures of $\Sigma^{(\mathsf{Merc})}_{\mathsf{SPSEQ}}$ under malicious public keys (Def. 4.27).

As a closing remark for Thm. 5.1, note that obviously not all SPS-EQ are mercurial signatures since they lack a public key relation that satisfies the correctness requirements for mercurial signatures from Def. 4.42. However, there exist SPS-EQ that can be extended to mercurial signatures by defining proper $\mathsf{Merc.ConvertPK}$, $\mathsf{Merc.ConvertSK}$,

Merc.KeyConvGen and Merc.AdaptSig algorithms. An example of such an SPS-EQ is the construction by Connolly et al. from [CLPK22].

Next, we prove that analogously, every secure mercurial signature is a secure SFPK (see Sect. 4.3) if it has a mechanism to check class membership that is similar to the SFPK.ChkRep algorithm. To our knowledge, this connection has not been made before this work.

Theorem 5.2 *Let $\lambda \in \mathbb{N}$ be the security parameter, let Σ_{Merc} be a mercurial signature scheme over message space $\mathcal{M}$, public key space E, secret key space H, signature space $\mathcal{S}$, message relation*

$$R_m = \{(m, m) \mid m \in \mathcal{M}\}$$

public key relation $R_{\mathsf{pk}} \subseteq E \times E$ and secret key relation $R_{\mathsf{sk}} \subseteq H \times H$. Furthermore let T be a set and assume that there exist additional ppt algorithms as follows:

Merc.TKGen *probablistic algorithm, takes as input public parameters* pp *and outputs a key pair* $(\mathsf{pk}, \mathsf{sk}) \in E \times H$ *(that is identically distributed as* Merc.KGen(pp)*) and a trapdoor $\tau \in T$ for the class $[\mathsf{pk}]_{R_{\mathsf{pk}}}$*

Merc.ChkRep *deterministic algorithm, takes as input a public key $\mathsf{pk} \in E$ and a trapdoor $\tau \in T$ for public key $\mathsf{pk}' \in E$ and outputs a bit $b \in \{0, 1\}$*

$$b = 1 \Leftrightarrow \mathsf{pk} \in [\mathsf{pk}']_{R_{\mathsf{pk}}}$$

Then, if Σ_{Merc} is existentially unforgeable (Def. 4.45) and real-or-random public-key class-hiding (Def. 4.47), we have that

(i)

$$\begin{aligned}\Sigma_{\mathsf{SFPK}}^{(\mathsf{Merc})} = (&\mathsf{SFPK.PGen} := \mathsf{Merc.PGen},\\ &\mathsf{SFPK.KGen} := \mathsf{Merc.KGen},\\ &\mathsf{SFPK.TKGen} := \mathsf{Merc.TKGen},\\ &\mathsf{SFPK.Sign} := \mathsf{Merc.Sign},\\ &\mathsf{SFPK.KeyConvGen} := \mathsf{Merc.KeyConvGen},\\ &\mathsf{SFPK.ChkRep} := \mathsf{Merc.ChkRep},\\ &\mathsf{SFPK.ChgPK} := \mathsf{Merc.ConvertPK},\\ &\mathsf{SFPK.ChgSK} := \mathsf{Merc.ConvertSK},\\ &\mathsf{SFPK.Vfy} := \mathsf{Merc.Vfy},\\ &\mathsf{SFPK.VKey} := \mathsf{Merc.VKey})\end{aligned}$$

is an SFPK over equivalence relation R_{pk}, message space $\mathcal{M}$, public key space E, secret key space H, signature space $\mathcal{S}$ and trapdoor space T.

(ii) $\Sigma_{\mathsf{SFPK}}^{(\mathsf{Merc})}$ is unforgeable (Def. 4.30).

(iii) $\Sigma_{\mathsf{SFPK}}^{(\mathsf{Merc})}$ *is adaptively real-or-random class-hiding without key corruption (Def. 4.38).*

Proof. **ad (i):** It follows from inspection that $\Sigma_{\mathsf{SFPK}}^{(\mathsf{Merc})}$ fulfills the syntax requirements of an SFPK. Note that whether one explicitly passes the key generation randomness ω from Def. 4.28 to the key generation algorithm or not is not an actual syntactical difference.

What is left to prove is that $\Sigma_{\mathsf{SFPK}}^{(\mathsf{Merc})}$ is a correct SFPK scheme according to Def. 4.29. By prerequisite, $\Sigma_{\mathsf{SFPK}}^{(\mathsf{Merc})}$ fulfills the first and third security requirements since the output distributions of $\mathsf{Merc.KGen}$ and $\mathsf{Merc.TKGen}$ are identical, furthermore $\mathsf{Merc.ChkRep}(\tau, \mathsf{pk})$ outputs 1 if and only if $\mathsf{pk} \in [\mathsf{pk}']_{R_{\mathsf{pk}}}$ where τ is a trapdoor for the class $[\mathsf{pk}']$.

Because Σ_{Merc} is a correct mercurial signature, $\Sigma_{\mathsf{SFPK}}^{(\mathsf{Merc})}$ obviously fulfills the second and fourth requirement of Def. 4.29 since Σ_{Merc} fulfills the third requirement of Def. 4.42.

So $\Sigma_{\mathsf{SFPK}}^{(\mathsf{Merc})}$ is a correct SFPK according to Def. 4.29.

ad (ii): Let $\mathcal{A}$ be a ppt adversary, $\lambda \in \mathbb{N}$ be the security parameter. We have

$$\begin{aligned}\mathsf{Adv}^{\mathsf{sfpk\text{-}euf}}_{\mathcal{A},\Sigma_{\mathsf{SFPK}}^{(\mathsf{Merc})}}(\lambda) &= \Pr[\mathsf{Adv}^{\mathsf{sfpk\text{-}euf}}_{\mathcal{A},\Sigma_{\mathsf{SFPK}}^{(\mathsf{Merc})}}(\lambda) = 1] \\ &= \Pr[\mathsf{Adv}^{\mathsf{merc\text{-}euf}}_{\mathcal{A},\Sigma_{\mathsf{Merc}}}(\lambda)] \\ &= \mathsf{Adv}^{\mathsf{merc\text{-}euf}}_{\mathcal{A},\Sigma_{\mathsf{Merc}}}(\lambda)\end{aligned}$$

negligible which proves $\Sigma_{\mathsf{SFPK}}^{(\mathsf{Merc})}$ is an unforgeable SFPK according to Def. 4.30. The second equality comes from the fact that since

$$R_m = \{(m, m) \mid m \in \mathcal{M}\}$$

we have

$$m^* \notin [m]_{R_m} \Leftrightarrow m^* \neq m$$

for all $(m, m^*) \in \mathcal{M} \times \mathcal{M}$ which yields that the winning conditions of $\mathsf{Exp}^{\mathsf{sfpk\text{-}euf}}_{\mathcal{A},\Sigma_{\mathsf{SFPK}}^{(\mathsf{Merc})}}(\lambda)$ (Def. 4.30) and $\mathsf{Exp}^{\mathsf{merc\text{-}euf}}_{\mathcal{A},\Sigma_{\mathsf{Merc}}}(\lambda)$ (Def. 4.45) are equivalent.

ad (iii): Let $\mathcal{A}$ be a ppt adversary, $\lambda \in \mathbb{N}$ be the security parameter. We have

$$\begin{aligned}\mathsf{Adv}^{\mathsf{adap\text{-}ror\text{-}ch\text{-}nkc}}_{\mathcal{A},\Sigma_{\mathsf{SFPK}}^{(\mathsf{Merc})}}(\lambda) &= \Pr[\mathsf{Exp}^{\mathsf{adap\text{-}ror\text{-}ch\text{-}nkc}}_{\mathcal{A},\Sigma_{\mathsf{SFPK}}^{(\mathsf{Merc})}}(\lambda) = 1] \\ &= \Pr[\mathsf{Exp}^{\mathsf{merc\text{-}pk\text{-}ch}}_{\mathcal{A},\Sigma_{\mathsf{Merc}}}(\lambda) = 1] \\ &= \mathsf{Adv}^{\mathsf{merc\text{-}pk\text{-}ch}}_{\mathcal{A},\Sigma_{\mathsf{Merc}}}(\lambda)\end{aligned}$$

not negligible where the second equality comes from the fact that by definition of $\Sigma_{\mathsf{SFPK}}^{(\mathsf{Merc})}$, the games $\mathsf{Exp}^{\mathsf{adap\text{-}ror\text{-}ch\text{-}nkc}}_{\mathcal{A},\Sigma_{\mathsf{SFPK}}^{(\mathsf{Merc})}}(\lambda)$ (Def. 4.38) and $\mathsf{Exp}^{\mathsf{merc\text{-}pk\text{-}ch}}_{\mathcal{A},\Sigma_{\mathsf{Merc}}}(\lambda)$ (Def. 4.47)

proceed identically. So by Def. 4.38, $\Sigma_{\mathsf{SFPK}}^{(\mathsf{Merc})}$ is adaptively real-or-random class-hiding without key corruption.

□

Note that we used the mercurial signature public-key class-hiding definition from the literature [CL19] here, which does not consider adversaries with any insights to the key generation process. So the resulting SFPK $\Sigma_{\mathsf{SFPK}}^{(\mathsf{Merc})}$ only fulfills a class-hiding definition from the weakest tier, namely adaptive real-or-random class-hiding without key corruption. For an overview of the different tiers of SFPK class-hiding definitions see Lem. 4.39.

5.1.2 Mercurial signature from flexible public key signatures and structure-preserving signatures on equivalence classes

In this subsection we examine under which assumptions mercurial signatures can be constructed from a combination of SFPK and SPS-EQ in a black-box way. This question is the natural counterpart of the previous Sect. 5.1.1 where we have proven that, with some additional assumptions, mercurial signatures can be seen as both SFPK and SPS-EQ. We examine what is the exact gap between the SFPK/SPS-EQ combination and a mercurial signature. We begin with the black-box construction of a mercurial signature from SFPK and SPS-EQ.

Theorem 5.3 *Let $E, H, \mathcal{M}, \mathcal{S}$ be sets, let Σ_{SPSEQ} be an SPS-EQ, Σ_{SFPK} be an SFPK (both over the same message space $\mathcal{M}$, public key space E, secret key space H and signature space $\mathcal{S}$). Let Σ_{SPSEQ} and Σ_{SFPK} fulfill the following requirements:*

(i) $\mathsf{SPSEQ.PGen}(\lambda)$ *and* $\mathsf{SFPK.PGen}(\lambda)$ *have identical supports for all* $\lambda \in \mathbb{N}$.

(ii) $\mathsf{SPSEQ.KGen}(\mathsf{pp}, l)$ *and* $\mathsf{SFPK.KGen}(\mathsf{pp}, \cdot)$ *have identical supports for all security parameters* $\lambda \in \mathbb{N}$, $\mathsf{pp} \leftarrow\!\!\$\ \mathsf{SPSEQ.PGen}(\lambda)$ *and length parameters* $l \in \mathbb{N}$.

(iii) $\mathsf{SPSEQ.Sign}(\mathsf{sk}, m)$ *and* $\mathsf{SFPK.Sign}(\mathsf{sk}, m)$ *have identical supports for all secret keys* $\mathsf{sk} \in H$ *and messages* $m \in \mathcal{M}$.

(iv) $\mathsf{SPSEQ.Vfy}(\mathsf{pk}, m, \sigma) = 1 \Leftrightarrow \mathsf{SFPK.Vfy}(\mathsf{pk}, m, \sigma) = 1$ *for all* $\mathsf{pk} \in E$, $m \in \mathcal{M}$ *and* $\sigma \in \mathcal{S}$

(v) $\mathsf{SPSEQ.VKey}(\mathsf{pk}, \mathsf{sk}) = 1 \Leftrightarrow \mathsf{SFPK.VKey}(\mathsf{pk}, \mathsf{sk}) = 1$ *for all key pairs* $(\mathsf{pk}, \mathsf{sk}) \in E \times H$

(vi) There is a ppt algorithm AdaptSig *with the following syntax and properties:*

$\mathsf{AdaptSig}(\mathsf{pk}, m, \sigma, r)$ *probablistic signature conversion algorithm, takes as input a public key* $\mathsf{pk} \in E$, *a message* $m \in \mathcal{M}$, *a signature* $\sigma \in \mathcal{S}$ *and key conversion randomness* r *for public parameters* pp *and outputs a signature* $\sigma' \in \mathcal{S}$

such that for all public parameters pp, $(\mathsf{pk}, \cdot) \leftarrow\!\$\ \mathsf{SFPK.KGen}(\mathsf{pp}), m \in \mathcal{M}$, $\sigma \in \mathcal{S}$ *with* $\mathsf{SFPK.Vfy}(\mathsf{pk}, m, \sigma) = 1$, $r \leftarrow\!\$\ \mathsf{SFPK.KeyConvGen}(\mathsf{pp})$, $\sigma' \leftarrow\!\$\ \mathsf{AdaptSig}(\mathsf{pk}, m, \sigma, r)$, $\mathsf{pk}' := \mathsf{SFPK.ChgPK}(\mathsf{pk}, r)$ *we have*

$$\mathsf{SFPK.Vfy}(\mathsf{pk}', m, \sigma') = 1$$

Then

$$\begin{aligned}\Sigma^*_{\mathsf{Merc}} := (&\mathsf{Merc.PGen} := \mathsf{SPSEQ.PGen},\\ &\mathsf{Merc.KGen} := \mathsf{SPSEQ.KGen},\\ &\mathsf{Merc.Sign} := \mathsf{SFPK.Sign},\\ &\mathsf{Merc.Vfy} := \mathsf{SFPK.Vfy},\\ &\mathsf{Merc.KeyConvGen} := \mathsf{SFPK.KeyConvGen},\\ &\mathsf{Merc.ConvertPK} := \mathsf{SFPK.ChgPK},\\ &\mathsf{Merc.ConvertSK} := \mathsf{SFPK.ChgSK},\\ &\mathsf{Merc.AdaptSig} := \mathsf{AdaptSig},\\ &\mathsf{Merc.ConvSigGen} := \mathsf{SPSEQ.ConvSigGen},\\ &\mathsf{Merc.ChgRep} := \mathsf{SPSEQ.ChgRep},\\ &\mathsf{Merc.VKey} := \mathsf{SFPK.VKey})\end{aligned}$$

is a correct mercurial signature scheme.

Proof. It follows from inspection that Σ^*_{Merc} fulfills the syntax requirements of a mercurial signature scheme. What is left to prove is that Σ^*_{Merc} is a correct mercurial signature scheme according to Def. 4.42.

Note that since the parameter generation algorithms of Σ_{SFPK} and Σ_{SPSEQ} have identical supports for all λ, we can use any parameters $\mathsf{pp} \leftarrow\!\$\ \mathsf{SFPK.KGen}(\lambda)$ as input for $\mathsf{SPSEQ.KGen}$ and vice versa (analogously for key generation and signing algorithms) and still obtain the same correctness guarantees.

Since the underlying SPS-EQ Σ_{SPSEQ} and SFPK Σ_{SFPK} are correct (according to Def. 4.20 and Def. 4.29, respectively), Σ^*_{Merc} obviously fulfills the first two correctness requirements from Def. 4.42 which are validity of honestly generated key pairs and basic digital signature correctness. Because Σ_{SFPK} is a correct SFPK (Def. 4.29), Σ^*_{Merc} fulfills the third correctness requirement for mercurial signatures since adapted key pairs stay valid. Σ^*_{Merc} fulfills the fourth security requirement for mercurial signatures by the prerequisites for $\mathsf{AdaptSig}$ (which requires that all consistently adapted valid key-message-signature triples stay valid). The fifth and last correctness requirement for mercurial signatures is fulfilled by Σ^*_{Merc} because the underlying SPS-EQ is correct according to Def. 4.20 (so changing the representative of a message class and adapting the signature accordingly preserves the validity of a key-message-signature triple).

So all in all, Σ^*_{Merc} is a correct mercurial signature scheme. □

Note that the above construction from Thm. 5.3 makes very strong assumptions about the underlying SFPK and SPS-EQ. Equivalently speaking, it is a strong assumption that an SFPK allows to adapt signatures to both messages and public keys in a public way and also provides an SPS-EQ-style ChgRep algorithm. It is unclear whether meaningful secure SFPK with these properties exist. A public adaption of signatures in the style of $\mathsf{AdaptSig}$ as mentioned above is not possible for general SFPK, even if they have recoverable signing keys (Def. 4.40). To see this, let Σ_{SFPK} be an SFPK. Furthermore let $(\mathsf{pk}, \mathsf{sk}, \tau) \leftarrow\!\!\$\ \mathsf{SFPK.TKGen}(\mathsf{pp})$ for some public parameters pp, $\sigma \leftarrow\!\!\$\ \mathsf{SFPK.Sign}(\mathsf{sk}, m)$ for some message m, $\mathsf{pk}' \leftarrow\!\!\$\ \mathsf{SFPK.ChgPK}(\mathsf{pk}, r)$ with randomness $r \leftarrow\!\!\$\ \mathsf{SFPK.KeyConvGen}(\mathsf{pp})$. If Σ_{SFPK} has recoverable signing keys, we can recover the signing key sk' for pk' as

$$\mathsf{sk}' := \mathsf{SFPK.Recover}(\mathsf{sk}, \tau, \mathsf{pk}')$$

according to Def. 4.40. This however is not a public recovery mechanism since the required trapdoor τ for the key pair $(\mathsf{pk}, \mathsf{sk})$ is not intended to be published. Knowing τ provides an adversary with a trivial way to decide whether a given public key pk^* is related to pk (i.e. $\mathsf{pk}^* \in [\mathsf{pk}]$), contradicting class-hiding. An alternative way to adapt (m, σ) to pk' would be to adapt the secret key sk corresponding to pk as $\mathsf{sk}' := \mathsf{SFPK.ChgSK}(\mathsf{sk}, r)$ which obviously requires knowledge of the secret key sk and thus also is no public adaption procedure.

Next we analyze the security of the mercurial signature Σ^*_{Merc} from Thm. 5.3, starting with unforgeability. Σ^*_{Merc} is unforgeable if the underlying flexible public key signature Σ_{SFPK} is unforgeable. While this might look surprising at first glance, it makes sense if one observes that all algorithms of Σ^*_{Merc} that are used in the mercurial unforgeability game (Def. 4.44) come from Σ_{SFPK} or their respective equivalents in Σ_{SPSEQ} are assumed to have identical output distributions.

Theorem 5.4 *Let Σ_{SFPK}, Σ_{SPSEQ} and Σ^*_{Merc} as in Thm. 5.3. Additionally assume*

(i) $\mathsf{SPSEQ.PGen}(\lambda)$ *and* $\mathsf{SFPK.PGen}(\lambda)$ *are identically distributed for all $\lambda \in \mathbb{N}$.*

(ii) $\mathsf{SPSEQ.KGen}(\mathsf{pp}, l)$ *and* $\mathsf{SFPK.KGen}(\mathsf{pp}, \cdot)$ *are identically distributed for all security parameters $\lambda \in \mathbb{N}$,* $\mathsf{pp} \leftarrow\!\!\$\ \mathsf{SPSEQ.PGen}(\lambda)$ *and length parameters $l \in \mathbb{N}$.*

(iii) $\mathsf{SPSEQ.Sign}(\mathsf{sk}, m)$ *and* $\mathsf{SFPK.Sign}(\mathsf{sk}, m)$ *are identically distributed for all secret keys $\mathsf{sk} \in H$ and messages $m \in \mathcal{M}$.*

(iv) Σ_{SFPK} *is existentially unforgeable under chosen-message attacks (see Def. 4.30)*

Then Σ^*_{Merc} is existentially unforgeable under chosen-message attacks according to Def. 4.44.

Proof. We prove the theorem by contraposition, i.e. assume $\mathcal{A}$ is a ppt adversary with $\mathsf{Adv}^{\mathsf{merc\text{-}euf}}_{\mathcal{A},\Sigma^*_{\mathsf{Merc}}}(\lambda)$. We construct a ppt adversary $\mathcal{B}$ with $\mathsf{Adv}^{\mathsf{sfpk\text{-}euf}}_{\mathcal{B},\Sigma_{\mathsf{SFPK}}}(\lambda)$ not negligible.

Let $\omega \leftarrow\!\!\$\ \{0,1\}^{\mathsf{poly}(\lambda)}$, $\lambda \in \mathbb{N}$, $\mathsf{pp} \leftarrow\!\!\$\ \mathsf{SFPK.PGen}(\lambda)$, $(\mathsf{pk}, \mathsf{sk}, \tau) \leftarrow\!\!\$\ \mathsf{SFPK.TKGen}(\mathsf{pp}, \omega)$. On input $(\mathsf{pp}, \mathsf{pk}, \tau)$, $\mathcal{B}$ acts as follows:

1. $\mathcal{B}$ runs $\mathcal{A}(\mathsf{pp}, \mathsf{pk})$.
   a) Whenever $\mathcal{A}$ queries a signature under sk for message $m \in \mathcal{M}$, $\mathcal{B}$ relays the query m to its own signing oracle and sends the answer back to $\mathcal{A}$.
2. $\mathcal{A}$ eventually outputs a forgery $(\mathsf{pk}^*, m^*, \sigma^*)$.
3. $\mathcal{B}$ outputs $(\mathsf{pk}^*, m^*, \sigma^*)$.

$\mathcal{B}$ obviously is a ppt since $\mathcal{A}$ is a ppt. By definition of Σ^*_{Merc} and the fact that

- $\mathsf{SFPK.PGen}(\lambda)$ and $\mathsf{SPSEQ.PGen}(\lambda)$ are identically distributed
- $\mathsf{SFPK.TKGen}(\mathsf{pp})$ and $\mathsf{SPSEQ.KGen}(\mathsf{pp}, l)$ are identically distributed (by prerequisite and since Σ_{SFPK} is correct)

we get that pk is distributed as prescribed by the unforgeability game $\mathsf{Exp}^{\mathsf{merc\text{-}euf}}_{\mathcal{A},\Sigma^*_{\mathsf{Merc}}}(\lambda)$.

What is left to do is analyzing the advantage of $\mathcal{B}$ in the unforgeability game $\mathsf{Exp}^{\mathsf{sfpk\text{-}euf}}_{\mathcal{B},\Sigma_{\mathsf{SFPK}}}(\lambda)$ for Σ_{SFPK}. We see that

$$\begin{aligned}\mathsf{Adv}^{\mathsf{sfpk\text{-}euf}}_{\mathcal{B},\Sigma_{\mathsf{SFPK}}}(\lambda) &= \Pr[\mathsf{Exp}^{\mathsf{sfpk\text{-}euf}}_{\mathcal{B},\Sigma_{\mathsf{SFPK}}}(\lambda) = 1] \\ &\geq \Pr[\mathsf{Exp}^{\mathsf{merc\text{-}euf}}_{\mathcal{A},\Sigma_{\mathsf{Merc}}}(\lambda) = 1] \\ &= \mathsf{Adv}^{\mathsf{merc\text{-}euf}}_{\mathcal{A},\Sigma_{\mathsf{Merc}}}(\lambda)\end{aligned}$$

where the inequality comes from the fact that the winning condition of $\mathsf{Exp}^{\mathsf{merc\text{-}euf}}_{\mathcal{A},\Sigma_{\mathsf{Merc}}}(\lambda)$ (Def. 4.44) implies the winning condition of $\mathsf{Exp}^{\mathsf{sfpk\text{-}euf}}_{\mathcal{B},\Sigma_{\mathsf{SFPK}}}(\lambda)$ (Def. 4.30). This is because

$$\mathsf{SFPK.ChkRep}(\mathsf{pk}^*, \tau) \Leftrightarrow \mathsf{pk}^* \in [\mathsf{pk}]_{R_{\mathsf{pk}}}$$

since Σ_{SFPK} is correct and because

$$(\forall m \in Q : m^* \notin [m]_{R_m}) \Rightarrow m^* \notin Q$$

This proves the theorem. □

Being done with unforgeability, we continue with analyzing the class-hiding properties of Σ^*_{Merc} (according to Def. 4.50, namely message class-hiding (Def. 4.46) and public-key class-hiding (Def. 4.47)). We see that the message class-hiding of Σ^*_{Merc} comes from the class-hiding property of the underlying SPS-EQ and the public-key class-hiding comes from a suitable class-hiding property of the underlying SFPK.

Theorem 5.5 *Let Σ_{SFPK}, Σ_{SPSEQ} and Σ^*_{Merc} as in Thm. 5.3. Furthermore assume that*

(i) $\mathsf{SPSEQ.PGen}(\lambda)$ *and* $\mathsf{SFPK.PGen}(\lambda)$ *are identically distributed for all* $\lambda \in \mathbb{N}$.

(ii) $\mathsf{SPSEQ.KGen}(\mathsf{pp}, l)$ *and* $\mathsf{SFPK.KGen}(\mathsf{pp}, \cdot)$ *are identically distributed for all security parameters* $\lambda \in \mathbb{N}$, $\mathsf{pp} \leftarrow\!\!\$\ \mathsf{SPSEQ.PGen}(\lambda)$ *and length parameters* $l \in \mathbb{N}$.

(iii) $\mathsf{SPSEQ.Sign}(\mathsf{sk}, m)$ *and* $\mathsf{SFPK.Sign}(\mathsf{sk}, m)$ *are identically distributed for all secret keys* $\mathsf{sk} \in H$ *and messages* $m \in \mathcal{M}$.

(iv) Σ_{SFPK} *is adaptively real-or-random class-hiding without key corruption (Def. 4.38).*

(v) Σ_{SPSEQ} *is message class-hiding (see Def. 4.23).*

Then Σ^*_{Merc} *is class-hiding according to Def. 4.50.*

Proof. We need to prove that Σ^*_{Merc} is message class-hiding (Def. 4.46) and public-key class-hiding (Def. 4.47).

We start with message class-hiding. Let $\mathcal{A}$ be a ppt adversary. We have

$$\begin{aligned}\mathsf{Adv}^{\mathsf{merc\text{-}mes\text{-}ch}}_{\mathcal{A},\Sigma_{\mathsf{Merc}}}(\lambda) &= \Pr[\mathsf{Exp}^{\mathsf{merc\text{-}mes\text{-}ch}}_{\mathcal{A},\Sigma_{\mathsf{Merc}}}(\lambda) = 1] \\ &= \Pr[\mathsf{Exp}^{\mathsf{spseq\text{-}ch}}_{\mathcal{A},\Sigma_{\mathsf{SPSEQ}}}(\lambda) = 1] \\ &= \mathsf{Adv}^{\mathsf{spseq\text{-}ch}}_{\mathcal{A},\Sigma_{\mathsf{SPSEQ}}}(\lambda)\end{aligned}$$

where the second equality comes from the fact that the by prerequisite (i) and the definition of the security games, the inputs for $\mathcal{A}$ in $\mathsf{Exp}^{\mathsf{merc\text{-}mes\text{-}ch}}_{\mathcal{A},\Sigma_{\mathsf{Merc}}}(\lambda)$ and $\mathsf{Exp}^{\mathsf{spseq\text{-}ch}}_{\mathcal{A},\Sigma_{\mathsf{SPSEQ}}}(\lambda)$ are identically distributed. So since Σ_{SPSEQ} is class-hiding according to Def. 4.23, we get that Σ^*_{Merc} is message class-hiding according to Def. 4.46.

What is left to prove is that Σ^*_{Merc} is public-key class-hiding according to Def. 4.47. Let $\mathcal{A}$ again be a ppt adversary. We have

$$\Pr[\mathsf{Exp}^{\mathsf{merc\text{-}ror\text{-}pk\text{-}ch}}_{\mathcal{A},\Sigma^*_{\mathsf{Merc}}}(\lambda) = 1] = \Pr[\mathsf{Exp}^{\mathsf{adap\text{-}fo\text{-}ch\text{-}nkc}}_{\mathcal{A},\Sigma_{\mathsf{SFPK}}}(\lambda) = 1]$$

since by prerequisites (i) and (ii) and the definition of the security games, the views for $\mathcal{A}$ in $\mathsf{Exp}^{\mathsf{merc\text{-}ror\text{-}pk\text{-}ch}}_{\mathcal{A},\Sigma^*_{\mathsf{Merc}}}(\lambda)$ and $\mathsf{Exp}^{\mathsf{adap\text{-}ror\text{-}ch\text{-}nkc}}_{\mathcal{A},\Sigma_{\mathsf{SFPK}}}(\lambda)$ are identically distributed. This yields

$$\begin{aligned}\mathsf{Adv}^{\mathsf{merc\text{-}ror\text{-}pk\text{-}ch}}_{\mathcal{A},\Sigma^*_{\mathsf{Merc}}}(\lambda) &= |\Pr[\mathsf{Exp}^{\mathsf{merc\text{-}ror\text{-}pk\text{-}ch}}_{\mathcal{A},\Sigma^*_{\mathsf{Merc}}}(\lambda) = 1] - \frac{1}{2}| \\ &= |\Pr[\mathsf{Exp}^{\mathsf{adap\text{-}fo\text{-}ch\text{-}nkc}}_{\mathcal{A},\Sigma_{\mathsf{SFPK}}}(\lambda) = 1] - \frac{1}{2}| \\ &= \mathsf{Adv}^{\mathsf{adap\text{-}fo\text{-}ch\text{-}nkc}}_{\mathcal{A},\Sigma_{\mathsf{SFPK}}}(\lambda)\end{aligned}$$

So since Σ_{SFPK} is adaptively real-or-random class-hiding according to Def. 4.38, Σ^*_{Merc} is (real-or-random) public-key class-hiding according to Def. 4.47.

So overall, Σ^*_{Merc} is class-hiding according to Def. 4.50, which proves the theorem. □

In the final part of the security discussion for Σ^*_{Merc} we prove that Σ^*_{Merc} has an origin-hiding $\mathsf{Merc.ChgRep}$ algorithm under reasonable assumptions if the underlying SPS-EQ Σ_{SPSEQ} perfectly adapts signatures under malicious public keys.

Theorem 5.6 *Let* Σ_{SFPK}, Σ_{SPSEQ} *and* Σ^*_{Merc} *as in Thm. 5.3. Assume:*

(i) Σ_{SPSEQ} *perfectly adapts signatures under malicious public keys (see Def. 4.27).*

(ii) For all $\mathsf{pp} \leftarrow\!\!\$\ \mathsf{Merc.PGen}(\lambda)$, $\mathsf{pk} \in E$, $\sigma \in \mathcal{S}$, $r_\sigma \leftarrow\!\!\$\ \mathsf{Merc.ConvSigGen}(\mathsf{pp})$ *we have that* m' *in* $(m', \cdot) \leftarrow\!\!\$\ \mathsf{Merc.ChgRep}(\mathsf{pk}, m, \sigma, r_\sigma)$ *is a uniformly random message in* $[m]_{R_m}$.

Then the change-representative algorithm Merc.ChgRep *is origin-hiding according to Def. 4.51.*

Proof. Let $m \in \mathcal{M}$ be a message, $\mathsf{pp} \leftarrow\!\!\$\ \mathsf{Merc.PGen}(\lambda)$, $\mathsf{pk} \in E$, $\sigma \in \mathcal{S}$ with $\mathsf{Merc.Vfy}(\mathsf{pk}, m, \sigma) = 1$, $r_\sigma \leftarrow\!\!\$\ \mathsf{Merc.ConvSigGen}(\mathsf{pp})$. We have

$$\mathsf{Merc.ChgRep} := \mathsf{SPSEQ.ChgRep}$$

and consider

$$(m', \sigma') \leftarrow\!\!\$\ \mathsf{Merc.ChgRep}(\mathsf{pk}, m, \sigma, r_\sigma)$$

We need to prove the properties of the output distribution of Merc.ChgRep that are required for origin-hiding (Def. 4.51).

m' is (distributed like a) uniformly random message in $[m]_{R_m}$ by prerequisite. Since Σ_{SPSEQ} perfectly adapts signatures under malicious public keys (according to Def. 4.27), σ' is a uniformly random valid signature on m' as required for origin-hiding in Def. 4.51.

Thus, the Merc.ChgRep algorithm of Σ^*_{Merc} is origin-hiding according to Def. 4.51. □

So Σ^*_{Merc} is an unforgeable and class-hiding mercurial signature with an origin-hiding Merc.ChgRep algorithm if Σ_{SFPK} is a secure SFPK (i.e. unforgeable and fulfills a suitable class-hiding notion) and Σ_{SPSEQ} is a message class-hiding SPS-EQ that perfectly adapts signatures under malicious public keys. Note that discussing origin-hiding of the AdaptSig algorithm of Σ^*_{Merc} makes no sense since this algorithm is not constructed from the underlying SFPK and SPS-EQ but its existence is just assumed in Thm. 5.3.

All in all, we can conclude from Sect. 5.1.1 that from a secure mercurial signature with an SFPK-style trapdoor mechanism, we can extract a secure SFPK and a secure SPS-EQ under reasonable assumptions. A black-box construction of a syntactically correct mercurial signature Σ^*_{Merc} from SFPK and SPS-EQ is possible but it is unclear whether it can be securely instantiated (see Sect. 5.1.2).

5.2 Relation between key-homomorphic signatures and SFPK

In this section, we analyze the relation between key-homomorphic signatures from [DS16] (see Sect. 4.1) and SFPK [BHKS18] (see Sect. 4.3). The question about the relation between the two signature types is very natural since we remark that both key-homomorphic signatures and SFPK provide a mechanism to randomize a valid key pair to a new valid one. Given a key pair $(\mathsf{pk}, \mathsf{sk})$ of a key-homomorphic signature (Def. 4.2), one can use any secret key Δ to compute a new secret key $\mathsf{sk}' := \mathsf{sk} + \Delta$ and adapt the public key accordingly (using the secret-to-public key homomorphism μ) by computing $\mathsf{pk}' := \mathsf{pk} \cdot \mu(\Delta)$. In contrast to this very specific mechanism, SFPKs have a more black-box way to randomize key pairs using the SFPK.ChgPK and SFPK.ChgSK algorithms

with some key pair $(\mathsf{pk}, \mathsf{sk})$ and the same randomness r. Furthermore note that the space that the randomness r is drawn from is not prescribed by the syntax of SFPK. In particular, r does not have to be a valid secret key, in contrast to the shift amount Δ used for key-homomorphic signatures.

One of the most striking differences between key-homomorphic signatures and flexible public-key signatures is that, in contrast to SFPK, key-homomorphic signatures provide a secret-to-public-key homomorphism μ and an integrated algorithm Adapt to publicly randomize existing valid key-message-signature triples (pk, m, σ) to new triples $(\mathsf{pk}', m, \sigma')$ by adapting key and signature. To emulate this adaption behaviour for (pk, m, σ) with SFPK, one has to randomize the public key pk and the corresponding secret key sk separately as $\mathsf{pk}' := \mathsf{SFPK.ChgPK}(\mathsf{pk}, r)$ and $\mathsf{sk}' := \mathsf{SFPK.ChgSK}(\mathsf{sk}, r)$ and then freshly sign σ with sk'. Alternatively, if the SFPK has recoverable signing keys (Def. 4.40), one could also recover the secret key sk' corresponding to a randomized version $\mathsf{pk}' := \mathsf{SFPK.ChgPK}(\mathsf{pk}, r)$ as $\mathsf{sk}' := \mathsf{SFPK.Recover}(\mathsf{sk}, \tau, \mathsf{pk}')$. However, since both of these methods require knowledge of secret key or trapdoor, they cannot be seen as an equivalent to the public adaption mechanism provided by key-homomorphic signatures.

On the other hand, in contrast to key-homomorphic signatures, secure SFPK provide an equivalence relation with usually small equivalence classes (see Lem. 4.32) on the public key space, as well as a trapdoor-based algorithm $\mathsf{SFPK.ChkRep}$ which allows to check whether two given public keys are related or not.

In the following, we will formally examine under which assumptions we can construct SFPK from key-homomorphic signatures and vice versa.

5.2.1 SFPK from key-homomorphic signatures

A natural SFPK construction from key-homomorphic signatures revolves around an equivalence relation R_μ on the public key space E of a key-homomorphic signature scheme w.r.t. homomorphism μ. Two keys pk and pk' are related via R_μ if there is some $\Delta \in H$ with

$$\mathsf{pk}' = \mathsf{pk} \cdot \mu(\Delta)$$

We will first formally prove that the above relation R_μ indeed is an equivalence relation and then formally construct an SFPK $\Sigma^{(\mathsf{KH})}_{\mathsf{SFPK}}$ over R_μ from a key-homomorphic signature scheme with secret-to-public-key homomorphism μ. As a first result concerning its (in-)security, we prove that $\Sigma^{(\mathsf{KH})}_{\mathsf{SFPK}}$ is not strongly unforgeable. We then prove that $\Sigma^{(\mathsf{KH})}_{\mathsf{SFPK}}$ is not even (weakly) unforgeable if the homomorphism μ is surjective. Note that all examples for key-homomorphic signature schemes given by Derler and Slamanig in [DS16] (and listed in Rem. 4.7) have surjective secret-to-public-key homomorphisms and are thus not suitable to instantiate our SFPK-from-key-homomorphic-signature construction $\Sigma^{(\mathsf{KH})}_{\mathsf{SFPK}}$.

As announced above, we start with the formal proof that R_μ is an equivalence relation.

Lemma 5.7 *Let $(H, +), (E, \cdot)$ be groups, $\mu : H \to E$ be a group homomorphism. Then*

R_μ with

$$\mathsf{pk} \sim_{R_\mu} \mathsf{pk}' :\Leftrightarrow \exists r \in H : \mathsf{pk}' = \mathsf{pk} \cdot \mu(r)$$

is an equivalence relation on E (according to Def. 3.3).

Proof. We need to prove that R_μ is reflexive, symmetrical and transitive. With the basic properties of group homomorphisms (Lem. 3.10), we get

$$\mathsf{pk} \cdot \mu(1_H) = \mathsf{pk} \cdot 1_E = \mathsf{pk}$$

for any $\mathsf{pk} \in E$ which proves that R_μ is reflexive.

Let $\mathsf{pk}, \mathsf{pk}' \in E$ with $\mathsf{pk} \sim_{R_\mu} \mathsf{pk}'$. So we have

$$\exists r \in H : \mathsf{pk}' = \mathsf{pk} \cdot \mu(r)$$

Again with Lem. 3.10, we get

$$\mathsf{pk}' \cdot \mu(r^{-1}) = \mathsf{pk}' \cdot \mu(r)^{-1} = \mathsf{pk} \cdot \mu(r) \cdot \mu(r)^{-1} = \mathsf{pk} \cdot 1_E = \mathsf{pk}$$

which proves that R_μ is symmetrical.

Let $\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{pk}_2 \in E$ with

$$\mathsf{pk}_0 \sim_{R_\mu} \mathsf{pk}_1, \mathsf{pk}_1 \sim_{R_\mu} \mathsf{pk}_2$$

So by definition of R_μ, we have

$$\exists r_0, r_1 : \mathsf{pk}_1 = \mathsf{pk}_0 \cdot \mu(r_0) \wedge \mathsf{pk}_2 = \mathsf{pk}_1 \cdot \mu(r_1)$$

This yields

$$\mathsf{pk}_2 = \mathsf{pk}_1 \cdot \mu(r_1) = \mathsf{pk}_0 \cdot \mu(r_0) \cdot \mu(r_1) = \mathsf{pk}_0 \cdot \mu(r_0 + r_1)$$

which is equivalent to

$$\mathsf{pk}_0 \sim_{R_\mu} \mathsf{pk}_2$$

This proves that R_μ also is transitive and thus, R_μ is an equivalence relation according to Def. 3.3. □

We continue with the natural SFPK constrution from key-homomorphic signatures and the proof of its insecurity.

Theorem 5.8 *Let Σ_{KH} be a key homomorphic signature scheme with secret key space H, public key space E, key homomorphism $\mu : H \to E$, message space $\mathcal{M}$ and signature*

space $\mathcal{S}$. We define the SFPK

$$\begin{aligned}\Sigma_{\mathsf{SFPK}}^{(\mathsf{KH})} = (&\mathsf{SFPK.PGen} := \mathsf{KH.PGen},\\ &\mathsf{SFPK.KGen} := \mathsf{KH.KGen},\\ &\mathsf{SFPK.TKGen} := \mathsf{TKGen},\\ &\mathsf{SFPK.Sign} := \mathsf{KH.Sign},\\ &\mathsf{SFPK.ChkRep} := \mathsf{ChkRep},\\ &\mathsf{SFPK.KeyConvGen},\\ &\mathsf{SFPK.ChgPK},\\ &\mathsf{SFPK.ChgSK},\\ &\mathsf{SFPK.Vfy} := \mathsf{KH.Vfy},\\ &\mathsf{SFPK.VKey})\end{aligned}$$

(with the same key, signature and message spaces and over equivalence relation R_μ) as follows (where $\mathsf{TKGen}, \mathsf{ChkRep}$ *are arbitrary algorithms s.t. the correctness requirements (i) and (iii) from Def. 4.29 are fulfilled):*

$\mathsf{SFPK.KeyConvGen}(\mathsf{pp})$

1 : **return** $r := \Delta \leftarrow\!\$\ H$

$\mathsf{SFPK.VKey}(\mathsf{pk}, \mathsf{sk})$

1 : **return** 1 **if** $\mathsf{pk} = \mu(\mathsf{sk})$
2 : **else return** 0

$\mathsf{SFPK.ChgSK}(\mathsf{sk}, r)$

1 : **return** $\mathsf{sk}' := \mathsf{sk} + r$

$\mathsf{SFPK.ChgPK}(\mathsf{pk}, r)$

1 : **return** $\mathsf{pk}' := \mathsf{pk} \cdot \mu(r)$

Then we have

(i) $\Sigma_{\mathsf{SFPK}}^{(\mathsf{KH})}$ *is a correct SFPK (Def. 4.29) over the equivalence relation R_μ from Lem. 5.7.*

(ii) $\Sigma_{\mathsf{SFPK}}^{(\mathsf{KH})}$ *is not strongly existentially unforgeable.*

(iii) If μ is surjective, then $\Sigma_{\mathsf{SFPK}}^{(\mathsf{KH})}$ *is not unforgeable.*

Proof. **ad (i)** We need to prove that the requirements from Def. 4.29 are fulfilled. (i) and (iii) are clear by prerequisite, (iv) is clear by definition of $\Sigma_{\mathsf{SFPK}}^{(\mathsf{KH})}$. (ii) (a) is clear since Σ_{KH} is a key-homomorphic signature (Def. 4.2) and thus a correct digital signature. (ii) (b) (ii) is fulfilled since we have

$$\mu(\mathsf{sk}') = \mu(\mathsf{sk} + r) = \mu(\mathsf{sk}) \cdot \mu(r) = \mathsf{pk} \cdot \mu(r) = \mathsf{pk}'$$

by definition of $\mathsf{KeyConvGen}$, ChgPK, ChgSK and the homomorphic property of μ. So digital signature correctness of Σ_{KH} yields (ii) (b) (i) and thus $\Sigma_{\mathsf{SFPK}}^{(\mathsf{KH})}$ is a correct SFPK over R_μ from Lem. 5.7 according to Def. 4.29.

ad (ii) To prove that $\Sigma^{(\mathsf{KH})}_{\mathsf{SFPK}}$ is not sEUF-CMA (Def. 4.31), we construct a ppt adversary $\mathcal{A}$ that has non-negligible advantage in the strong SFPK EUF game for $\Sigma^{(\mathsf{KH})}_{\mathsf{SFPK}}$ (see Def. 4.31).

Let $\mathsf{pp} \leftarrow\!\!\$\ \mathsf{KH.PGen}(\lambda)$, $(\mathsf{pk}, \mathsf{sk}, \tau) \leftarrow\!\!\$\ \mathsf{TKGen}(\mathsf{pp}, \omega)$ with $\omega \leftarrow\!\!\$\ \{0,1\}^{\mathsf{poly}(\lambda)}$ as in said game. On input pk, τ, $\mathcal{A}$ acts as follows:

1. $\mathcal{A}$ queries the signing oracle for $m \leftarrow\!\!\$\ \mathcal{M}$.
2. Eventually, $\mathcal{A}$ obtains $\sigma \leftarrow\!\!\$\ \mathsf{KH.Sign}(\mathsf{sk}, m)$.
3. $\mathcal{A}$ samples $\Delta \leftarrow\!\!\$\ H$ computes $(\mathsf{pk}', \sigma') \leftarrow\!\!\$\ \mathsf{KH.Adapt}(\mathsf{pk}, m, \sigma, \Delta)$.
4. $\mathcal{A}$ outputs $(\mathsf{pk}', m, \sigma')$.

Let $(\mathsf{pk}', m, \sigma') \leftarrow\!\!\$\ \mathcal{A}(\mathsf{pk}, \tau)$. $\mathcal{A}$ obviously is a ppt algorithm, furthermore we have

$$\mathsf{pk}' = \mathsf{pk} \cdot \mu(\Delta) \in [\mathsf{pk}]_{R_\mu}$$

by definition of R_μ and the correctness requirement for $\mathsf{KH.Adapt}$. This yields $\mathsf{ChkRep}(\tau, \mathsf{pk}') = 1$. Because of the correctness requirement from Def. 4.2 for the $\mathsf{KH.Adapt}$ algorithm, we also have

$$\mathsf{KH.Vfy}(\mathsf{pk}', m, \sigma') = 1$$

Lastly, we have $(m, \sigma') \notin Q$ using the additional prerequisite that $\sigma \neq \sigma'$ and $Q = \{(m, \sigma)\}$. We can reasonably assume $\sigma \neq \sigma'$ since under the circumstances described above this would hold for all examples of key-homomorphic signature schemes Σ_{KH} that were listed by Derler and Slamanig in [DS16] (see Rem. 4.7).

Since $(\mathsf{pk}', m, \sigma') \leftarrow\!\!\$\ \mathcal{A}(\mathsf{pk}, \tau)$ was arbitrary, this proves

$$\Pr[\mathsf{Exp}^{\mathsf{sfpk\text{-}seuf}}_{\mathcal{A}, \Sigma^{(\mathsf{KH})}_{\mathsf{SFPK}}}(\lambda) = 1] = 1$$

so $\Sigma^{(\mathsf{KH})}_{\mathsf{SFPK}}$ is not strongly existentially unforgeable under chosen-message attacks according to Def. 4.31.

ad (iii) Intuitively, if μ is surjective, the quotient set E/R_μ (Def. 3.3) contains only one equivalence class and thus it is trivial to win the SFPK unforgeability game for $\Sigma^{(\mathsf{KH})}_{\mathsf{SFPK}}$. More formally, we prove $E/R_\mu = \{E\}$ by proving that any two public keys are related via R_μ. Let $\lambda \in \mathbb{N}$ be the security parameter, $\mathsf{pk}, \tilde{\mathsf{pk}} \in E$. Since μ is surjective, we have that

$$\exists \tilde{r} \in H : \mu(\tilde{r}) = \mathsf{pk}^{-1} \cdot \tilde{\mathsf{pk}}$$

So by definition of R_μ, we get

$$\tilde{\mathsf{pk}} = \mathsf{pk} \cdot \mathsf{pk}^{-1} \cdot \tilde{\mathsf{pk}} = \mathsf{pk} \cdot \mu(\tilde{r}) \in [\mathsf{pk}]_{R_\mu}$$

So since two arbitrary public keys are related, we get that the quotient set E/R_μ contains just one equivalence class, i.e.

$$E/R_\mu = \{E\}$$

Consider an adversary $\mathcal{A}$ which samples a fresh key pair $(\mathsf{pk}', \mathsf{sk}') \leftarrow\!\!\$\ \mathsf{SFPK.KGen}(\mathsf{pp})$ and a message $m \leftarrow\!\!\$\ \mathcal{M}$ and outputs a forgery $\sigma \leftarrow\!\!\$\ \mathsf{SFPK.Sign}(\mathsf{sk}', m)$ for m. By definition of the SFPK unforgeability game (Def. 4.30), we have

$$\mathsf{Adv}^{\mathsf{sfpk\text{-}euf}}_{\mathcal{A},\Sigma^{(\mathsf{KH})}_{\mathsf{SFPK}}}(\lambda) = 1$$

and thus $\Sigma^{(\mathsf{KH})}_{\mathsf{SFPK}}$ is not EUF-CMA secure according to Def. 4.30.

□

All in all, we have proven that the above natural construction of SFPK from key-homomorphic signatures is at least hard to properly instantiate (since all commonly known key-homomorphic signatures from Rem. 4.7 are not suitable according to Thm. 5.8), if not entirely useless. However, we have not formally proven that it is actually impossible to construct SFPK from key-homomorphic signatures in a black-box way.

5.2.2 Key-homomorphic signatures from SFPK

In this section we analyze under which assumptions we can construct key-homomorphic signatures from SFPK. As already sketched in the beginning of Sect. 5.2, SPFK do not provide a public adaption mechanism like key-homomorphic signatures. Furthermore, SFPK do not have a secret-to-public-key homomorphism in general.

We construct a black-box key-homomorphic scheme from a given SFPK as shown in the following definition. Note that this requires assuming that $\mathsf{SFPK.ChgSK}$ and $\mathsf{SFPK.ChgPK}$ define a secret-to-public-key homomorphism μ as well as the existence of a suitable Adapt which are strong assumptions. We are not aware of an SFPK that fulfills them.

Definition 5.9 (**key-homomorphic signature from SFPK**) *Let Σ_{SFPK} be an SFPK with groups $(H, +), (E, \cdot)$ as secret and public key space that fulfills the following conditions:*

(i) There exists a secret-to-public-key homomorphism $\mu : H \to E$ such that

$$\mathsf{SFPK.ChgSK}(\mathsf{sk}, \Delta) = \mathsf{sk} + \Delta \wedge \mathsf{SFPK.ChgPK}(\mathsf{pk}, \Delta) = \mathsf{pk} \cdot \mu(\Delta)$$

for all $\Delta \in H$.

(ii) There exists a ppt algorithm Adapt that takes as input a public key $\mathsf{pk} \in E$, a message $m \in \mathcal{M}$, a signature $\sigma \in \mathcal{S}$ and a shift amount secret key $\Delta \in H$ and outputs an adapted public key $\mathsf{pk}' \in E$ and an adapted signature $\sigma' \in \mathcal{S}$ which

satisfy the following requirement: For all $\Delta \in H$, $\lambda \in \mathbb{N}$, $\mathsf{pp} \leftarrow\!\!\$\ \mathsf{SFPK.PGen}(\lambda)$, $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{SFPK.KGen}(\mathsf{pp})$, $m \in \mathcal{M}$, $\sigma \in \mathcal{S}$, *we have:*

$$\begin{aligned} &\mathsf{SFPK.Vfy}(\mathsf{pk}, m, \sigma) = 1 \wedge (\mathsf{pk}', \sigma') \leftarrow\!\!\$\ \mathsf{Adapt}(\mathsf{pk}, m, \sigma, \Delta) \\ \Rightarrow\ &\Pr[\mathsf{SFPK.Vfy}(\mathsf{pk}', m, \sigma') = 1] = 1 \wedge \mathsf{pk}' = \mu(\Delta) \cdot \mathsf{pk} \end{aligned}$$

We construct a key-homomorphic signature scheme $\Sigma_{\mathsf{KH}}^{(\mathsf{SFPK})}$ *from* Σ_{SFPK} *as*

$$\begin{aligned} \Sigma_{\mathsf{KH}}^{(\mathsf{SFPK})} = (&\mathsf{KH.PGen} := \mathsf{SFPK.PGen}, \\ &\mathsf{KH.KGen} := \mathsf{SFPK.KGen}, \\ &\mathsf{KH.Sign} := \mathsf{SFPK.Sign}, \\ &\mathsf{KH.Adapt} := \mathsf{Adapt}, \\ &\mathsf{KH.Vfy} := \mathsf{SFPK.Vfy}) \end{aligned}$$

It is easy to see that $\Sigma_{\mathsf{KH}}^{(\mathsf{SFPK})}$ indeed is a correct key-homomorphic signature scheme according to Def. 4.2. The correctness of Σ_{SFPK} according to Def. 4.29 yields that $\Sigma_{\mathsf{KH}}^{(\mathsf{SFPK})}$ without Adapt is a correct digital signature scheme. Furthermore, Adapt fulfills the requirements from Def. 4.2 by assumption and μ is a secret-to-public-key homomorphism.

The above construction uses strong assumptions and thus illustrates the gap between SFPK and key-homomorphic signature schemes. SFPK do neither provide secret-to-public-key homomorphisms nor a public way to adapt signatures "out of the box", however, it is possible that key changing with $\mathsf{SFPK.ChgSK}$ and $\mathsf{SFPK.ChgPK}$ is done in a homomorphic way as shown in the above Def. 5.9.

To conclude the discussion about key-homomorphic signatures from SFPK, we prove that the SFPK Σ_{SFPK} used to construct the key-homomorphic signature scheme $\Sigma_{\mathsf{KH}}^{(\mathsf{SFPK})}$ from Def. 5.9 cannot be strongly existentially unforgeable under chosen message attacks. This also implies that key-homomorphic SFPKs cannot be strongly existentially unforgeable.

Lemma 5.10 *The SFPK* Σ_{SFPK} *from Def. 5.9 is not strongly existentially unforgeable.*

Proof. Note that the input and output spaces of Adapt from Def. 5.9 are composed of the key, message and signature spaces of Σ_{SFPK}. So the construction of a ppt adversary $\mathcal{A}$ that breaks the sEUF-CMA of Σ_{SFPK} can be done analogously to the construction in the proof that $\Sigma_{\mathsf{SFPK}}^{(\mathsf{KH})}$ is not sEUF-CMA (Thm. 5.8). $\mathcal{A}$ draws a random message m, submits it to the signing oracle, randomizes the resulting signature σ using Adapt and outputs the randomized signature σ' as a forgery for m. The analysis of this straightforward attack is analogous to Thm. 5.8 and omitted here for brevity. □

5.3 Analysis of key-homomorphic properties of the Backes warm-up SFPK

In Rem. 4.7, we listed some examples for key-homomorphic signatures that were originally given by Derler and Slamanig [DS16]. In this chapter, we are going to examine the warm-up SFPK construction from [BHKS18] for key-homomorphic properties according to [DS16] (see Sect. 4.1.1). We decided to cut the respective analysis of the Backes multi-user SFPK [BHKS18], the SPS-EQ schemes by Fuchsbauer et al. [FHS14] and Conolly et al. [CLPK22] and the mercurial signature from [CL19] for scope reasons and leave it as open work.

The *warm-up SFPK scheme* $\Sigma_{\mathsf{SFPK}}^{\mathsf{bck\text{-}wu}}$ from [BHKS18] is defined over the equivalence relation R_{exp} from the following definition.

Definition 5.11 *Let $G = \langle g \rangle$ be a cyclic group of prime order $p \in \mathbb{P}$, $l \in \mathbb{N}$. We define the relation R_{exp} over $(G \setminus \{1_G\})^l$ where*

$$M \sim_{R_{\mathsf{exp}}} N :\Leftrightarrow \exists r \in \mathbb{Z}_p^* : M^r = N$$

The following lemma formally proves that R_{exp} indeed is an equivalence relation.

Lemma 5.12 *R_{exp} from Def. 5.11 is an equivalence relation according to Def. 3.3.*

Proof. Reflexivity is clear since $M^1 = M$ for all $M \in G^l$. Symmetry comes from the fact that

$$M^r = N \Leftrightarrow M = N^{1/r}$$

for all $M, N \in (G \setminus \{1_G\})^l$, $r \in \mathbb{Z}_p^*$. What is left to prove is the transitivity of R_{exp}. Let $M, N, Q \in (G \setminus \{1_G\})^l$ with

$$M \sim_{R_{\mathsf{exp}}} N, N \sim_{R_{\mathsf{exp}}} Q$$

which by definition means that

$$\exists r, t \in \mathbb{Z}_p^* : M^r = N \wedge N^t = Q$$

So we get

$$Q = N^t = (M^r)^t = M^{rt}$$

which is equivalent to

$$M \sim_{R_{\mathsf{exp}}} Q$$

So R_{exp} also is transitive and thus an equivalence relation according to Def. 3.3. □

The Backes warm-up scheme $\Sigma_{\mathsf{SFPK}}^{\mathsf{bck\text{-}wu}}$ works in the type-2 and type-3 bilinear group setting and uses a *programmable hash function* $\mathcal{H}$ which is a type of enhanced hash function. However, for simplicity, we can imagine this function as a normal hash function with a group as a codomain since the special PHF properties are only used in the security proof for $\Sigma_{\mathsf{SFPK}}^{\mathsf{bck\text{-}wu}}$ (from [BHKS18]) which we omit in this work. The general idea of $\Sigma_{\mathsf{SFPK}}^{\mathsf{bck\text{-}wu}}$ is based on a modified version of Waters' signatures [Wat05] as described

by Chatterjee and Menezes [CM11]. Signing works by hashing the message using a personal (public) PHF key, then blinding the result and multiplying with a personal (secret) group element, generated from a secret base and exponent. In the following, we give the formal definition of $\Sigma^{\mathsf{bck\text{-}wu}}_{\mathsf{SFPK}}$ based on [BHKS18].

Definition 5.13 (**Backes warm-up SFPK**) *The* Backes warm-up SFPK $\Sigma^{\mathsf{bck\text{-}wu}}_{\mathsf{SFPK}}$ *is defined as follows:*

$\mathsf{SFPK.PGen}(\lambda)$

1 : $\mathrm{BG} := (G_1, G_2, G_T, p, e, g_1, g_2) \leftarrow\!\$\ \mathsf{BGGen}(\lambda, 3)$
2 : **return** BG

$\mathsf{SFPK.KGen}(\mathsf{pp}, \omega)$

1 : $K_{\mathsf{PHF}} \leftarrow\!\$\ \mathsf{PHF.KGen}(\lambda)$
2 : $A, B, C, D, X \leftarrow\!\$\ G_1$
3 : $y \leftarrow\!\$\ \mathbb{Z}_p^*$
4 : $t := e(X^y, g_2)$
5 : $\mathsf{pk} := (A, B, C, D, t, K_{\mathsf{PHF}})$
6 : $\mathsf{sk} := (y, X, \mathsf{pk})$
7 : **return** $(\mathsf{pk}, \mathsf{sk})$

$\mathsf{SFPK.TKGen}(\mathsf{pp}, \omega)$

1 : $K_{\mathsf{PHF}} \leftarrow\!\$\ (g_1^{\zeta_i} \mid 0 \le i \le \lambda, \zeta_i \leftarrow\!\$\ \mathbb{Z}_p)$
2 : $a, b, c, d, x, y \leftarrow\!\$\ \mathbb{Z}_p^*$
3 : $t := e(g_1^{xy}, g_2)$
4 : $\mathsf{pk} := (g_1^a, g_1^b, g_1^c, g_1^{xd}, t, K_{\mathsf{PHF}})$
5 : $\mathsf{sk} := (y, g_1^x, \mathsf{pk})$
6 : $\tau := (d, g_2^y, g_2^a, g_2^b, g_2^c, g_2^{\zeta_0}, g_2^{\zeta_1}, \ldots, g_2^{\zeta_\lambda})$
7 : **return** $(\mathsf{pk}, \mathsf{sk}, \tau)$

$\mathsf{SFPK.Sign}(\mathsf{sk}, m)$

1 : ***parse*** $(y, X, \mathsf{pk}) := \mathsf{sk}$
2 : $r \leftarrow\!\$\ \mathbb{Z}_p^*$
3 : **return** $\sigma := (X^y \cdot (\mathcal{H}_{K_{\mathsf{PHF}}}(m))^r, g_1^r, g_2^r)$

$\mathsf{SFPK.KeyConvGen}(\mathsf{pp})$

1 : **return** $r \leftarrow\!\$\ \mathbb{Z}_p^*$

$\mathsf{SFPK.ChgPK}(\mathsf{pk}, r)$

1 : ***parse*** $(A, B, C, D, t, K_{\mathsf{PHF}}) := \mathsf{pk}$
2 : **return** $\mathsf{pk}' := (A^r, B^r, C^r, D^r, t^r, (K_{\mathsf{PHF}})^r)$

$\mathsf{SFPK.ChgSK}(\mathsf{sk}, r)$

1 : ***parse*** $(y, X, \mathsf{pk}) := \mathsf{sk}$
2 : $\mathsf{pk}' := \mathsf{SFPK.ChgPK}(\mathsf{pk}, r)$
3 : **return** $\mathsf{sk}' := (y, (X^r), \mathsf{pk}')$

$\mathsf{SFPK.ChkRep}(\tau, \mathsf{pk})$

1 : ***parse*** $(\mathsf{pk}_1, \mathsf{pk}_2, \mathsf{pk}_3, X, t, \mathsf{pk}_4, \ldots, \mathsf{pk}_{\lambda+4}) := \mathsf{pk}'$,
2 : $(d, Y_2, \tau_1, \ldots, \tau_{\lambda+4}) := \tau$
3 : **return** $e(X^{d^{-1}}, Y_2) = t \wedge$
4 : $\bigwedge_{i=1}^{\lambda+4} \bigwedge_{j=1}^{\lambda+4} e(\mathsf{pk}_i, \tau_j) = e(\mathsf{pk}_j, \tau_i)$

$\mathsf{SFPK.Vfy}(\mathsf{pk}, m, \sigma)$

1: ***parse*** $(\sigma_1, \sigma_2, \sigma_3) := \sigma$,
2: $(A, B, C, D, t, K_{\mathsf{PHF}}) := \mathsf{pk}$
3: **return** $e(\sigma_2, g_2) = e(g_1, \sigma_3) \wedge$
4: $e(\sigma_1, g_2) = t \cdot e(\mathcal{H}_{K_{\mathsf{PHF}}}(m), \sigma_3)$

$\mathsf{SFPK.VKey}(\mathsf{pk}, \mathsf{sk})$

1: ***parse*** $(A, B, C, D, t, K_{\mathsf{PHF}}) := \mathsf{pk}$,
2: $(y, X, \mathsf{pk}') := \mathsf{sk}$
3: **return** $\mathsf{pk} = \mathsf{pk}' \wedge t = e(X^y, g_2)$

$\mathsf{SFPK.Recover}(\mathsf{sk}, \tau, \mathsf{pk}')$

1: ***parse*** $(y, g_1^x, \mathsf{pk}) := \mathsf{sk}$,
2: $(d, g_2^y, g_2^a, g_2^b, g_2^c, g_2^{\zeta_0}, \ldots, g_2^{\zeta_\lambda}) := \tau$,
3: $(A^r, B^r, C^r, D^r, t^r, (K_{\mathsf{PHF}})^r) := \mathsf{pk}'$
4: $X' := (D^r)^{(d^{-1})}$
5: **return** $\mathsf{sk}' := (y, X', \mathsf{pk}')$

For scope reasons, we do not formally prove neither correctness nor security of $\Sigma_{\mathsf{SFPK}}^{\mathsf{bck\text{-}wu}}$ but only summarize the results from [BHKS18] in the following theorem.

Theorem 5.14 *(i) If the decisional linear assumption (Def. A.2) holds for all* λ, $\mathrm{BG} \leftarrow\!\$\ \mathsf{SFPK.PGen}(\lambda)$ *and* PHF *is* $(1, \mathsf{poly}(\lambda))$*-programmable then* $\Sigma_{\mathsf{SFPK}}^{\mathsf{bck\text{-}wu}}$ *is unforgeable (Def. 4.30).*

(ii) If for all $\lambda \in \mathbb{N}$, $\mathrm{BG} \leftarrow\!\$\ \mathsf{SFPK.PGen}(\lambda)$ *the decisional Diffie-Hellman assumption (Def. A.1) holds in* G_1 *then* $\Sigma_{\mathsf{SFPK}}^{\mathsf{bck\text{-}wu}}$ *is class-hiding (Def. 4.33).*

At first glance, it seems that the projection μ to the public key is a good candidate for a secret-to-public-key homomorphism for $\Sigma_{\mathsf{SFPK}}^{\mathsf{bck\text{-}wu}}$ according to Def. 4.1. However, in the following Thm. 5.15, we prove that this μ is actually not a secret-to-public-key homomorphism for $\Sigma_{\mathsf{SFPK}}^{\mathsf{bck\text{-}wu}}$.

Theorem 5.15 *Let* $\Sigma_{\mathsf{SFPK}}^{\mathsf{bck\text{-}wu}}$ *be the Backes warm-up SFPK from Def. 5.13. Let* $\mathrm{BG} := (G_1, G_2, G_T, p, e, g_1, g_2) \leftarrow\!\$\ \mathsf{SFPK.PGen}(\lambda)$ *be bilinear group. Then*

(i) $E := G_1^4 \times G_T \times G_1^{\lambda+1}$ *is the public key space,* $H := \mathbb{Z}_p^* \times G_1 \times E$ *is the secret key space of* $\Sigma_{\mathsf{SFPK}}^{\mathsf{bck\text{-}wu}}$.

(ii) $\mu : H \to E, (y, X, \mathsf{pk}) \mapsto \mathsf{pk}$ *is not a secret-to-public-key homomorphism for* $\Sigma_{\mathsf{SFPK}}^{\mathsf{bck\text{-}wu}}$.

Proof. **ad (i)** This follows from inspection. Both H and E are groups that are direct products of smaller groups so the group operation in H is defined component-wise, using the group operations from the smaller groups.

ad (ii) We denote the group operation in secret key space H with $+_H$ and the group operation in public key space E with $\cdot_E$. Let $\mathsf{sk}, \mathsf{sk}' \in H$ with

$$\mathsf{sk} := (y, X, \mathsf{pk}), \mathsf{sk}' := (y', X', \mathsf{pk}')$$

and $\mathsf{pk}, \mathsf{pk}' \in E$ with

$$\mathsf{pk} := (A, B, C, D, t, K_{\mathsf{PHF}}), \mathsf{pk}' := (A', B', C', D', t', K'_{\mathsf{PHF}})$$

We prove the claim by proving that $\mathsf{pk} \cdot_E \mathsf{pk}' = \mu(\mathsf{sk} +_H \mathsf{sk}')$ is not a valid secret key for $\mathsf{sk} +_H \mathsf{sk}'$. We have

$$\begin{aligned}
&\mathsf{sk} +_H \mathsf{sk}' = (y \cdot y', X \cdot X', \mathsf{pk} \cdot_E \mathsf{pk}'),\\
&\mathsf{pk} \cdot_E \mathsf{pk}' = (A \cdot A', B \cdot B', C \cdot C', D \cdot D', t \cdot t', K_{\mathsf{PHF}} \cdot K'_{\mathsf{PHF}})
\end{aligned}$$

According to the $\mathsf{SFPK.KGen}$ and $\mathsf{SFPK.VKey}$ algorithm of $\Sigma^{\mathsf{bck\text{-}wu}}_{\mathsf{SFPK}}$ (Def. 5.13), the corresponding public key pk^* to $\mathsf{sk} +_H \mathsf{sk}'$ must fulfill

$$t^* = e((X \cdot X')^{y \cdot y'}, g_2)$$

where $t^* \in G_T$ is the fifth component of pk^*. Inserting the fifth component of $\mathsf{pk} \cdot_E \mathsf{pk}'$ into the above equation, we get

$$\begin{aligned}
t \cdot t' &= e(X^y, g_2) \cdot e(X'^{y'}, g_2) = e(X^y \cdot X'^{y'}, g_2) = e(g_1^{x \cdot y} \cdot g_1^{x' \cdot y'}, g_2)\\
&= e(g_1^{x \cdot y + x' \cdot y'}, g_2)\\
&\neq e(g_1^{x \cdot y \cdot y' + x' \cdot y \cdot y'}, g_2) = e((g_1^{x+x'})^{y \cdot y'}) = e((X \cdot X')^{y \cdot y'}, g_2)
\end{aligned}$$

where the inequality holds with high probability and comes from the fact that

$$g_1^{x \cdot y + x' \cdot y'} \neq g_1^{x \cdot y \cdot y' + x' \cdot y \cdot y'}$$

holds with high probability and the observation that because of the non-degeneracy property of e, $e(\cdot, g_2)$ is an injective mapping. This proves that $\mathsf{pk} \cdot \mathsf{pk}'$ is not a valid public key for $\mathsf{sk} +_H \mathsf{sk}'$ which means that since $\mathsf{pk}, \mathsf{pk}'$ were valid public keys for $\mathsf{sk}, \mathsf{sk}'$, respectively, μ is not a secret-to-public-key homomorphism for $\Sigma^{\mathsf{bck\text{-}wu}}_{\mathsf{SFPK}}$. □

Note that Thm. 5.15 does not prove that $\Sigma^{\mathsf{bck\text{-}wu}}_{\mathsf{SFPK}}$ is not a key-homomorphic signature scheme according to Def. 4.2. There might be a different homomorphism $\hat{\mu} \neq \mu$ that fulfills Def. 4.2. However, to the best of our knowledge, such a $\hat{\mu}$ does not exist.

To end this subsection, we want to make the following remark about the role of Waters signatures [Wat05] for the construction of key-homomorphic signatures and SFPK. In [DS16], Derler and Slamanig prove that the Waters signature variant from [BFG13], which has shared hashing parameters (i.e. the parameters for the instantiation of the Waters hash function are part of the signature schemes public parameters) is a key-homomorphic signature according to Def. 4.2. They claim that without shared hashing parameters, the Waters signature is not key-homomorphic since they conjecture that it would be impossible to define a $\mathsf{KH.Adapt}$ algorithm satisfying Def. 4.2 in this case. On the other hand, Backes et al. [BHKS18] use another variant (from [CM11]) of the Waters

signature (which has no shared hashing parameters, i.e. the parameters to instantiate the Waters hash function are generated in the key generation algorithm SFPK.KGen instead of the public parameter generation algorithm SFPK.PGen) to construct the above flexible public key signature (Def. 5.13).

So the difference between the two variants of the Waters hash function used to construct a key-homomorphic signature in [DS16] and an SFPK in [BHKS18], respectively, is the point at which the hash function used for signing is instantiated. This gives an insight to why Waters hash functions can be used to build both key-homomorphic signatures and SFPK, despite the fact that we have elaborated fundamental conceptual differences between the two primitives in Sect. 5.2.

6 Applications of the advanced signature primitives

In the following we will give a brief overview of the advanced cryptographic primitives that have already been constructed from the four signature types discussed in this thesis. For a more detailed overview including explanations of the primitives, refer back to Chapter 2.

- In [DS16], Derler and Slamanig introduced key-homomorphic signature schemes (Sect. 4.1.1) and showed how to construct ring signatures [RST06], universal designated verifier signatures [SBWP03], simulation-sound extractable non-interactive zero-knowledge proof systems and multikey-homomorphic signatures [DS16].
  - They also introduced a related primitive called publicly-key-homomorphic signatures and showed how to construct multisignatures [IN83] from them.

- In [FHS19], Fuchsbauer et al. showed how to construct anonymous credential systems (ACS) from secure SPS-EQ [FHS14]. As another application of SPS-EQ, Bobolz et al. have constructed a privacy-preserving incentive system from an SPS-EQ in [BEK+20].

- In [BHKS18], Backes et al. introduced flexible public key signatures (Sect. 4.3) and constructed stealth address protocols [Tod] and ring signatures [RST06] from them. In the same paper, the authors also showed how to construct a group signature scheme from a combination of SFPK (Sect. 4.3) and SPS-EQ (Sect. 4.2) . Note that while group signatures were originally introduced by Chaum and van Heyst [CH91], the construction from [BHKS18] uses the more elaborate security definitions by Bellare et al. [BMW03].

- In [CL19], Crites and Lysyanskaya introduced mercurial signatures and showed how to construct delegatable anonymous credentials from them in a black-box way.

In the following, we discuss two attempts to construct some of the above cryptographic primitives from an alternative signature type. More precisely, we discuss the construction of the group signature by Backes et al. [BHKS18] from mercurial signatures as well as the SSE-NIZK proof system by Derler and Slamanig [DS16] from SFPK.

Group signature from mercurials We will first attempt to construct the group signature by Backes et al. [BHKS18] from mercurial signatures. Since in Sect. 5.1.1, it was proven that, under reasonable assumptions, a mercurial signature can play the role of both an SFPK and an SPS-EQ, it is a natural question whether the Backes group signature can be constructed from mercurials alone. Note that this is a non-trivial question since (without further analysis) it is not clear under which assumptions a given mercurial signature Σ_{Merc} can take the role of both the SFPK and the SPS-EQ in the original Backes group signature construction from [BHKS18]. Furthermore, when using a mercurial instead of an SFPK to sign messages, it is possible to efficiently randomize the created message-signature pairs subsequently using the $\mathsf{Merc.ChgRep}$ algorithm (which SFPKs do not have an equivalent for). We discuss the implications of this for the security of the group signature construction in Thm. 6.3.

Group signatures (originally introduced in [CH91]) allow a signer S to sign a message on behalf of a group of n signers in total (including S himself). When producing such a signature, S stays anonymous among the signers in the group, i.e. it is impossible to efficiently tell publicly that S was the signer who created that signature. Note that if not stated otherwise, the term "group" in this subsection may not be confused with the mathematical structure introduced in Sect. 3.1. In this work, a group signature is a tuple $\Sigma_{\mathsf{GS}} = (\mathsf{GS.KGen}, \mathsf{GS.Sign}, \mathsf{GS.Vfy}, \mathsf{GS.Open})$ of ppt algorithms. For details about the syntax and security of group signatures refer to Appendix B.

We will start with stating the formal pseudocode of our mercurial-signature-based variant of the Backes group signature from [BHKS18].

Definition 6.1 (**mercurial-based Backes group signature**) *Let $l \in \mathbb{N}$ be the length parameter for the mercurial signature, $n \in \mathbb{N}$ the group size, λ the security parameter. Let Σ_{Merc} be a mercurial signature scheme with public key space E, secret key space H, message space $\mathcal{M}$, signature space $\mathcal{S}$, message relation R_m and public key relation R_{pk} that fulfills the following properties:*

(i) $E = \mathcal{M}$

(ii) The output spaces of $\mathsf{Merc.KeyConvGen}$ *and* $\mathsf{Merc.ConvSigGen}$ *are the same.*

(iii) There is a ppt trapdoor key generation algorithm TKGen *that takes as input public parameters* pp *and outputs a trapdoor τ in some trapdoor space T and a key pair $(\mathsf{pk}, \mathsf{sk}) \in E \times H$. Furthermore there is a check-representative algorithm* ChkRep *that takes as input a class trapdoor $\tau \in T$ as a public key $\mathsf{pk} \in E$ and outputs a bit $b \in \{0, 1\}$.* TKGen *and* ChkRep *fulfill the following requirements:*

a) For all $\lambda \in \mathbb{N}$, $\mathsf{pp} \leftarrow\!\!\$\ \mathsf{Merc.PGen}(\lambda)$, $\mathsf{Merc.KGen}(\mathsf{pp})$ and $\mathsf{TKGen}(\mathsf{pp})$ are identically distributed.

b) For all $\lambda \in \mathbb{N}$, $\mathsf{pp} \leftarrow\!\!\$\ \mathsf{Merc.PGen}(\lambda)$, $(\mathsf{pk}, \mathsf{sk}, \tau) \leftarrow\!\!\$\ \mathsf{TKGen}(\mathsf{pp})$ and all public keys $\mathsf{pk}' \in E$ we have

$$\mathsf{ChkRep}(\tau, \mathsf{pk}') = 1 \Leftrightarrow \mathsf{pk}' \in [\mathsf{pk}]_{R_{\mathsf{pk}}}$$

(iv) R_m *and* R_{pk} *are compatible, which means that for all* $\mathsf{pp} \leftarrow\!\$\ \mathsf{Merc.PGen}(\lambda)$, $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\$\ \mathsf{Merc.KGen}(\lambda)$, $(\mathsf{pk}_{\mathsf{cert}}, \mathsf{sk}_{\mathsf{cert}}) \leftarrow\!\$\ \mathsf{Merc.KGen}(\lambda)$, $\sigma_{\mathsf{cert}} \leftarrow\!\$\ \mathsf{Merc.Sign}(\mathsf{sk}_{\mathsf{cert}}, \mathsf{pk})$, $r \leftarrow\!\$\ \mathsf{Merc.KeyConvGen}(\mathsf{pp})$, *we get that for*

$$\mathsf{pk}_1 := \mathsf{Merc.ConvertPK}(\mathsf{pk}, r), \mathsf{pk}_2 \leftarrow\!\$\ \mathsf{Merc.ChgRep}(\mathsf{pk}, \sigma_{\mathsf{cert}}, r, \mathsf{pk}_{\mathsf{cert}})$$

we have

$$\mathsf{pk}_1 = \mathsf{pk}_2$$

(v) Let $\mathsf{pp} \leftarrow\!\$\ \mathsf{Merc.PGen}(\lambda)$ *and for* $1 \leq i \leq n$, *let* $(\mathsf{pk}_i, \cdot) \leftarrow\!\$\ \mathsf{Merc.KGen}(\mathsf{pp}, l)$. *We have*

$$\Pr[\exists 1 \leq i, j \leq n : \mathsf{pk}_i = \mathsf{pk}_j] \leq \epsilon(\lambda)$$

for a negligible function ϵ.

We define the mercurial-based Backes group signature $\Sigma^{(\mathsf{Merc})}_{\mathsf{GS},B}$ *based on* Σ_{Merc} *as follows:*

$\mathsf{GS.KGen}(\lambda, n)$

1: $\mathsf{pp} \leftarrow\!\$\ \mathsf{Merc.PGen}(\lambda)$
2: $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\$\ \mathsf{Merc.KGen}(\mathsf{pp}, l)$
3: **for** $i \in [n]$
4: $\quad(\mathsf{pk}_i, \mathsf{sk}_i, \tau_i) \leftarrow\!\$\ \mathsf{TKGen}(\mathsf{pp}, l)$
5: $\quad\sigma_{\mathsf{cert},i} \leftarrow\!\$\ \mathsf{Merc.Sign}(\mathsf{sk}, \mathsf{pk}_i)$
6: **return** $(\mathsf{gpk} := (\mathsf{pp}, \mathsf{pk}),$
7: $\quad\mathsf{gmsk} := ((\tau_i, \mathsf{pk}_i))_{i=1}^n,$
8: $\quad\mathsf{gsk}[i] := (\mathsf{pk}_i, \mathsf{sk}_i, \sigma_{\mathsf{cert},i}))$

$\mathsf{GS.Sign}(\mathsf{gsk}[i], m)$

1: ***parse*** $\mathsf{gsk}[i] = (\mathsf{pk}_i, \mathsf{sk}_i, \sigma_{\mathsf{cert},i})$
2: $r \leftarrow\!\$\ \mathsf{Merc.KeyConvGen}(\mathsf{pp})$
3: $\mathsf{pk}'_i := \mathsf{Merc.ConvertPK}(\mathsf{pk}_i, r)$
4: $\mathsf{sk}'_i := \mathsf{Merc.ConvertSK}(\mathsf{sk}_i, r)$
5: $(\mathsf{pk}'_i, \sigma'_{\mathsf{cert},i}) \leftarrow\!\$\ \mathsf{Merc.ChgRep}(\mathsf{pk}, \mathsf{pk}_i, \sigma_{\mathsf{cert},i}, r)$
6: $M := m||\sigma'_{\mathsf{cert},i}||\mathsf{pk}'_i$
7: $\sigma \leftarrow\!\$\ \mathsf{Merc.Sign}(\mathsf{sk}'_i, M)$
8: **return** $\sigma_{\mathsf{GS}} := (\mathsf{pk}'_i, \sigma, \sigma'_{\mathsf{cert},i})$

$\mathsf{GS.Vfy}(\mathsf{gpk}, m, \sigma)$

1: ***parse*** $\sigma_{\mathsf{GS}} := (\mathsf{pk}'_j, \sigma, \sigma'_{\mathsf{cert},j}),$
2: $\mathsf{gpk} := (\mathsf{pp}, \mathsf{pk})$
3: **if** $\mathsf{Merc.Vfy}(\mathsf{pk}'_j, \sigma'_{\mathsf{cert},j}, \mathsf{pk}) = 0$
4: $\quad$**return** 0
5: $M := m||\sigma'_{\mathsf{cert},j}||\mathsf{pk}'_j$
6: **return** $\mathsf{Merc.Vfy}(\mathsf{pk}'_j, M, \sigma)$

$\mathsf{GS.Open}(\mathsf{gmsk}, m, \sigma)$

1: ***parse*** $\sigma_{\mathsf{GS}} := (\mathsf{pk}'_j, \sigma, \sigma'_{\mathsf{cert},j}),$
2: $\mathsf{gmsk} := ((\tau_i, \mathsf{pk}_i))_{i=1}^n$
3: **if** $\mathsf{GS.Vfy}(\mathsf{gpk}, m, \sigma_{\mathsf{GS}}) = 0$
4: $\quad$**return** $\perp$
5: **if** $\neg\exists i \in [n] : \mathsf{ChkRep}(\tau_i, \mathsf{pk}'_j) = 1$
6: $\quad$**return** $\perp$
7: **else return** i

The prerequisites in Def. 6.1 include strong assumptions on the underlying mercurial signature which are not fulfilled by state-of-the-art mercurial signature constructions from e.g. [CL19, CLPK22]. In the following, we go through the prerequisites and discuss the details. In the original construction from [BHKS18], every signer is equipped with an SFPK key pair $(\mathsf{pk}_i, \mathsf{sk}_i)$ together with an SPS-EQ signature $\sigma_{\mathsf{cert},i}$ on its public key. Signing works by randomizing the public key pk_i to a different representative

$\mathsf{pk}'_i \leftarrow\!\$\ \mathsf{SFPK.ChgPK}(\mathsf{pk}_i, r)$ and adapting the secret key accordingly by computing $\mathsf{sk}'_i \leftarrow\!\$\ \mathsf{SFPK.ChgSK}(\mathsf{sk}_i, r)$. The signer then also randomizes the certificate $\sigma_{\mathsf{cert},i}$ for the public key before he signs the randomized public key pk'_i, the randomized certificate $\sigma'_{\mathsf{cert},i}$ and the actual message m with the randomized secret key as

$$\sigma \leftarrow\!\$\ \mathsf{SFPK.Sign}(\mathsf{sk}'_i, m||\sigma'_{\mathsf{cert},i}||\mathsf{pk}'_i)$$

The resulting group signature consists of this signature sigma σ as well as the certified public key $(\mathsf{pk}'_i, \sigma'_{\mathsf{cert},i})$ involved in its creation. So only if the public key space is equal to the message space (see requirements in Def. 6.1), the certificates for the public keys as well as the actual signatures can be created using the Merc.Sign algorithm of a mercurial signature. This requires the creation of $n+1$ key pairs for that signature, one for each signer and one for the group manager (needed to create the initial certificates $\sigma_{\mathsf{cert},i}$ for the group members' public keys). In [CL19], Crites and Lysyanskaya state that they are not aware of a mercurial signature construction whose message space contains the public key space as a subset.

The opening algorithm GS.Open in the original construction relies on the SFPKs trapdoor check algorithm SFPK.ChkRep to test all SFPK user public keys pk_i for containment in the class of the public key pk'_j in the signature in question. General mercurial signatures do not have such a trapdoor mechanic (i.e. a TKGen and ChkRep algorithm that fulfill similar correctness requirements to the SFPK correctness definition (see Def. 4.29 and the prerequisites of the above Def. 6.1)). It is also notable that the idea for ChkRep of $\Sigma^{\mathsf{bck\text{-}wu}}_{\mathsf{SFPK}}$ [BHKS18] from Def. 5.13 likely cannot be adapted to the Crites-Lysyanskaya mercurial signature [CL19] from Def. 4.52 since they have fundamentally different public key spaces: the public keys of $\Sigma^{\mathsf{bck\text{-}wu}}_{\mathsf{SFPK}}$ include an element from the codomain group G_T of a pairing (amongst elements of domain group G_1) while the public keys of $\Sigma^{\mathsf{CL}}_{\mathsf{Merc}}$ only consist of G_1 elements. Note that it might be possible to use public-key encryption and zero-knowledge proofs as in the group signature by Bellare et al. from [BMW03] to modify the GS.Sign algorithm of $\Sigma^{(\mathsf{Merc})}_{\mathsf{GS},B}$ in an encrypt-then-prove way that makes efficient subsequent opening of signatures possible. However, this would make the construction mostly pointless since a large benefit of the original SFPK-and-SPS-EQ-based group signature construction by Backes et al. (that $\Sigma^{(\mathsf{Merc})}_{\mathsf{GS},B}$ is based on) is that it is more efficient because it explicitly does not use an encrypt-then-prove-like GS.Sign algorithm. So the existence of some TKGen and ChkRep algorithms implementing such an SFPK-style trapdoor mechanism in Def. 6.1 is a strong assumption. We are not aware of a mercurial signature construction that fulfills it.

The fourth prerequisite in Def. 6.1 is the equivalent to what was defined as compatibility of SFPK and SPS-EQ in [Sch20]. It basically requires that adaption of a public key pk using $\mathsf{pk}' := \mathsf{Merc.ConvertPK}(\mathsf{pk}, r)$ and randomization of a certified public key using $\mathsf{pk}' \leftarrow\!\$\ \mathsf{Merc.ChgRep}(\mathsf{pk}_{\mathsf{cert}}, \mathsf{pk}, \sigma, r)$ (here, the public key is treated like a message) actually yield the same public key pk'.

The correctness of the above group signature follows directly from inspection and the correctness of the underlying mercurial signature and therefore the proof is omitted

here. Note that without the fifth prerequisite, the opening algorithm would not work as required by group signature correctness (Def. B.2) since it cannot tell randomized versions of pk_i, pk_j with $\mathsf{pk}_i \sim_{R_{\mathsf{pk}}} \mathsf{pk}_j$ apart.

We next briefly cover the security of the above mercurial-based group signature, starting with full-traceability (Def. B.4).

Theorem 6.2 *Let Σ_{Merc}, $\Sigma^{(\mathsf{Merc})}_{\mathsf{GS},B}$ as in Def. 6.1. If Σ_{Merc} is unforgeable (Def. 4.45), then $\Sigma^{(\mathsf{Merc})}_{\mathsf{GS},B}$ is fully-traceable (Def. B.4).*

It is notable that the following proof does not use the original EUF-CMA definition for mercurials that was presented in [CL19] (see Def. 4.44) but the modified variant from Def. 4.45 which additionally gives the adversary the possibility to observe signatures produced with randomized versions of the challenge secret key (where the randomness is chosen by the adversary itself). This is needed to simulate the oracle access to the $\mathsf{GS.Sign}$ algorithm from Def. 6.1 during the reductions in the following security proof. Note that, as already mentioned in Sect. 4.4, it is left as an open research question whether mercurial signatures that fulfill Def. 4.45 do exist.

Proof. The proof is analogous to the full-traceability proof from [Sch20] for the original group signature construction from [BHKS18]. For brevity, we only provide a proof sketch here. Observe that there are the following two ways for an adversary $\mathcal{A}$ to win the full-traceability game from Def. B.4:

- $\mathcal{A}$ outputs a valid but unopenable signature, which for $\Sigma^{(\mathsf{Merc})}_{\mathsf{GS},B}$ (Def. 6.1) means that $\mathcal{A}$ somehow creates a valid signature $\sigma'_{\mathsf{cert},i}$ for a public key pk' (without knowing the corresponding secret key sk') that is not related to any user's public key pk_i, i.e.
$$\mathsf{pk}' \notin [\mathsf{pk}_i]_{R_{\mathsf{pk}}}$$
- $\mathcal{A}$ outputs a valid signature that opens to a honest user i which is not included in the set of users that $\mathcal{A}$ corrupted during the collusion phase of $\mathsf{Exp}^{\mathsf{trace}}_{\mathcal{A},\Sigma^{(\mathsf{Merc})}_{\mathsf{GS},B}}(\lambda)$.

  For $\Sigma^{(\mathsf{Merc})}_{\mathsf{GS},B}$ (Def. 6.1), this means that $\mathcal{A}$ somehow creates a valid signature for a message (with some appended authentication data) for pk_i without having access to the corresponding secret key sk_i (since user i is honest i.e. its secret has not been corrupted by $\mathcal{A}$).

In the first case, $\mathcal{A}$ would have created a forgery for the mercurial signature key pair that is used to certify the mercurial signature public keys pk_i from the users' personal signing keys in $\mathsf{GS.KGen}$. In the second case, $\mathcal{A}$ would have created a forgery for the mercurial signature key pair $(\mathsf{pk}_i, \mathsf{sk}_i)$ included in the i-th user's personal signing key. These observations allow to reduce the full-traceability (Def. B.4) of the group signature $\Sigma^{(\mathsf{Merc})}_{\mathsf{GS},B}$ from Def. 6.1 to the unforgeability of Σ_{Merc} (Def. 4.45) in a way that is analogous to [Sch20]. □

To finish the discussion of the mercurial-based variant of the group signature from [BHKS18] (Def. 6.1), we prove that it is not fully-anonymous according to Def. B.3.

The problem here is that, under reasonable assumptions, the Merc.ChgRep algorithm allows an adversary to randomize the challenge signature σ in the anonymity game from Def. B.3 and submit it to the opening oracle to learn the identity of the signer behind the challenge signature.

Theorem 6.3 *Let Σ_{Merc}, $\Sigma^{(\mathsf{Merc})}_{\mathsf{GS},B}$ as in Def. 6.1. Assume that for any $\mathsf{pp} \leftarrow\!\$\ \mathsf{Merc.PGen}(\lambda)$, $\mathsf{pk} \in E$, $m \in \mathcal{M}$, $\sigma \in \mathcal{S}$ with $\mathsf{Merc.Vfy}(\mathsf{pk}, m, \sigma) = 1$ there is an efficiently computable signature conversion randomness r_σ such that for $(m', \sigma') \leftarrow\!\$\ \mathsf{Merc.ChgRep}(\mathsf{pk}, m, \sigma, r_\sigma)$ we have*

$$\Pr[m = m' \wedge \sigma \neq \sigma'] = \frac{1}{2} + \eta(\lambda)$$

for a non-negligible function η. Then $\Sigma^{(\mathsf{Merc})}_{\mathsf{GS},B}$ is not fully-anonymous (Def. B.3).

Note that while the requirement for Merc.ChgRep might seem oddly specific at first, the Crites-Lysyanskaya mercurial signature (Def. 4.52) is an example for a mercurial signature scheme with such a signature conversion randomness, namely $r_\sigma = 1 \in \mathbb{Z}_p^*$.

Proof. We prove the claim by constructing an adversary $\mathcal{A}$ with $\mathsf{Adv}^{\mathsf{anon}}_{\mathcal{A},\Sigma^{(\mathsf{Merc})}_{\mathsf{GS},B}}(\lambda)$ not negligible. Let $(\mathsf{gpk}, \mathsf{gmsk}, (\mathsf{gsk}[j])_{j=1}^n) \leftarrow\!\$\ \mathsf{GS.KGen}(\lambda, n)$. On input $(\mathsf{choose}, \mathsf{gpk}, (\mathsf{gsk}[j])_{j=1}^n)$, $\mathcal{A}$ behaves as follows:

1. $\mathcal{A}$ samples and outputs $i_0, i_1 \leftarrow\!\$\ [n]$, $m \leftarrow\!\$\ \mathcal{M}$ together with state information state.
2. $\mathcal{A}$ is eventually called again on input $(\mathsf{guess}, \mathsf{state}, \sigma)$ with $\sigma \leftarrow\!\$\ \mathsf{GS.Sign}(\mathsf{gsk}[i_b], m)$
3. $\mathcal{A}$ parses $\sigma = (\mathsf{pk}', \sigma_{\mathsf{Merc}}, \sigma'_{\mathsf{cert}})$ and computes $M := m||\sigma'_{\mathsf{cert}}||\mathsf{pk}'$.
4. $\mathcal{A}$ computes r_σ from the prerequisites of this theorem.
5. $\mathcal{A}$ computes $(M^*, \sigma^*_{\mathsf{Merc}}) \leftarrow\!\$\ \mathsf{Merc.ChgRep}(\mathsf{pk}', M, \sigma, r_\sigma)$.
6. $\mathcal{A}$ submits $(m, \sigma^* := (\mathsf{pk}', \sigma^*_{\mathsf{Merc}}, \sigma'_{\mathsf{cert}}))$ to the opening oracle, eventually obtaining an identity $k := \mathsf{GS.Open}(\mathsf{gmsk}, m, \sigma^*)$.
7. If $k = i_0$, $\mathcal{A}$ outputs 0, else $\mathcal{A}$ outputs 1.

By prerequisite on the computation of r_σ, $\mathcal{A}$ obviously is a ppt adversary since it only calls ppt algorithms and all other actions it performs can obviously be implemented as ppt algorithms. What is left to analyze is the probability that $k = i_b$ and $(m, \sigma) \neq (m, \sigma^*)$, i.e. that $\mathcal{A}$ indeed does learn the hidden identity i_b behind the challenge signature σ without "cheating", i.e. without querying σ to the opening oracle.

We see that, since both σ and σ^* have the same public key $\mathsf{pk}' \in [\mathsf{pk}_{i_b}]_{R_{\mathsf{pk}}}$ as their first component, the definition of GS.Open yields that

$$\mathsf{GS.Vfy}(\mathsf{gpk}, m, \sigma^*) = 1 \Rightarrow k = i_b$$

$\mathsf{GS.Vfy}(\mathsf{gpk}, m, \sigma^*) = 1$ holds because $(\mathsf{pk}', \sigma'_{\mathsf{cert}})$ is a valid message-signature pair under pk and $(M^*, \sigma^*_{\mathsf{Merc}})$ is a valid message-signature pair under pk' since $(M, \sigma_{\mathsf{Merc}})$ is a valid

pair under pk' and $\mathsf{Merc.ChgRep}$ works correctly according to Def. 4.42. Furthermore, by definition of σ^* we have

$$(m, \sigma^*) \neq (m, \sigma) \Leftrightarrow \sigma^* \neq \sigma \Leftrightarrow \sigma^*_{\mathsf{Merc}} \neq \sigma_{\mathsf{Merc}}$$

Overall this yields that

$$\begin{aligned}\Pr[b' = b] &\geq \Pr[k = i_b \wedge (m, \sigma) \neq (m, \sigma^*)] \\ &\geq \Pr[M^* = M \wedge \sigma^*_{\mathsf{Merc}} \neq \sigma_{\mathsf{Merc}}] \\ &= \frac{1}{2} + \eta(\lambda)\end{aligned}$$

which ultimately yields

$$\mathsf{Adv}^{\mathsf{anon}}_{\mathcal{A}, \Sigma^{(\mathsf{Merc})}_{\mathsf{GS}, B}}(\lambda) = \eta(\lambda)$$

which is not negligible, proving that $\Sigma^{(\mathsf{Merc})}_{\mathsf{GS}, B}$ is not fully-anonymous according to Def. B.3. □

Summing up our results, we have identified the following two crucial problems when it comes to constructing the group signature from [BHKS18] from mercurial signatures (see $\Sigma^{(\mathsf{Merc})}_{\mathsf{GS}, B}$ in Def. 6.1) instead of SFPK and SPS-EQ:

- Mercurial signatures do not have an equivalent to the trapdoor mechanic from SFPK (i.e. the $\mathsf{SFPK.TKGen}$ and $\mathsf{SFPK.ChkRep}$ algorithms) which the opening algorithm from the original group signature construction relies on.
- Under realistic assumptions, the $\mathsf{Merc.ChgRep}$ allows to break the full-anonymity of $\Sigma^{(\mathsf{Merc})}_{\mathsf{GS}, B}$ as seen in the proof of Thm. 6.3. Since SFPKs do not have an equivalent to the $\mathsf{Merc.ChgRep}$ algorithm, the attack from Thm. 6.3 does not work for the original group signature construction by Backes et al. [BHKS18].

Non-interactive zero-knowledge proofs from SFPK Next, we discuss the simulation-sound extractable non-interactive zero-knowledge (SSE-NIZK) proof construction from key-homomorphic signatures (Sect. 4.1.1) that was presented in [DS16]. As part of the research for this thesis, we attempt to do an analogous SSE-NIZK construction based on SFPK (Sect. 4.3).

Let L be some NP-language defined via some NP-relation R. The original SSE-NIZK construction Π_{sse} by Derler and Slamanig [DS16] uses three ingredients, namely an adaptable key-homomorphic signature scheme Σ_{KH}, a strong sEUF-CMA one-time signature Σ_{ot} and a complete, witness indistinguishable NIZK proof system Π for the language L'. L' is defined via the relation R' with

$$((x, \mathsf{cpk}, \mathsf{pk}), (w, \mathsf{csk} - \mathsf{sk})) \in R' \Leftrightarrow (x, w) \in R \vee \mathsf{cpk} = \mathsf{pk} \cdot \mu(\mathsf{csk} - \mathsf{sk})$$

where μ is the secret-to-public-key homomorphism of Σ_{KH} and $\mathsf{pk}, \mathsf{cpk}$ and $\mathsf{sk}, \mathsf{csk}$ are

public and secret keys for Σ_{KH}. So one can either use an "actual" witness w to prove x or prove knowledge of the shift amount $\Delta := \mathsf{csk} - \mathsf{sk}$ derive the key pair $(\mathsf{pk}, \mathsf{sk})$ from the root key pair $(\mathsf{cpk}, \mathsf{csk})$.

To prove a statement x using Π_{sse}, one generates two fresh key pairs $(\mathsf{pk}_{\mathsf{ot}}, \mathsf{sk}_{\mathsf{ot}})$, $(\mathsf{pk}, \mathsf{sk})$ for Σ_{ot} and Σ_{KH}, respectively. The one-time public key $\mathsf{pk}_{\mathsf{ot}}$ is authenticated using sk, producing some certificate σ. A proof π for statement $(x, \mathsf{cpk}, \mathsf{pk})$ is generated using Π and witness $(w, \bot)$. We then sign π, x, pk and σ using the one-time signature key $\mathsf{sk}_{\mathsf{ot}}$, producing a signature σ_{ot}. To simulate a proof, we setup Π in a way that we know the secret key csk corresponding to cpk. The simulation of a proof for $(x, \mathsf{cpk}, \mathsf{pk})$ then uses the witness $(\bot, \mathsf{csk} - \mathsf{sk})$ instead of $(w, \bot)$.

When attempting to replace Σ_{KH} with some SFPK Σ_{SFPK} in the above construction, we naturally adjust the relation R', resulting in the relation R'_{SFPK} defined as follows

$$((x, \mathsf{cpk}, \mathsf{pk}), (w, r)) \in R'_{\mathsf{SFPK}} \Leftrightarrow (x, w) \in R \vee \mathsf{pk} = \mathsf{SFPK.ChgPK}(\mathsf{cpk}, r)$$

Since the equivalence relation R_{pk} that the SFPK Σ_{SFPK} is defined over can have more than one equivalence class, we cannot guarantee that for arbitrary $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{SFPK.KGen}(\mathsf{pp})$ generated for proving a statement x, cpk and pk are in the same equivalence class (i.e. an r changing cpk to pk exists). It is unclear how to generally (i.e. for a black-box SFPK) generate such a key pair $(\mathsf{pk}, \mathsf{sk})$ with $\mathsf{pk} \sim_{R_{\mathsf{pk}}} \mathsf{cpk}$ without access to csk.

The above problem illustrates that in contrast to key-homomorphic signatures, SFPK provide an equivalence relation with small classes, while the "natural" equivalence relation R_μ for key-homomorphic signatures (Lem. 5.7) consists of only one class under reasonable assumptions (see Thm. 5.8 and its proof). Furthermore, the proof of the simulation-sound extractability of the original SSE-NIZK construction Π_{sse} [DS16] makes use of the adaptability of the underlying key-homomorphic signature scheme. We already mentioned in Sect. 5.2.2 that we are not aware of a way to prove that general SFPK provide an adaption algorithm in the spirit of key-homomorphic signatures. Note that the above discussion does not mean that no SSE-NIZK proof constructions from SFPK or related primitives exist.

7 Conclusion and future work

In this final chapter, we summarize the results of this thesis and point out challenges and questions that could be addressed in future work.

Advanced signature framework In Chapter 4, we created a unified definitional framework for key-homomorphic signatures [DS16] (Sect. 4.1), SPS-EQ [HS14, FHS14] (Sect. 4.2), SFPK [BHKS18] (Sect. 4.3) and mercurial signatures [CL19] (Sect. 4.4). Note that while we usually used the original definitions as a basis, we decided to base parts of our SPS-EQ definitions in Sect. 4.2 on a more recent publication [CLPK22] on SPS-EQ. As pointed out by Derler and Slamanig [DS16], we only consider linear shifts as examples for functions to apply to secret keys in the context of key-homomorphic signatures (see syntax in Def. 4.2). Whether there exist key-homomorphic signatures w.r.t. other types of functions as well is left as an open question in [DS16], which we decided to not address in this work for scope reasons. In Chapter 4, we have found the following results about the four individual advanced signature types:

- The secret-to-public-key homomorphism μ of a key-homomorphic signature scheme with canonical key generation (Def. 4.3) is a surjective OWF (see Lem. 4.6 and Lem. 4.5, respectively).
- If one replaces the input shift amount Δ in the $\mathsf{KH.Adapt}$ algorithm with its image $\mu(\Delta)$ under the secret-to-public-key homomorphism μ, the scheme becomes inherently insecure, not even fulfilling UUF-NMA security for digital signatures (Def. 3.18). This result was originally found in [DS16] but we wrote a full proof for the claim (see Lem. 4.8) instead of only providing an attack, without proving that it actually works.
- We have found a condition under which the adaption algorithm $\mathsf{KH.Adapt}$ of a key-homomorphic signature scheme (Def. 4.2) can be used to trivially break unforgeability of the scheme. More precisely, if there exists an efficiently computable shift amount Δ that does not change the input public key pk but changes the input signature σ with non-negligible probability, the output signature σ can be used as a forgery under pk. For a detailed proof, refer to Lem. 4.9.
- We have written a full formal proof of the perfect adaptability (Def. 4.11) and perfect public adaptability (Def. 4.17) of BLS signatures [BLS04] (Def. 4.12) in Lem. 4.14 and Lem. 4.18, respectively. The respective $\mathsf{KH.Adapt}$ and $\mathsf{PKH.Combine}$ algorithms were already given in [DS16], along with a one-line sketches of the distributional argument needed to prove perfect adaptability and perfect public

adaptability (w.r.t. Def. 4.11 and Def. 4.17). We however wrote out the full proofs for the respective statements, including a detailed distributional analysis.

- We discussed different class-hiding definitions for SPS-EQ, namely real-or-random class-hiding (see Def. 4.22, introduced in [FHS14]) and message class-hiding (see Def. 4.23, introduced in [FHS19]). We have proven that under reasonable assumptions, real-or-random class-hiding is stronger than message class-hiding (see Lem. 4.24 and Lem. 4.25 for the formal proof).

- We established a notion of perfect signature adaptation for SPS-EQ (Def. 4.26) that is more general than the original one that was introduced in [BHKS18]. More precisely, our definition allows the $\mathsf{SPSEQ.ChgRep}$ algorithm to compute the new message class representative in a non-deterministic way and also considers SPS-EQ over arbitrary equivalence relations.

- We have formulated and proven the small classes lemma (Lem. 4.32) which gives a necessary condition for the unforgeability of SFPK and mercurial signatures whose key generation algorithms output uniformly random public keys.

- We discussed the wide variety of different (public-key) class-hiding notions for SFPK. We first compared find-original class-hiding (Def. 4.33; introduced in [BHKS18]) to real-or-random public-key class-hiding (Def. 4.47, introduced in [CL19]), finding that real-or-random class-hiding is stronger than the find-original variant (see Lem. 4.36, Lem. 4.35 for a detailed formal proof). We also discussed different levels of insight into the key generation process for adversaries in class-hiding games, leading to weaker tiers of class-hiding definitions. Our results w.r.t. SFPK class-hiding are summarized in Lem. 4.39. Note that an analogous discussion could be done for mercurial signatures (see Rem. 4.49) which we decided not to do here for space reasons. In fact, real-or-random public-key class-hiding was first introduced for mercurial signatures in [CL19] but can be adapted to SFPK in a straightforward manner as we demonstrated in Def. 4.34. Analogously, the find-original public-key class-hiding notion that Backes et al. introduced for SFPK in [BHKS18] can be adapted to mercurial signatures in a straightforward manner, as we demonstrated in Def. 4.48.

- In Lem. 4.43, we gave a constructive proof that the existence of a public key relation $R_{\mathsf{pk}} \subset E \times E$ on the public key space E of an SFPK implies the existence of a secret key equivalence relation $R_{\mathsf{sk}} \subset H \times H$ on its secret key space H which fulfills the corresponding correctness requirement for mercurial signatures. So the explicit mentioning of a secret key relation for mercurials in [CL19] in contrast to the implicit definition for SFPKs (see Lem. 4.43; equivalence relations on the secret key space are not explicitly mentioned in the original SFPK paper [BHKS18]) does not make an actual syntactic difference.

- In Def. 4.45, we added an unforgeability definition for mercurial signatures that is different from the original one that was given in [CL19] (see Def. 4.44). The differ-

ence between the two definitions is that in Def. 4.45, the adversary is additionally given access to a randomize-then-sign oracle inspired by the SFPK unforgeability definition given by Backes et al. [BHKS18] (Def. 4.30) which first randomizes the challenge secret key with the submitted randomness before signing a message with it. This behavior cannot be emulated with the Merc.AdaptSig algorithm since it is unclear whether signatures output by Merc.AdaptSig are identically distributed to fresh signatures generated using Merc.Sign.

Mercurial signatures, SFPK and SPS-EQ After establishing a unified definitional framework for the four signature types and proving some basic results about their security notions (Chapter 4), we analyzed their relations.

We first made the observation that mercurials provide functionality of both SFPK and SPS-EQ, giving rise to the question whether mercurials are a combination of SFPK and SPS-EQ. We addressed this question in Sect. 5.1. For this, we first proved that a secure mercurial signature is a secure SPS-EQ (see Thm. 5.1 in Sect. 5.1.1). Note that a similar claim was already made in [CL19]. However, the authors only considered the syntax and unforgeability notions of the two signature types. We adapted their proof sketch to our definitional framework from Chapter 4 and made a full security analysis of the SPS-EQ $\Sigma_{\mathsf{SPSEQ}}^{(\mathsf{Merc})}$ we constructed from a black-box mercurial signature Σ_{Merc}. More precisely, we have proven the following security implications:

- If Σ_{Merc} is unforgeable (Def. 4.44), then $\Sigma_{\mathsf{SPSEQ}}^{(\mathsf{Merc})}$ is unforgeable (Def. 4.21). Informally, this makes use of the fact that when choosing the public key relation R_{pk} of Σ_{Merc} as the identity relation, the winning conditions of the respective unforgeability games (Def. 4.44 for mercurials and Def. 4.21 for SPS-EQ) are equivalent.

- If Σ_{Merc} is message class-hiding (Def. 4.46) than $\Sigma_{\mathsf{SPSEQ}}^{(\mathsf{Merc})}$ is (message) class-hiding (Def. 4.23). This comes from the fact that the respective security games $\mathsf{Exp}_{\mathcal{A},\Sigma_{\mathsf{Merc}}}^{\mathsf{merc\text{-}mes\text{-}ch}}(\lambda)$ and $\mathsf{Exp}_{\mathcal{A},\Sigma_{\mathsf{SPSEQ}}^{(\mathsf{Merc})}}^{\mathsf{spseq\text{-}ch}}(\lambda)$ proceed identically for any fixed adversary $\mathcal{A}$ and security parameter λ.

- We found a connection between origin-hiding of Merc.ChgRep (Def. 4.51) and perfect signature adaptation of SPS-EQ (Def. 4.26). More precisely, if Merc.Sign outputs uniformly random valid signatures then origin-hiding of Merc.ChgRep implies perfect adaptation of signatures of $\Sigma_{\mathsf{SPSEQ}}^{(\mathsf{Merc})}$. This is a straightforward identical-distribution argument.

Next, as a second contribution in Sect. 5.1.1, we have proven that from a secure mercurial signature Σ_{Merc} with an SFPK-style ChkRep algorithm, one can extract a secure SFPK $\Sigma_{\mathsf{SFPK}}^{(\mathsf{Merc})}$ (Thm. 5.2). More precisely, unforgeability of mercurial signatures (Def. 4.44) implies SFPK unforgeability (Def. 4.30) in a straightforward manner since the winning condition in the mercurial unforgeability game is more restrictive than the one in the SFPK unforgeability game. Furthermore, the literature public-key class-hiding for mercurial signatures from [CL19] (Def. 4.47) translates to adaptive real-or-random

SFPK class-hiding without key corruption (Def. 4.38), a class-hiding notion for SFPK that gives the adversary no insight to the key generation process.

As a last contribution in Sect. 5.1, we constructed a secure mercurial signature Σ^*_{Merc} from a secure black-box SFPK Σ_{SFPK} and Σ_{SPSEQ} (see Thm. 5.3 in Sect. 5.1.2). This integration of Σ_{SFPK} and Σ_{SPSEQ} into a mercurial signature requires strong assumptions on the output distributions of the algorithms of the two signature schemes, so a more intuitive idea of our mercurial signature Σ^*_{Merc} is to perceive it is an SFPK with a ChgRep algorithm in the style of SPS-EQ and an $\mathsf{AdaptSig}$ algorithm in the style of a mercurial signature. Note that in contrast to general mercurial signatures, Σ^*_{Merc} has a ChkRep algorithm in the spirit of SFPK. Furthermore, the $\mathsf{Merc.AdaptSig}$ algorithm can be seen as the "gap" between mercurial signatures and an SFPK-SPS-EQ combination since neither general SFPK nor general SPS-EQ provide a way to publicly adapt signatures on a message m to different public keys using the same randomness used to randomize keys.

Key-homomorphic signatures and SFPK After having analyzed the connection between mercurial signatures [CL19], SFPK [BHKS18] and SPS-EQ [HS14, FHS14] in Sect. 5.1, we turned towards the connection of key-homomorphic signatures [DS16] and SFPK in Sect. 5.2. It is immediate that both of them have a mechanism to efficiently randomize a key pair $(\mathsf{pk}, \mathsf{sk})$ into a new one, furthermore both signature types can be used to construct ring signatures [RST06] (the respective ring signature constructions from key-homomorphic signatures and SFPK can be found in [DS16] and [BHKS18], respectively).

Despite these first-glance resemblances, in Sect. 5.2, key-homomorphic signatures and SFPK turn out to be different primitives. We list the differences briefly in the following.

- Key-homomorphic signatures provide an adaption algorithm $\mathsf{KH.Adapt}$ that allows to publicly adapt a signature σ on a message m that is valid under some public key pk to a new public key pk'. Publicly hereby means that the secret key sk corresponding to pk does not have to be known to the adapting party. It is not possible to emulate such a mechanism with a black-box SFPK in a way that adaption can still be done publicly.

- Key-homomorphic signatures provide a secret-to-public-key homomorphism (Def. 4.1) $\mu : H \to E$ between their secret key space H and their public key space E. The existence of such a homomorphism μ is not implied by the syntax of an SFPK. The individual μ furthermore prescribes how the aforementioned randomization of key pairs works for a specific key-homomorphic signature scheme. More precisely, the new secret key sk' is computed as $\mathsf{sk}' := \mathsf{sk} + \Delta$ and the respective public key is adapted as $\mathsf{pk}' := \mathsf{pk} \cdot \mu(\Delta)$ for a secret key Δ that is seen as a shift amount. For SFPKs, the key pair randomization is done using the algorithms $\mathsf{SFPK.ChgPK}$ and $\mathsf{SFPK.ChgSK}$ with the same randomness r on the public and secret key, respectively. $\mathsf{SFPK.ChgPK}$ and $\mathsf{SFPK.ChgSK}$ can be seen as black boxes whose inner workings are entirely determined by the individual SFPK.

Furthermore, the randomness r does not have to be a secret key as it needs to be for the key pair randomization process of key-homomorphic signatures.

- Secure (more precisely, unforgeable) SFPK provide an equivalence relation on the public key space. This relation has small equivalence classes if the key generation algorithm $\mathsf{SFPK.KGen}$ outputs uniformly random public keys, as proven in the small classes lemma (Lem. 4.32). Thm. 5.8 illustrates that it seems to be hard to construct an SFPK with such a public key relation from key-homomorphic signatures in a black-box way. In this theorem, we have constructed a canonical SFPK $\Sigma^{(\mathsf{KH})}_{\mathsf{SFPK}}$ from a key-homomorphic signature scheme by using the key pair randomization procedure for key-homomorphic signatures to define the key change algorithms $\mathsf{SFPK.ChgSK}$ and $\mathsf{SFPK.ChgPK}$ as follows:

$\mathsf{SFPK.ChgSK}(\mathsf{sk}, r)$	$\mathsf{SFPK.ChgPK}(\mathsf{pk}, r)$
1 : **return** $\mathsf{sk}' := \mathsf{sk} + r$	1 : **return** $\mathsf{pk}' := \mathsf{pk} \cdot \mu(r)$

The resulting SFPK $\Sigma^{(\mathsf{KH})}_{\mathsf{SFPK}}$ is over the equivalence relation R_μ from Lem. 5.7 which is defined as follows:

$$\mathsf{pk} \sim_{R_\mu} \mathsf{pk}' :\Leftrightarrow \exists r \in H : \mathsf{pk}' = \mathsf{pk} \cdot \mu(r)$$

In case of a surjective μ (for examples of such schemes, see Rem. 4.7), the quotient set of this equivalence relation just consists of one equivalence class which means that $\Sigma^{(\mathsf{KH})}_{\mathsf{SFPK}}$ is obviously not unforgeable according to Def. 4.30. This is because any signature for an arbitrary message under an arbitrary key pair can be used as an existential forgery as discussed in Thm. 5.8.

- To conclude the discussion on conceptual differences between SFPK and key-homomorphic signatures, we analyzed an existing SFPK construction, namely the warm-up SFPK $\Sigma^{\mathsf{bck\text{-}wu}}_{\mathsf{SFPK}}$ from [BHKS18] (Def. 5.13), for key-homomorphic properties in Sect. 5.3. We have not found a secret-to-public-key homomorphism μ that fulfills Def. 4.2. We however proved the projection to the public key space, a natural candidate for such a μ, unsuitable in Thm. 5.15.

Applications of the four signature types In Chapter 6, we further illustrated the similarities and differences between the four digital signature types from Chapter 4. We first listed existing construtions of advanced cryptographic primitives from the signature types from Chapter 4 and then gave alternative constructions for some of them that were based on a different signature type than the original one. In the following we briefly list our construction attempts and the insights they gave to the signature types from Chapter 4.

- We attempted to construct the SFPK-and-SPS-EQ group signature from [BHKS18] from mercurial signatures in Chapter 6. Our construction requires the used mercurial to have an SFPK-like ChkRep algorithm for the signature opening process

to work in the same trapdoor-based-way as it did for the original group signature construction from [BHKS18]. Our construction (Def. 6.1) is fully-traceable (Def. B.4) but not fully anonymous (Def. B.3). This is due to the fact that the Merc.ChgRep algorithm allows for a simple efficient attack that breaks anonymity (for details, see Thm. 6.3).

- We discussed why it is not promising to construct SSE-NIZK proof systems from SFPK with an analogous approach to the key-homomorphic-signature-based construction from [DS16]. This discussion illustrated the key differences between key-homomorphic signatures and SFPK (which we already discussed in Sect. 5.2): a key-homomorphic signature scheme provides a KH.Adapt algorithm that allows to publicly adapt signatures while an SFPK provides an equivalence relation on the public key space with small classes.

Bibliography

[AGOT14] Masayuki Abe, Jens Groth, Miyako Ohkubo, and Mehdi Tibouchi. Structure-preserving signatures from type ii pairings. In *Annual Cryptology Conference*, pages 390–407. Springer, 2014. 7, 8, 25, 27

[BCC+09] Mira Belenkiy, Jan Camenisch, Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Hovav Shacham. Randomizable proofs and delegatable anonymous credentials. In *Annual International Cryptology Conference*, pages 108–125. Springer, 2009. 5

[BCC+16] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Essam Ghadafi, and Jens Groth. Foundations of fully dynamic group signatures. In *International Conference on Applied Cryptography and Network Security*, pages 117–136. Springer, 2016. 113

[BEK+20] Jan Bobolz, Fabian Eidens, Stephan Krenn, Daniel Slamanig, and Christoph Striecks. Privacy-preserving incentive systems with highly efficient point-collection. In *Proceedings of the 15th ACM Asia Conference on Computer and Communications Security*, pages 319–333, 2020. 4, 93

[BF20] Balthazar Bauer and Georg Fuchsbauer. Efficient signatures on randomizable ciphertexts. In *International Conference on Security and Cryptography for Networks*, pages 359–381. Springer, 2020. 7

[BFG13] David Bernhard, Georg Fuchsbauer, and Essam Ghadafi. Efficient signatures of knowledge and daa in the standard model. In *International Conference on Applied Cryptography and Network Security*, pages 518–533. Springer, 2013. 7, 8, 9, 25, 27, 90

[BHKS18] Michael Backes, Lucjan Hanzlik, Kamil Kluczniak, and Jonas Schneider. Signatures with flexible public key: Introducing equivalence classes for public keys. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 405–434. Springer, 2018. 1, 3, 4, 9, 21, 35, 43, 44, 48, 49, 57, 58, 69, 80, 87, 88, 89, 90, 91, 93, 94, 95, 96, 97, 99, 101, 102, 103, 104, 105, 106, 111

[BHSB19] Michael Backes, Lucjan Hanzlik, and Jonas Schneider-Bensch. Membership privacy for fully dynamic group signatures. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 2181–2198, 2019. 55

[BLS04] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. *Journal of cryptology*, 17(4):297–319, 2004. 7, 8, 24, 27, 29, 30, 101

[BMW03] Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In *International conference on the theory and applications of cryptographic techniques*, pages 614–629. Springer, 2003. 93, 96, 113

[BS20] Dan Boneh and Victor Shoup. A graduate course in applied cryptography. *Draft 0.5*, 2020. 51

[BSZ05] Mihir Bellare, Haixia Shi, and Chong Zhang. Foundations of group signatures: The case of dynamic groups. In *Cryptographers' Track at the RSA Conference*, pages 136–153. Springer, 2005. 113

[CH91] David Chaum and Eugène van Heyst. Group signatures. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 257–265. Springer, 1991. 93, 94, 113

[CL19] Elizabeth C Crites and Anna Lysyanskaya. Delegatable anonymous credentials from mercurial signatures. In *Cryptographers' Track at the RSA Conference*, pages 535–555. Springer, 2019. 1, 3, 5, 6, 7, 9, 21, 49, 58, 59, 60, 61, 63, 64, 65, 66, 68, 69, 70, 72, 75, 87, 93, 95, 96, 97, 101, 102, 103, 104

[CL20] Elizabeth C Crites and Anna Lysyanskaya. Mercurial signatures for variable-length messages. *Cryptology ePrint Archive*, 2020. 5, 7

[CLPK22] Aisling Connolly, Pascal Lafourcade, and Octavio Perez Kempner. Improved constructions of anonymous credentials from structure-preserving signatures on equivalence classes. In *IACR International Conference on Public-Key Cryptography*, pages 409–438. Springer, 2022. 4, 7, 35, 36, 73, 87, 95, 101

[CM11] Sanjit Chatterjee and Alfred Menezes. On cryptographic protocols employing asymmetric pairings—the role of ψ revisited. *Discrete Applied Mathematics*, 159(13):1311–1322, 2011. 88, 90

[DS16] David Derler and Daniel Slamanig. Key-homomorphic signatures: Definitions and applications to multiparty signatures and non-interactive zero-knowledge. Cryptology ePrint Archive, Paper 2016/792, 2016. `https://eprint.iacr.org/2016/792`. 1, 3, 7, 8, 9, 16, 21, 22, 24, 25, 27, 28, 29, 30, 31, 32, 33, 69, 80, 81, 84, 87, 90, 91, 93, 99, 100, 101, 104, 106

[DS19] David Derler and Daniel Slamanig. Key-homomorphic signatures: definitions and applications to multiparty signatures and non-interactive zero-knowledge. *Designs, Codes and Cryptography*, 87(6):1373–1413, 2019. 1, 13, 14, 21

[FHS14] Georg Fuchsbauer, Christian Hanser, and Daniel Slamanig. Euf-cma-secure structure-preserving signatures on equivalence classes. *IACR Cryptol. ePrint Arch.*, 2014:944, 2014. 1, 3, 4, 9, 21, 35, 36, 37, 69, 87, 93, 101, 102, 104

[FHS19] Georg Fuchsbauer, Christian Hanser, and Daniel Slamanig. Structure-preserving signatures on equivalence classes and constant-size anonymous credentials. *Journal of Cryptology*, 32(2):498–546, 2019. 4, 39, 41, 43, 93, 102

[Fuc14] Georg Fuchsbauer. Breaking existential unforgeability of a signature scheme from asiacrypt 2014. *Cryptology ePrint Archive*, 2014. 3, 4

[Gha16] Essam Ghadafi. Short structure-preserving signatures. In *Cryptographers' Track at the RSA Conference*, pages 305–321. Springer, 2016. 7, 8, 25, 27

[GQ88] Louis Claude Guillou and Jean-Jacques Quisquater. A "paradoxical" indentity-based signature scheme resulting from zero-knowledge. In *Conference on the Theory and Application of Cryptography*, pages 216–231. Springer, 1988. 7, 24, 27, 28

[HS14] Christian Hanser and Daniel Slamanig. Structure-preserving signatures on equivalence classes and their application to anonymous credentials. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 491–511. Springer, 2014. 1, 3, 4, 9, 21, 35, 36, 37, 39, 69, 101, 104

[HS21] Lucjan Hanzlik and Daniel Slamanig. With a little help from my friends: constructing practical anonymous credentials. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 2004–2023, 2021. 4, 5

[IN83] Kazuharu Itakura and Katsuhiro Nakamura. A public-key cryptosystem suitable for digital multisignatures. *NEC Research & Development*, (71):1–8, 1983. 8, 93

[KSD19] Mojtaba Khalili, Daniel Slamanig, and Mohammad Dakhilalian. Structure-preserving signatures on equivalence classes from standard assumptions. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 63–93. Springer, 2019. 4

[KW03] Jonathan Katz and Nan Wang. Efficiency improvements for signature schemes with tight security reductions. In *Proceedings of the 10th ACM conference on Computer and communications security*, pages 155–164, 2003. 7, 25, 27

[MSBM22] Omid Mir, Daniel Slamanig, Balthazar Bauer, and René Mayrhofer. Practical delegatable anonymous credentials from equivalence class signatures. *Cryptology ePrint Archive*, 2022. 4, 7

[PS16] David Pointcheval and Olivier Sanders. Short randomizable signatures. In *Cryptographers' Track at the RSA Conference*, pages 111–126. Springer, 2016. 7, 8, 25, 27

[RST06] Ronald L Rivest, Adi Shamir, and Yael Tauman. How to leak a secret: Theory and applications of ring signatures. In *Theoretical Computer Science*, pages 164–186. Springer, 2006. 4, 7, 93, 104

[SBWP03] Ron Steinfeld, Laurence Bull, Huaxiong Wang, and Josef Pieprzyk. Universal designated-verifier signatures. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 523–542. Springer, 2003. 7, 93

[Sch91] Claus-Peter Schnorr. Efficient signature generation by smart cards. *Journal of cryptology*, 4(3):161–174, 1991. 7, 21, 22, 24, 27

[Sch20] Patrick Schürmann. *A Group Signature Scheme from Flexible Public Key Signatures and Structure-Preserving Signatures on Equivalence Classes*. Universität Paderborn, 2020. 96, 97, 113

[Tod] Peter Todd. Stealth addresses. 2014. *URL: https://lists. linuxfoundation. org/pipermail/bitcoin-dev/2014-January/004020. html (visited on 2017-02-10). APPENDIX Multisignature transactions (←) Overall share (in%)(→) per block 0.2 0.4 0.6 0.8*, 1(50):100. 4, 58, 93

[Wat05] Brent Waters. Efficient identity-based encryption without random oracles. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 114–127. Springer, 2005. 25, 87, 90

Appendix A

Computational Hardness Assumptions

In this chapter, we are going to briefly summarize the computational hardness assumptions that we use in this work. We start with the decisional Diffie-Hellman assumption. It states that given three powers in a cyclic group, it is hard to tell whether the exponent of the third power is the product of the first two exponents or not. It is used in [BHKS18] to prove that the warm-up SFPK construction (see Def. 5.13) is class-hiding according to Def. 4.33.

Definition A.1 (**decisional Diffie-Hellman assumption**) *Let $G = \langle g \rangle$ be a cyclic group of prime order $p \in \mathbb{P}$. We define the following security experiment for G between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$:*

$\mathsf{Exp}^{\mathsf{ddh}}_{\mathcal{A},G}(\lambda)$

1: $\alpha, \beta, \gamma_0 \leftarrow\!\!\$\ \mathbb{Z}_p$
2: $\gamma_1 := \alpha \cdot \beta$
3: $b \leftarrow\!\!\$\ \{0,1\}$
4: $b' \leftarrow\!\!\$\ \mathcal{A}(g^\alpha, g^\beta, g^{\gamma_b})$
5: **return** $b = b'$

We define the advantage of $\mathcal{A}$ in the above security game as

$$\mathsf{Adv}^{\mathsf{ddh}}_{\mathcal{A},G}(\lambda) := |\Pr[\mathsf{Exp}^{\mathsf{ddh}}_{\mathcal{A},G}(\lambda) = 1] - \frac{1}{2}|$$

We say that the decisional Diffie-Hellman assumption *holds for G if for all ppt adversaries $\mathcal{A}$ the advantage $\mathsf{Adv}^{\mathsf{ddh}}_{\mathcal{A},G}(\lambda)$ is negligible.*

We next formally define the symmetric decisional linear assumption as it is used by Backes et al. in [BHKS18] to prove the unforgeability of their warm-up SFPK construction $\Sigma^{\mathsf{bck\text{-}wu}}_{\mathsf{SFPK}}$ (see Thm. 5.14).

Definition A.2 (**symmetric decisional linear assumption**) *Let* $\mathrm{BG} := (G_1, G_2, G_T, p, e, g_1, g_2) \leftarrow\!\!\$\ \mathsf{BGGen}(\lambda, 3)$ *be a bilinear group. We define the following security game for* BG *between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$:*

$\mathsf{Exp}^{\mathsf{s\text{-}d\text{-}lin}}_{\mathcal{A},\mathrm{BG}}(\lambda)$

$f, h, a, c \leftarrow\!\!\$\ \mathbb{Z}_p^*$

$z_0 := a + c, z_1 \leftarrow\!\!\$\ \mathbb{Z}_p$

$b \leftarrow\!\!\$\ \{0,1\}$

$f_1 := g_1^f, h_1 := g_1^h$

$f_2 := g_2^f, h_2 := g_2^h$

$b' \leftarrow\!\!\$\ \mathcal{A}((f_1, h_1, f_1^a, h_1^c, g_1^{z_b}), (f_2, h_2, f_2^a, h_2^c, g_2^{z_b}))$

return 1 **if** $b = b'$

We define the advantage of $\mathcal{A}$ in the above security game as

$$\mathsf{Adv}^{\mathsf{s\text{-}d\text{-}lin}}_{\mathcal{A},\mathrm{BG}}(\lambda) := |\Pr[\mathsf{Exp}^{\mathsf{s\text{-}d\text{-}lin}}_{\mathcal{A},\mathrm{BG}}(\lambda) = 1] - \frac{1}{2}|$$

We say that the symmetric decisional linear assumption *holds for* BG *if for all ppt adversaries $\mathcal{A}$ the advantage $\mathsf{Adv}^{\mathsf{s\text{-}d\text{-}lin}}_{\mathcal{A},\mathrm{BG}}(\lambda)$ is negligible.*

Appendix B

Group signatures

In this section, we will recap the basic syntax and security definitions for group signatures. Group signatures were first introduced by Chaum and van Heist in [CH91]. A group signature involves a group of n signers where each of them has a *personal signing key* $\mathsf{gsk}[i]$ as well as access to the *group public key* gpk. Group member i uses their personal signing key $\mathsf{gsk}[i]$ to sign messages on behalf of the group. Subsequently, any party (in particular, non-group members) can use the group public key gpk to verify that some message was signed by a member of the group. However, without the *group manager secret key* gmsk, it is not possibly to efficiently recover the identity of the signer i who produced some group signature σ on some message m. gmsk is generated by the *group manager*, who is an authority that is not part of the group. Advanced models of group signatures exist that allow to alter the group of signers subsequently as well as splitting the responsibilities of the group manager between multiple entities (see *partially dynamic group signatures* from [BSZ05] and *fully-dynamic group signatures* from [BCC+16]). In this work, we will only deal with *static group signatures* which involve a static group of n signers (i.e. once the group is established, no one can join or leave it). This group is administrated by a single group manager which is responsible for both key generation and opening of signatures. We will use the static group signature definitions given by Schürmann in [Sch20]. Schürmann based his work on [BMW03] which broke down multiple overlapping security notions for group signatures to the two central notions of *full-anonymity* and *full-traceability*.

Before dealing with security, we first formally define the syntax and correctness of a static group signature scheme. For brevity and since we do not use other group signature types in this work, we simply refer to static group signature schemes as group signature schemes in the following.

Definition B.1 (**group signature syntax**) *A* group signature scheme Σ_{GS} *is a tuple of ppt algorithms as follows:*

$\mathsf{GS.KGen}(\lambda, n)$: *probablistic key generation algorithm, on input the security parameter* $\lambda \in \mathbb{N}$*, the group size* $n \in \mathbb{N}$*, it outputs a group public key* gpk*, a group manager secret key* gmsk *and personal signing keys* $(\mathsf{gsk}[j])_{j=1}^{n}$

$\mathsf{GS.Sign}(\mathsf{gsk}[i], m)$: *probablistic signing algorithm, on input a personal signing key* $\mathsf{gsk}[i]$ *of some group member* $i \in [n]$ *and a message* m*, it outputs a signature* σ

$\mathsf{GS.Vfy}(\mathsf{gpk}, m, \sigma)$: *deterministic verification algorithm, on input a group public key* gpk*, a message* m *and a signature* σ*, it outputs a bit* $b \in \{0,1\}$

$\mathsf{GS.Open}(\mathsf{gmsk}, m, \sigma)$: *deterministic opening algorithm, on input a group manager secret key* gmsk*, a message* m *and a signature* σ*, it outputs an identity* $i \in [n]$ *or the error symbol* $\perp \notin [n]$

If not explicitly stated otherwise, we always assume that the algorithms of a group signature scheme are named as in Def. B.1. With the syntax of group signatures defined, we next turn to their correctness. In a nutshell, all honestly generated signatures must be valid and it must be possible to retrieve the signer's identity for them (using gmsk). In this work, we only require the above constraints to hold with high probability.

Definition B.2 (**(computational) group signature correctness**) *Let* Σ_{GS} *be a group signature scheme,* $\lambda \in \mathbb{N}$ *be the security parameter,* $n \in \mathbb{N}$ *be the group size,* $(\mathsf{gpk}, \mathsf{gmsk}, (\mathsf{gsk}[i])_{i=1}^{n}) \leftarrow\!\!\$\ \mathsf{GS.KGen}(\lambda, n)$*,* $i \in [n]$*,* m *be a message.* Σ_{GS} *is* (computationally) correct *if there exists negligible functions* κ_1, κ_2 *such that the following two conditions are met:*

(i) $\Pr[\mathsf{GS.Vfy}(\mathsf{gpk}, m, \mathsf{GS.Sign}(\mathsf{gsk}[i], m)) = 1] = 1 - \kappa_1(\lambda)$

(ii) $\Pr[\mathsf{GS.Open}(\mathsf{gmsk}, m, \mathsf{GS.Sign}(\mathsf{gsk}[i], m)) = i] = 1 - \kappa_2(\lambda)$

We next cover security of group signatures. We begin with the notion of full-anonymity which basically requires that without the group manager secret key gmsk, it is not possible for an adversary to efficiently tell which of the group members produced a given valid group signature. This should even hold if the adversary is allowed to choose two signers, one of which then must produce a signature.

Definition B.3 (**full-anonymity**) *Let* $\lambda \in \mathbb{N}$ *be the security parameter, let* $n \in \mathbb{N}$ *be the group size. We define the following security experiment for a group signature scheme* Σ_{GS} *between an adversary* $\mathcal{A}$ *and a challenger* $\mathcal{C}$*:*

$\mathsf{Exp}^{\mathsf{anon}}_{\mathcal{A},\Sigma_{\mathsf{GS}}}(\lambda)$

1 : $b \leftarrow\!\!\$\ \{0,1\}$
2 : $(\mathsf{gpk}, \mathsf{gmsk}, (\mathsf{gsk}[j])_{j=1}^{n}) \leftarrow\!\!\$\ \mathsf{GS.KGen}(\lambda, n)$
3 : $(\mathsf{state}, i_0, i_1, m) \leftarrow\!\!\$\ \mathcal{A}^{\mathsf{GS.Open}(\mathsf{gmsk},\cdot,\cdot)}(\mathsf{choose}, \mathsf{gpk}, (\mathsf{gsk}[j])_{j=1}^{n})$
4 : $\sigma \leftarrow\!\!\$\ \mathsf{GS.Sign}(\mathsf{gsk}[i_b], m)$
5 : $b' \leftarrow\!\!\$\ \mathcal{A}^{\mathsf{GS.Open}(\mathsf{gmsk},\cdot,\cdot)}(\mathsf{guess}, \mathsf{state}, \sigma)$
6 : **if** $\mathcal{A}$ *did not query* $\mathsf{GS.Open}$ *oracle with* (m, σ) **in** guess *phase* **return** $b' = b$
7 : **else return** 0

We define the advantage of $\mathcal{A}$ *against* Σ_{GS} *in the above security game as*

$$\mathsf{Adv}^{\mathsf{anon}}_{\mathcal{A},\Sigma_{\mathsf{GS}}}(\lambda) := |\Pr[\mathsf{Exp}^{\mathsf{anon}}_{\mathcal{A},\Sigma_{\mathsf{GS}}}(\lambda) = 1] - \frac{1}{2}|$$

We call Σ_{GS} fully-anonymous *if for all ppt adversaries* $\mathcal{A}$ $\mathsf{Adv}^{\mathsf{anon}}_{\mathcal{A},\Sigma_{\mathsf{GS}}}(\lambda)$ *is negligible in* λ*.*

The full-anonymity security experiment $\mathsf{Exp}^{\mathsf{anon}}_{\mathcal{A},\Sigma_{\mathsf{GS}}}(\lambda)$ is split in two phases. In the choose phase, the adversary $\mathcal{A}$ has to choose two identities i_0, i_1, one of which must subsequently produce the challenge signature σ on the challenge message m (which is also chosen by $\mathcal{A}$). While doing so, $\mathcal{A}$ can query the challenger for the signer identities behind arbitrary signatures. In the guess phase, the adversary is provided with a challenge signature σ created by one of the previously chosen signers i_0, i_1. $\mathcal{A}$ must then decide which of those two signers produced σ. While still having access to the GS.Open oracle, $\mathcal{A}$ of course is not allowed to query the challenge message-signature pair to the oracle since this would make guessing the signer identity trivial.

With full-anonymity defined, we next turn towards the other important security requirement for group signatures which is *full-traceability*. The basic idea behind it is that it should be infeasible to create a group signature that cannot be traced back to one of the signers involved in its creation at opening time.

Definition B.4 (**full-traceability**) *Let $\lambda \in \mathbb{N}$ be the security parameter, let $n \in \mathbb{N}$ be the group size. We define the following security experiment for a group signature scheme Σ_{GS} between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$:*

$\mathsf{Exp}^{\mathsf{trace}}_{\mathcal{A},\Sigma_{\mathsf{GS}}}(\lambda)$

$(\mathsf{gpk}, \mathsf{gmsk}, (\mathsf{gsk}[j])_{j=1}^{n}) \leftarrow\!\!\$\ \mathsf{GS.KGen}(\lambda)$
$\mathsf{C} := \emptyset$
$\mathsf{state} \leftarrow\!\!\$\ \mathcal{A}^{\mathsf{GS.Sign}(\mathsf{gsk}[\cdot],\cdot),\mathcal{O}_{\mathsf{C}}(\cdot)}(\mathsf{choose}, \mathsf{state})$
$(m, \sigma) \leftarrow\!\!\$\ \mathcal{A}^{\mathsf{GS.Sign}(\mathsf{gsk}[\cdot],\cdot)}(\mathsf{guess}, \mathsf{state})$
if $\mathsf{GS.Vfy}(\mathsf{gpk}, m, \sigma) = 0$
  return 0
if $\mathsf{GS.Open}(\mathsf{gmsk}, m, \sigma) = \perp$
  return 1
if $\exists i \in [n] : \mathsf{GS.Open}(\mathsf{gmsk}, m, \sigma) = i \wedge i \notin \mathsf{C} \wedge (i, m)$ *not queried by* $\mathcal{A}$ **to** *the* GS.Open *oracle*
  return 1
else return 0

with $\mathcal{O}_{\mathsf{C}}$ being an oracle as follows:

$\mathcal{O}_{\mathsf{C}}(j)$

$\mathsf{C} := \mathsf{C} \cup \{j\}$
return $\mathsf{gsk}[j]$

We define the advantage of $\mathcal{A}$ against Σ_{GS} in the above security game as

$$\mathsf{Adv}^{\mathsf{trace}}_{\mathcal{A},\Sigma_{\mathsf{GS}}}(\lambda) := \Pr[\mathsf{Exp}^{\mathsf{trace}}_{\mathcal{A},\Sigma_{\mathsf{GS}}}(\lambda) = 1]$$

We call Σ_{GS} fully-traceable *if for all ppt adversaries $\mathcal{A}$ $\mathsf{Adv}^{\mathsf{trace}}_{\mathcal{A},\Sigma_{\mathsf{GS}}}(\lambda)$ is negligible in λ.*

The full-traceability experiment $\mathsf{Exp}^{\mathsf{trace}}_{\mathcal{A},\Sigma_{\mathsf{GS}}}(\lambda)$ is also split in two phases. In the choose phase, the adversary can form the set of signers that are involved into the creation of the malicious group signature σ in the subsequent guess phase. A signer j being involved hereby means that $\mathcal{A}$ gains access to their personal signing key $\mathsf{gsk}[j]$. The signers colluding with $\mathcal{A}$ are recorded in the collusion set C. In the subsequent guess phase, $\mathcal{A}$ is tasked with computing a group signature σ that is not traced back to any signer $i \in \mathsf{C}$ by GS.Open. This means that $\mathcal{A}$ must either create a signature that is unopenable or opens to an honest signer $i \notin \mathsf{C}$.

Note that the anonymity and traceability experiments $\mathsf{Exp}^{\mathsf{anon}}_{\mathcal{A},\Sigma_{\mathsf{GS}}}(\lambda)$ and $\mathsf{Exp}^{\mathsf{trace}}_{\mathcal{A},\Sigma_{\mathsf{GS}}}(\lambda)$ from Def. B.3 and Def. B.4 actually also depend on the group size n. To capture this in the respective advantage definitions, it would be necessary to define negligible two argument functions. Since this is just a technical detail that does not change the argumentation in our discussions about group signatures, we omit it in this thesis.