UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft

Fakultät für Elektrotechnik, Informatik und Mathematik
Arbeitsgruppe Codes und Kryptographie

A Group Signature Scheme from Flexible Public Key Signatures and Structure-Preserving Signatures on Equivalence Classes

Bachelor's Thesis

in Partial Fulfillment of the Requirements for the Degree of

Bachelor of Science

by
Patrick Schürmann

submitted to:
Prof. Dr. Johannes Blömer
and
Prof. Dr. Christian Scheideler

Paderborn, October 22, 2020

Contents

1 Introduction

Digital signatures are one of the most important primitives in modern cryptography, since they are needed to ensure message integrity and authenticity which makes them complementary to encryption schemes which preserve confidentiality of messages. However, if it comes to hiding the signers identity, regular digital signature schemes are inadequate since they are constructed in a way that everyone in possession of the signers public key can verify that a certain signature was produced by that very signer. This is not always a desired property, because use cases where the signer of a message wants to stay anonymous clearly exist, e.g. leaking secret information while ensuring its integrity and authenticating it. This motivates the idea of signatures that provide the usual message integrity as well as some level of authenticity but at the same time do not reveal the signers identity.

An example of a primitive with these properties are the so-called *ring signatures* which were first introduced by Rivest et al. [15]. It allows an user to choose any set of possible signers that includes himself and sign any message using his own secret key and all signers public keys. Since public keys are available to everybody who is willing to do a bit of research, this means that a signature can be created without the help and therefore without knowledge or approval of the signers from the chosen set. In addition, no specific public key is needed to verify a given ring signature for a message, so the set of public keys which are part of the signature is the only information about the signer that the verifier has. This concludes that the original signer remains hidden among the set of possible signers he chose. It must be stressed here that ring signatures do not require any setup before signatures can be made (only the public keys used for signing must be available in a public key infrastructure), so with the anonymity of the signer pointed out above this yields that after a signature was created, there is no way to find out who of the possible signers created that very signature. It is clear to see that use cases exist where the signer's anonymity needs to be revoked, for example consider a company member leaking confidential information and authenticating it with a ring signature. This motivates the idea of a primitive where a certain entity can efficiently compute the identity of a signer who created a given signature for a given message. An example for such a primitive are *group signatures* which were first introduced by Chaum and Van Heyst [9].

In the setting of a group signature scheme, there exists a group of n members and one group manager with every member having its own secret signing key. These keys, alongside with the *group public key* and *group manager secret key* are generated by the group manager in a separate setup phase. So in contrast to ring signatures, group signature schemes need dedicated preparation and explicit creation of a group, therefore every group member knows that all the others can produce signatures that the entire

group can be accounted for. This implies a certain level of consent of every group member for every signature another group member might produce.

Using the *group signing algorithm* and their personal secret key, all members can produce a signature which can be checked for validity under the group public key. Thus, if a valid message-signature-pair is seen, all someone can learn from it is that some member of the group signed this message but not who did exactly. This gives signers using the group signature algorithm anonymity among the group members (similiar level of anonymity than granted by ring signatures) which regular digital signature schemes cannot provide. Added to that, if a group member breaks down, all other group members can nevertheless use the group signing algorithm to produce signatures, since just like in a ring signature scheme, they can do so without the help of any other group member. In contrast to ring signatures, an *open algorithm* which is part of the group signature scheme allows the group manager to trace back signatures to their creator by returning his identity on input a signature and the group manager secret key. This means that if needed, the group manager can always revoke the anonymity of the signer of any signature.

2 Current state of research

In this chapter, we will summarize the current state of research on group signatures, signatures with flexible public key (SFPK) and structure-preserving signatures on equivalence classes (SPS-EQ), which are the central primitives for our work in this thesis. We will also explain the three different models of group signatures and give examples of concrete group signature schemes. With that done we will cover the relevant security notions for SFPK, SPS-EQ and group signature schemes. This chapter concludes with a sketch of our contribution in this thesis (see Sect. 2.1).

Different models of group signatures Three different models of group signature schemes exist which are static, partially dynamic and fully dynamic group signatures. The main difference between these three models is the way the list of group members can change over time as well as how the group authorities are split up into multiple entities. In a *static group signature scheme* the number of members of the group does not change so everything concerning the group is set up during the key generation by the group manager. The formal state-of-the-art definition of this model was given by Bellare et al. and can be found in [5] together with the respective attacker model and security definitions. The formal model of partially dynamic group signature schemes was introduced by Bellare, Shi and Zhang in [6]. In contrast to static group signature schemes, these schemes allow for users to become a part of an already-existing group by executing the *join algorithm*, which in general is a protocol with the two group authorities. One of these authorities is responsible for key generation and the other one for the opening of signatures. So we see that unlike in static group signature schemes, there is not only a single group authority anymore, which allows for more fine-grained security notions for partially dynamic group signature schemes with only one of the two tasks key generation and opening being done maliciously. The third variant, the so-called *fully dynamic group signature schemes* also allow revocation of group membership by the group authorities, which makes them the group signature model that is most applicable in practice. The formal state-of-the-art model for this was introduced by Bootle et al. in [8].

The static group signature scheme by Backes et al. Backes et al. [3] provided a generic scheme for the construction of static group signature schemes which relies on a new cryptographic primitive they also introduced, the so-called *signatures with flexible public keys (SFPK)*. We will refer to this scheme by *gFPK-GS*. SFPK itself is independent from group signature schemes but can be used to further enhance anonymity of group members in such a scheme. Backes et al. [3] combine SFPK with *structure-preserving-signatures on equivalence classes (SPS-EQ)* originally presented by Slamanig

and Hanser [12] to define *gFPK-GS*. The basic idea of SFPK is to divide the key space into the equivalence classes of an equivalence relation R and to have algorithms $\mathsf{ChgPK}_{\mathsf{SFPK}}$ and $\mathsf{ChgSK}_{\mathsf{SFPK}}$ to change public and secret key to another representative of the same equivalence class using some randomness r. If the same randomness r is used to change both the public and secret key of a key pair, the key pair stays valid, meaning that all signatures created with the altered secret key are valid under the altered public key. Apart from unforgeablity notions, another security property called *class-hiding* is relevant for signatures with flexible public key. Class-hiding means that given two different public keys, it is hard to tell whether these keys belong to the same class (thus are related) or not. However, using the *check-representative algorithm* $\mathsf{ChkRep}_{\mathsf{SFPK}}$, one can easily verify whether a given public key pk is related to another public key pk' whose equivalence class is encoded by a trapdoor τ. $\mathsf{ChkRep}_{\mathsf{SFPK}}$ takes as input pk and τ.

Structure-preserving signatures on equivalence classes (SPS-EQ) is a primitive that is complementary to the later-introduced signatures with flexible public key, more precisely, instead of the key space, the message space is divided into equivalence classes. Using the *change representative algorithm* $\mathsf{ChgRep}_{\mathsf{SPS}}$ on a message-signature-pair (m, σ), one can easily obtain another representative of the same class as m and at the same time adjust the signature σ so that the new message-signature-pair is valid under the same public key as the original one. On the other hand, a class-hiding property that is analogous to the one for SFPK is defined for SPS-EQ. It means that given two messages it is hard to decide whether they belong to the same equivalence class of a relation R or not.

Each group member's secret key in *gFPK-GS* consists of the SFPK key pair of that user and a certificate for the public key from that key pair, which is an SPS-EQ signature. To sign a message with *gFPK-GS* the user first randomizes his SFPK key pair using $\mathsf{ChgPK}_{\mathsf{SFPK}}$ and $\mathsf{ChgSK}_{\mathsf{SFPK}}$, obtaining a new representative of the equivalence class of his original SFPK public key and a fitting SFPK secret key. He then computes a new certificate for the new public key using $\mathsf{ChgRep}_{\mathsf{SPS}}$ from the SPS-EQ scheme. The resulting signature is a signature of the message, the new public key and its certificate under the new SFPK secret key.

For *gFPK-GS* to be secure in the static model of Bellare et al. [5], strong unforgeability of the SFPK and unforgeability of the SPS-EQ (it must be stressed that even without the word strong, these would be different notions), the respective class-hiding property for both the SFPK and the SPS-EQ as well as perfect adaptation of signatures of the SPS-EQ are required. According to Backes et al. [3], *gFPK-GS* has a shorter signature size than the state-of-the-art scheme introduced by Libert et al. in [13] and can be instantiated from standard assumptions.

Other examples of construction of group signatures When it comes to construction of concrete group signature schemes, various alternatives to proceed exist. In [5], Bellare et al. introduced a static group signature scheme that is based on public key encryption and zero-knowledge proofs. When a group member signs a message using the group sign algorithm, it first signs it alongside with identity information using a regular digital signature scheme, then encrypts that signature using a public key encryption scheme and

the group public key and finally outputs the resulting ciphertext alongside with a zero-knowledge proof that the ciphertext contains what it is supposed to do. In detail this means that it proves knowledge of a certificate on the signers public key (created using a secret key only known to the group manager during the setup phase, so knowledge of this certificate can be seen as a proof of the signers identity and group membership) as well as knowledge of a valid signature of the message to be signed under that very public key. We refer to this type of group signature construction as *sign-and-encrypt-and-prove group signatures*. We see that the encryption in above scheme is done in order to hide the identity of the signer. It is noticeable that in the scheme *gFPK-GS* by Backes et al. [3], the class-hiding property of the SFPK already makes a public key in a specific group signature unlinkable to any group member, which means that no additional encryption is needed.

The partially dynamic scheme proposed by Pointcheval and Sanders in [14], which is based on randomizable signatures can be seen as a way to construct partially dynamic group signature schemes which we will refer to as *commit-and-prove group signatures*. To join a group in their scheme, a user commits himself on his secret key sk and obtains a signature σ on sk by starting a protocol with the group manager in the course of which he only reveals g^{sk} and proves knowledge of sk. To sign a message using the group sign algorithm, he randomizes the signature σ on sk to a new valid signature σ' on sk and then adds it to the actual signature together with a proof of knowledge for sk.

An example for a fully dynamic group signature is the scheme proposed by Backes et al. in [4] which is based on SFPK and SPS-EQ like their static scheme from [3]. In [4], Backes et al. addressed application scenarios where even group membership itself is confidential information by introducing the security notion of *membership privacy* for fully dynamic group signatures. The scheme they introduced in [4] is a generic construction for a membership-private group signature scheme from standard assumptions. More details on membership privacy as well as a sketch how the fully-dynamic scheme by Backes et al. [4] works will be given in Chapter 6.

Security of signatures with flexible public key Apart from class-hiding, a suitable unforgeability notion is needed for SFPK. We will be using the definition that was introduced by Backes et al. in [3] which will be explained in the following paragraph. In contrast to usual unforgeability definitions which consider adversaries that can make only signing queries to a signing oracle $\mathcal{O}$, the adversary $\mathcal{A}$ in the unforgeability game for flexible public key signatures can also query what one might call a *randomize-then-sign-oracle*, which takes as input a message m as well as randomness r and outputs a signature of that message m under the secret key sk' created from the original secret key sk and randomness r using $\mathsf{ChgSK}_{\mathsf{SFPK}}$.

At the beginning of the security game, a key pair $(\mathsf{pk}, \mathsf{sk})$ and a *trapdoor* τ for the class of pk is generated. The adversary is given access to τ, pk as well as a signing and a randomize-then-sign-oracle which both use sk and is required to output a public key pk', a message m^* and a signature σ^*. $\mathcal{A}$ wins the game if and only if m^* has never been submitted to any of the oracles, σ^* is a valid signature on m^* under pk' and pk' belongs

to the same equivalence class as the original public key pk (which can be verified using the trapdoor τ).

An SFPK scheme is called *existentially unforgeable under chosen message attacks* if all probablistic polynomial-time adversaries $\mathcal{A}$ win the game described above with negligible advantage. This means that it is not possible to efficiently compute a candidate forgery for a key pair whose public key belongs to any representative of the equivalence class of a given public key pk. A less strict version of this game leads to the definition of strong existential unforgeability for SFPK. In the respective game, the adversary also wins if he submits a distinct signature for a message that has been signed before.

Security of structure-preserving signatures on equivalence classes As already stated, the class-hiding notion for SPS-EQ is quite analogous to that for SFPK, the only difference being that class-hiding for SPS-EQ considers messages instead of public keys. Another security property of SPS-EQ is the so-called *perfect adaptation of signatures* which is needed to prove security of *gFPK-GS* according to Backes et al. [3]. We will use the definition of perfect adaptation of signatures by Backes et al. [3], which we will explain alongside with an unforgeability definition for SPS-EQ by Slamanig et al. [11] in the next paragraphs. It is important to mention here that the message space in an SPS-EQ scheme is always required to be a set of vectors of group elements. The security game for unforgeability of SPS-EQ according to [11] is very similar to that for unforgeability of regular digital signature schemes. The adversary $\mathcal{A}$ is given the public key pk of a key pair (pk, sk) as well as access to a signing oracle $\mathcal{O}$ that uses sk. He may submit messages as signing queries to that oracle and eventually outputs a candidate forgery (m^*, σ^*). In contrast to the unforgeability game for regular digital signatures, the adversary also loses if m^* belongs to the same class as one of his queries to $\mathcal{O}$. So in order to win the game, the adversary must produce a valid signature for a message m^* of a class that no message has been signed from by the signing oracle. Again, an SPS-EQ scheme is called *existentially unforgeable under chosen message attacks* if the probability of any probablistic polymial-time adversary winning above security game is negligible. An SPS-EQ scheme is said to *perfectly adapt signatures* if for any valid message-signature-pair (M, σ) created using a key pair (pk, sk), the pair (M^r, $\mathsf{Sign}(M^r$, sk)) has the same distribution as the pair $\mathsf{ChgRep}_{\mathsf{SPS}}(M, \sigma, r, \mathsf{pk})$ where r in the call of $\mathsf{ChgRep}_{\mathsf{SPS}}$ is the randomness used to get a new representative of $[M]_R$ and adapting the signature and pk is the public key the old pair is and the new one shall be valid under. This implies that for any randomness r a new signature of the new representative of $[M]_R$ obtained via r is indistinguishable from an adapted signature obtained using $\mathsf{ChgRep}_{\mathsf{SPS}}$. We see that this defines a non-trivial property of $\mathsf{ChgRep}_{\mathsf{SPS}}$.

It is important to note that the SPS-EQ scheme presented by Hanser et al. in [12] was later proven insecure, more precisely, an adaptive chosen message attack against its existential unforgeability which was overlooked in the original work was given by Fuchsbauer in [10]. This issue was addressed by Slamanig et al. in [11] where they propose a new SPS-EQ scheme which they then prove to be EUF-CMA secure. This means that Backes et al. [3] cannot use the SPS-EQ scheme from [12] to securely instantiate

gFPK-GS in the model of Bellare et al. from [5], since the used SPS-EQ scheme has to be existentially unforgeable to fulfill both full-traceability and full-anonymity [3].

Security of group signatures Bellare et al. stressed the need for strong and precise security notions for static group signature schemes in [5], referring to the vast set of informal and overlapping security definitions that existed before above publication. The authors aimed at defining the two main security requirements for static group signatures, namely traceability and anonymity in a way that they imply all of the above-mentioned existing requirements (which also were intended to formalize traceability and anonymity). The resulting definitions of *full-anonymity* and *full-traceability* from [5], which allow for the above-mentioned implications, will be explained in Sect. 3.3. An intuitive idea of anonymity for group signatures would be that it is infeasible to recover a signer's identity from a signature when not in possession of the group manager secret key. However, the exact capabilities of an attacker attempting to break anonymity have not been specified before [5], only informal definitions as e.g. in [1] were given. Bellare et al. [5] consider an attacker $\mathcal{A}$ who needs to tell which one of two given signers produced a signature on a given message. $\mathcal{A}$ also chooses both the message and the two possible signers himself. Furthermore, he is given access to an *opening oracle* $\mathcal{O}$, which returns the identity of the originator of any signature it is queried for. $\mathcal{A}$ can submit arbitrary signatures to $\mathcal{O}$ except for the one he needs to find the originator of.

Considering an adversary with access to an opening oracle captures the possibility that someone who is attempting to break the anonymity of a signature might have obtained information from previous openings he observed. In addition, the adversary $\mathcal{A}$ is given the secret keys of all group members to also model the possibility that he colludes with an arbitrary number of group members.

As already mentioned in the introduction (Chapter 1), anonymity of the user behind any signature can be revoked at any time by the group manager using the group manager secret key and the opening algorithm. In order for this to really deter users from producing malicious signatures, it must be guaranteed that no one can forge a signature that cannot be traced back to its originator at opening time. As for anonymity, only informal descriptions of such a property were given up to [5], which did not exactly model an attacker's capabilities. Bellare et al. [5] consider an attacker $\mathcal{A}$ that has access to the group manager secret key and can choose arbitrary many group members to collude with, meaning that he gains access to their personal secret keys. The authors consider $\mathcal{A}$ to break full traceability of a group signature scheme if he manages to output a forged signature that cannot be traced to any of the group members he colluded with by the open algorithm. It must be stressed that this attacker model of Bellare et al. considers the possibility that the group manager secret key can be compromised as well as that large parts of the group (even all other members) collude with one another, attempting to blame a malicious signature on a non-colluded group member. The case of the entire group working together to forge a signature that cannot be opened is covered as well.

2.1 Our work

In this thesis, we will look at the static group signature scheme presented by Backes et al. in [3] and prove that it is secure in the group signature model presented by Bellare et al. in [5]. As stated above, the construction by Backes et al. is based on signatures with flexible public key (SFPK) and structure-preserving signatures on equivalence classes (SPS-EQ), two primitives that can be seen as extensions to regular digital signature schemes. Details on these primitives as well as formal definitions will follow in Chapter 3, a detailed definition of the static group signature scheme by Backes et al. from [3] can be found in Chapter 4. Backes et al. claim in their paper [3] that their group signature scheme fulfills both the full-anonymity and full-traceability definition established by Bellare et al. [5] and provide a high level proof of each of those two claims. More precisely, they propose that both the used SFPK and SPS-EQ scheme need to be existentially unforgeable under chosen-message attacks for their group signature scheme to be fully-traceable. According to Backes et al. [3] their scheme is furthermore fully-anonymous if the SFPK is strongly existentially unforgeable and class-hiding and the SPS-EQ is existentially unforgeable and perfectly adapts signatures. We add a detailed formal proof of the full-traceability and the full-anonymity of the group signature scheme by Backes et al. [3] in Chapter 5. In Sect. 5.3 we will discuss the additions and changes we made to the extended proof sketches for these security properties which Backes et al. gave in [3].

3 Definitions and notation

In this chapter we formally introduce the concepts of group signatures, signatures with flexible public key (SFPK) and structure-preserving signatures on equivalence classes (SPS-EQ) as well as the respective security notions for these primitives. Before this, we will define the important term of a bilinear group which most of the cryptographic primitives covered in this thesis are based on. To start off, we need to clarify the basic notation used in this thesis.

- By $y \leftarrow_{\$} \mathcal{A}(x)$, we denote that the execution of the probablistic algorithm $\mathcal{A}$ on input x yields result y.
- The set of all elements having positive probability of being output by algorithm $\mathcal{A}$ on input x is denoted by $[\mathcal{A}(x)]$.
- If an algorithm in this thesis is not explicitly said to be probablistic, it is deterministic.
- If algorithm $\mathcal{A}$ has access 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 $x \leftarrow_{\$} \{0,1\}^*$, $\mathcal{A}(x)$ uses internal randomness and terminates in polynomial time in the bitlength of x.
- $r \leftarrow_{\$} S$ means that r is chosen uniformly at random from set S.
- $1_{\mathbb{G}}$ denotes the identity element in group $\mathbb{G}$.
- $[n]$ denotes the set $\{1, \ldots, n\}$.
- If two elements m, n from a set M are in relation R, we denote this by mRn. If $m, n \in M$ are not related via R, we denote this by $m \not R\, n$.
- By $[m]_R$ we denote the equivalence class of element m with respect to equivalence relation R which is defined as $[m]_R := \{n \mid m\ R\ n\}$. It is clear by definition that

  $$m\ R\ n \Leftrightarrow n \in [m]_R \tag{3.1}$$

  holds. In this thesis we will use either one of these two notations depending on which is more intuitive from the context.
- By coin we denote the set from which random values influencing the behavior of a probablistic algorithm of an SFPK (Def. 3.14) or an SPS-EQ (Def. 3.21) scheme

are drawn. The set coin can depend on the message space and the equivalence relation a scheme is defined over.

For completeness, the important notion of negligible functions will be formally introduced in the following definition.

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

$$\forall c > 0 : \exists n_0 \in \mathbb{N} : \forall n \geq n_0 : |f(n)| < \frac{1}{n^c} \tag{3.2}$$

Apart from the usual notions of negligible and polynomially-bounded functions, we need negligibility of two-argument functions in this thesis. The following definition of this property can be found in [5].

Definition 3.2 (**nice functions, negl. two-argument functions**) *A function $n : \mathbb{N} \to \mathbb{N}$ is called* nice *if n is polynomially-bounded and $n(k)$ is computable in polynomial time for every $k \in \mathbb{N}$. A two-argument function $s : \mathbb{N} \times \mathbb{N} \to \mathbb{R}$ is* negligible *if for every nice function $n : \mathbb{N} \to \mathbb{N}$ s_n is negligible, with s_n being defined as*

$$s_n : \mathbb{N} \to \mathbb{R}, k \mapsto s(k, n(k)) \tag{3.3}$$

3.1 Bilinear groups

Since most of the cryptographic primitives covered in this thesis are based on bilinear groups, a formal definition of this important object is given in the following section. We will begin with the definition of bilinear maps since a bilinear map is part of every bilinear group.

Definition 3.3 (**bilinear map**) *Let $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ be groups of prime order p with g_1, g_2 being generators of $\mathbb{G}_1, \mathbb{G}_2$. A* bilinear map *or* pairing *is an efficiently computable mapping $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ that fulfills the following conditions:*

$$\forall (g, h) \in \mathbb{G}_1 \times \mathbb{G}_2, (a, b) \in \mathbb{Z}_p^2 : e(g^a, h^b) = e(g, h)^{a \cdot b} \tag{3.4}$$

and

$$e(g_1, g_2) \text{ is a generator of } \mathbb{G}_T \tag{3.5}$$

The property defined via Eq. (3.4) is called *bilinearity*, the one defined with Eq. (3.5) is *non-degeneracy*. Bilinearity basically means that a mapping is linear in both components, non-degeneracy means that when evaluating a mapping for a pair of generators for the respective groups, a generator for the image space group is returned. The following remark justifies why we can assume the prime order groups in Def. 3.3 to contain generators.

Remark 3.4 *All prime order groups are cyclic. Therefore a generator exists for each of them (more precisely, every element different from their identity element is a generator).*

Proof. This directly follows from Lagrange's theorem which yields that every element in

a prime order group must either be an identity element or a generator (because of the only possible element orders being 1 and the group order p). With the fact that every group contains exactly one identity element and a prime order group must contain more than one element (since 1 is not a prime) we can conclude the claim. □

Next we use Def. 3.3 and Rem. 3.4 to define the term bilinear group.

Definition 3.5 (**bilinear group**) *A tuple* $\mathsf{BG} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$ *is called a* bilinear group *if* p *is a prime,* $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ *are groups of order* p*,* $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ *is a bilinear map and* g_i *is a generator of* $\mathbb{G}_i$*.*

Definition 3.6 (**bilinear group generator**) *A* bilinear group generator *is a deterministic polynomial-time algorithm* BGGen *that on input a security parameter* λ *outputs a bilinear group* $\mathsf{BG} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$ *where* p *is a prime represented by* λ *bits.*

With BGGen being a deterministic algorithm, it can be assumed that a group generated with BGGen on input a public security parameter λ is publicly known. This will be useful when it comes to the formal security proofs for *gFPK-GS* in Chapter 5.

3.2 Group signature schemes

The following definitions of group signature schemes and their correctness are based on those given by Bellare et al. in [5]. Since the group signature scheme introduced in [3] which we will analyze in this thesis is a static scheme, we will only define static group signature schemes formally. Moreover, for simplicity in terminology, we omit the word static when referring to static group signature schemes. We start with defining the syntax of a group signature scheme.

Definition 3.7 (**group signature scheme**) *A* group signature scheme *is a tuple* $\mathsf{GS} = (\mathsf{GKGen}, \mathsf{GSign}, \mathsf{GVf}, \mathsf{Open})$ *of polynomial-time algorithms as follows:*

$\mathsf{GKGen}(\lambda, n)$ *probablistic group key generation algorithm, on input a security parameter* λ *and a group size* n*, it outputs* $(\mathsf{gpk}, \mathsf{gmsk}, \mathsf{gsk})$*, where* gpk *is the group public key,* gmsk *is the group manager secret key,* gsk *is a vector of* n *elements with* $\mathsf{gsk}[\mathsf{i}]$ *being the personal secret signing key of group member* i

$\mathsf{GSign}(\mathsf{gsk}[\mathsf{i}], m)$ *probablistic group signing algorithm, on input the secret signing key* $\mathsf{gsk}[\mathsf{i}]$ *of a group member* $i \in [n]$ *and a message* m*, it outputs a signature of* m *under* $\mathsf{gsk}[\mathsf{i}]$

$\mathsf{GVf}(\mathsf{gpk}, m, \sigma)$ *deterministic verification algorithm, on input the group public key* gpk*, a message* m *and a signature* σ *for* m *it returns* 0 *or* 1

$\mathsf{Open}(\mathsf{gmsk}, m, \sigma)$ *deterministic open algorithm, on input the group manager secret key* gmsk*, a message* m *and a signature* σ *for* m *it returns an identity* $i \in [n]$ *or the error symbol* $\perp \notin [n]$

- *For simplicity, the set of identities in a group of size* n *is always assumed to be* $[n]$*.*

- *We call a signature σ a* true signature *if and only if* $\exists i \in [n]$, *message* $m : \sigma \in [\mathsf{GSign}(\mathsf{gsk}[i], m)]$.
- *Furthermore, we call a signature σ for a message m* valid under gpk *if and only if* $\mathsf{GVf}(\mathsf{gpk}, m, \sigma) = 1$. *If the particular group public key is not important or implicitly known then we will simply call such a signature* valid.
- *We say that a message-signature pair (m, σ)* opens to identity i *if* $\mathsf{Open}(\mathsf{gmsk}, m, \sigma) = i$. *We call (m, σ)* unopenable *if* $\mathsf{Open}(\mathsf{gmsk}, m, \sigma) = \perp$ *and (m, σ) is valid under* gpk. *If the message is not important or obvious then we omit it in above terminology and use the term* unopenable signature *or* signature that opens to i.

Above definition only specifies which algorithms make up a group signature scheme and gives names to important in- and outputs they have. To be able to work with group signature schemes in both theory and practice, we need to define what it means for such a scheme work correctly, which results in the following constraints:

- all true signatures must be valid under gpk
- Open must retrieve the identity of the signers of true signatures

A formalization of these requirements is the following definition.

Definition 3.8 (**(perfect) correctness of a group signature scheme**) *Let* $\mathsf{GS} = (\mathsf{GKGen}, \mathsf{GSign}, \mathsf{GVf}, \mathsf{Open})$ *be a group signature scheme. Let $\lambda \in \mathbb{N}$ security parameter, $n \in \mathbb{N}$ group size,* $(\mathsf{gpk}, \mathsf{gmsk}, \mathsf{gsk}) \leftarrow_{\$} \mathsf{GKGen}(\lambda, n)$. GS *is called* (perfectly) correct *if for all $i \in [n]$ and messages m the following conditions are fulfilled:*

$$\Pr[\mathsf{GVf}(\mathsf{gpk}, m, \mathsf{GSign}(\mathsf{gsk}[i], m)) = 1] = 1 \tag{3.6}$$

and

$$\Pr[\mathsf{Open}(\mathsf{gmsk}, m, \mathsf{GSign}(\mathsf{gsk}[i], m)) = i] = 1 \tag{3.7}$$

It is possible to relax the above definition of perfect correctness to a certain extent, such that schemes which have a negligible chance of failing to meet above requirements are still considered correct. This results in the following definition of *computational correctness*.

Definition 3.9 (**computational correctness of a group signature scheme**) *Let* $\mathsf{GS} = (\mathsf{GKGen}, \mathsf{GSign}, \mathsf{GVf}, \mathsf{Open})$ *be a group signature scheme. Let $\lambda \in \mathbb{N}$ security parameter, $n \in \mathbb{N}$ group size,* $(\mathsf{gpk}, \mathsf{gmsk}, \mathsf{gsk}) \leftarrow_{\$} \mathsf{GKGen}(\lambda, n)$. GS *is called* (computationally) correct *if for all $i \in [n]$ and messages m there exist negligible functions κ_1, κ_2 such that the following conditions are fulfilled:*

$$\Pr[\mathsf{GVf}(\mathsf{gpk}, m, \mathsf{GSign}(\mathsf{gsk}[i], m)) = 1] = 1 - \kappa_1(\lambda) \tag{3.8}$$

and

$$\Pr[\mathsf{Open}(\mathsf{gmsk}, m, \mathsf{GSign}(\mathsf{gsk}[i], m)) = i] = 1 - \kappa_2(\lambda) \tag{3.9}$$

It is plain to see that perfect correctness implies computational correctness (since the zero function clearly is negligible). In Sect. 4.2 we will discuss under which assumptions we can prove perfect and computational correctness of *gFPK-GS*.

3.3 Security of group signature schemes

The following definitions of the group signature security requirements full-anonymity and full-traceability are based on the ones given in [5]. As stated in Chapter 2, these two are the only relevant security properties for group signature security since all others are implied by one of them. We will start with defining full-anonymity of a group signature scheme. The idea of this security property is that an adversary not in possession of the group manager secret key cannot efficiently determine the identity of the signer who created a given group signature. This should hold even in the case that the adversary has access to the personal secret keys of all group members. We see that hereby the case of collusion with an arbitrary subset of the group members is captured. The formalization chosen by Bellare et al. [5] describes the challenge for this adversary as a distinguishing game where two different identities are chosen by the adversary and a group signature is created with the key material for one of those identities.

Definition 3.10 (**full-anonymity**) *Let $\lambda \in \mathbb{N}$ be a security parameter, $n \in \mathbb{N}$ group size. We define the following security experiment for a group signature scheme $GS = (\mathsf{GKGen}, \mathsf{GSign}, \mathsf{GVf}, \mathsf{Open})$ and an adversary $\mathcal{A}$:*

$Exp^{\mathsf{anon}-b}_{GS,\mathcal{A}}(\lambda, n)$

1: $(\mathsf{gpk}, \mathsf{gmsk}, \mathsf{gsk}) \leftarrow_\$ \mathsf{GKGen}(\lambda, n)$
2: $(\mathsf{St}, i_0, i_1, m) \leftarrow_\$ \mathcal{A}^{\mathsf{Open}(\mathsf{gmsk},\cdot,\cdot)}(\mathsf{choose}, \mathsf{gpk}, \mathsf{gsk})$
3: $\sigma \leftarrow_\$ \mathsf{GSign}(\mathsf{gsk}[\mathsf{i_b}], m)$
4: $d \leftarrow_\$ \mathcal{A}^{\mathsf{Open}(\mathsf{gmsk},\cdot,\cdot)}(\mathsf{guess}, \mathsf{St}, \sigma)$
5: **if** $\mathcal{A}$ *did not query Open-oracle with* (m, σ) **in** *guess phase* **then return** d
6: **else return** 0

We define the advantage of $\mathcal{A}$ in breaking full-anonymity of GS as

$$Adv^{\mathsf{anon}}_{GS,\mathcal{A}}(\lambda, n) = |\Pr[Exp^{\mathsf{anon}-0}_{GS,\mathcal{A}}(\lambda, n) = 1] - \Pr[Exp^{\mathsf{anon}-1}_{GS,\mathcal{A}}(\lambda, n) = 1]| \tag{3.10}$$

We say that GS is fully-anonymous *if for all PPT-adversaries $\mathcal{A}$ $Adv^{\mathsf{anon}}_{GS,\mathcal{A}}(\lambda, n)$ is negligible in λ.*

Apart from minor notational changes, above definition is identical to the one that can be found in [5]. Above security experiment is split in two phases, the choose-phase and the guess-phase, indicated by the parameter passed to $\mathcal{A}$. The parameter St $\mathcal{A}$ is passed in the guess-phase holds state information from the choose-phase like for example the queries $\mathcal{A}$ already made to it's opening oracle. $\mathcal{A}$ is given access to an opening oracle to model the fact that $\mathcal{A}$ can learn information from openings it observed that can

influence its choice of identities i_0, i_1 and message m as well as the guess d. A message-signature pair $(\tilde{m}, \tilde{\sigma})$ that $\mathcal{A}$ submits to the open oracle is called an *opening query.* $\mathcal{A}$ is of course not allowed to query the opening oracle for the challenge pair (m, σ) since this would make the game entirely trivial. Note that if $\mathcal{A}$ queries its oracle for (m, σ) nevertheless the experiment will always return 0, so the adversaries advantage would be 0 if he always did so. In the later proof of full-anonymity of *gFPK-GS* (see Thm. 5.3), the bit guessing variant of above security game will be far more convenient than above distinguishing game (a discussion of this can be found in Sect. 3.7). In the following we will define the bit guessing game for full-anonymity and prove that both games lead to equivalent security requirements, so every adversary with non-negligible advantage in the distinguishing game also has non-negligible advantage in the bit guessing game and vice versa.

Definition 3.11 (**full-anonymity (bit guessing variant)**) *Let $\lambda \in \mathbb{N}$ be a security parameter, $n \in \mathbb{N}$ group size. We define the following security experiment for a group signature scheme $GS = (\mathsf{GKGen}, \mathsf{GSign}, \mathsf{GVf}, \mathsf{Open})$ and an adversary $\mathcal{A}$:*

$Exp^{\mathsf{anon}^*}_{GS,\mathcal{A}}(\lambda, n)$

1: $b \leftarrow\!\!\$\ \{0,1\}$
2: $(\mathsf{gpk}, \mathsf{gmsk}, \mathsf{gsk}) \leftarrow\!\!\$\ \mathsf{GKGen}(\lambda, n)$
3: $(\mathsf{St}, i_0, i_1, m) \leftarrow\!\!\$\ \mathcal{A}^{\mathsf{Open}(\mathsf{gmsk},\cdot,\cdot)}(\mathsf{choose}, \mathsf{gpk}, \mathsf{gsk})$
4: $\sigma \leftarrow\!\!\$\ \mathsf{GSign}(\mathsf{gsk}[\mathsf{i_b}], m)$
5: $d \leftarrow\!\!\$\ \mathcal{A}^{\mathsf{Open}(\mathsf{gmsk},\cdot,\cdot)}(\mathsf{guess}, \mathsf{St}, \sigma)$
6: **if** $\mathcal{A}$ *did not query Open-oracle with* (m, σ) **in** *guess phase* **then return** $d = b$
7: **else return** 0

We define the bit guessing advantage of $\mathcal{A}$ in breaking full-anonymity of GS as

$$Adv^{\mathsf{anon}^*}_{GS,\mathcal{A}}(\lambda, n) = |\Pr[Exp^{\mathsf{anon}^*}_{GS,\mathcal{A}}(\lambda, n) = 1] - \frac{1}{2}| \tag{3.11}$$

We say that GS is fully-anonymous (in the bit guessing variant) *if for all PPT-adversaries $\mathcal{A}$ $Adv^{\mathsf{anon}^*}_{GS,\mathcal{A}}(\lambda, n)$ is negligible in λ.*

The phases of the game, the input to adversary $\mathcal{A}$ in both of the phases as well as the oracles $\mathcal{A}$ has access to are the same as in the distinguishing variant of the full-anonymity game (Def. 3.10). Next we will formally state and prove the equivalence of the two full-anonymity definitions.

Theorem 3.12 *Let GS be a group signature scheme, $\lambda \in \mathbb{N}$ security parameter, $n \in \mathbb{N}$ group size. For a PPT adversary $\mathcal{A}$ we have*

$$Adv^{\mathsf{anon}}_{GS,\mathcal{A}}(\lambda, n) = 2 \cdot Adv^{\mathsf{anon}^*}_{GS,\mathcal{A}}(\lambda, n) \tag{3.12}$$

This yields that GS is fully-anonymous if and only if GS is fully-anonymous in the bit guessing variant.

Proof. The proof is analogous to the one of Theorem 2.10 in [7], where a corresponding theorem was proven for semantic security of computational ciphers. Let W_b denote the event that $\mathcal{A}$ outputs 1 in $Exp^{\mathsf{anon}-b}_{GS,\mathcal{A}}(\lambda, n)$ for $b \in \{0, 1\}$. The probabilities considered below are with respect to $Exp^{\mathsf{anon}^*}_{GS,\mathcal{A}}(\lambda, n)$. If we condition on $b = 0$, all random variables computed throughout this experiment are exactly like they are in $Exp^{\mathsf{anon}-0}_{GS,\mathcal{A}}(\lambda, n)$ (analogous for $b = 1$). So with d denoting the output of $\mathcal{A}$ we get

$$\Pr[d = 1 \mid b = 0] = \Pr[W_0] \text{ and } \Pr[d = 1 \mid b = 1] = \Pr[W_1]$$

Analyzing the winning probability of $\mathcal{A}$ in the bit guessing game yields

$$\begin{aligned}
\Pr[d = b] =& \Pr[d = 0 \mid b = 0] \cdot \Pr[b = 0] + \Pr[d = 1 \mid b = 1] \cdot \Pr[b = 1] \\
=& \Pr[d = 0 \mid b = 0] \cdot \frac{1}{2} + \Pr[d = 1 \mid b = 1] \cdot \frac{1}{2} \\
=& \frac{1}{2}(\Pr[d = 0 \mid b = 0] + \Pr[d = 1 \mid b = 1]) \\
=& \frac{1}{2}(1 - \Pr[d = 1 \mid b = 0] + \Pr[d = 1 \mid b = 1]) \\
=& \frac{1}{2}(1 - \Pr[W_0] + \Pr[W_1])
\end{aligned}$$

Inserting this into the definition of the bit guessing advantage yields

$$\begin{aligned}
Adv^{\mathsf{anon}^*}_{GS,\mathcal{A}}(\lambda, n) =& |\Pr[d = b] - \frac{1}{2}| \\
=& |\frac{1}{2}(1 - \Pr[W_0] + \Pr[W_1]) - \frac{1}{2}| \\
=& |\frac{1}{2}(-\Pr[W_0] + \Pr[W_1])| \\
=& \frac{1}{2}|-\Pr[W_0] + \Pr[W_1]| \\
=& \frac{1}{2}|\Pr[W_0] - \Pr[W_1]| \\
=& \frac{1}{2} Adv^{\mathsf{anon}}_{GS,\mathcal{A}}(\lambda, n)
\end{aligned}$$

which is equivalent to

$$Adv^{\mathsf{anon}}_{GS,\mathcal{A}}(\lambda, n) = 2 \cdot Adv^{\mathsf{anon}^*}_{GS,\mathcal{A}}(\lambda, n)$$

Since $Adv^{\mathsf{anon}}_{GS,\mathcal{A}}(\lambda, n)$ is negligible if and only if $2 \cdot Adv^{\mathsf{anon}^*}_{GS,\mathcal{A}}(\lambda, n)$ is negligible, we get that GS is fully-anonymous if and only if GS is fully-anonymous in the bit guessing variant. □

The definition of full-anonymity used by Backes et al. in [3] is based on the definition given by Bellare et al. in [5] but contains a major flaw with respect to advantage definition which basically renders it useless for making any meaningful statement about

a group signature schemes security. A detailed description and discussion of said flaw can be found in Sect. 3.7.2.

The other relevant security requirement for group signature schemes is full-traceability. As already mentioned in the introductory chapters, the basic idea behind this definition is that there shall be no efficient way to create a group signature that is traced back to a user not involved in it's creation at opening time.

Definition 3.13 (**full-traceability**) *Let $\lambda \in \mathbb{N}$ be the security parameter, $n \in \mathbb{N}$ group size. We define the following security experiment for a group signature scheme $GS = (\mathsf{GKGen}, \mathsf{GSign}, \mathsf{GVf}, \mathsf{Open})$ and an adversary $\mathcal{A}$:*

$Exp^{\mathsf{trace}}_{GS,\mathcal{A}}(\lambda, n)$

1: $(\mathsf{gpk}, \mathsf{gmsk}, \mathsf{gsk}) \leftarrow_{\$} \mathsf{GKGen}(\lambda, n)$
2: $\mathsf{C} = \emptyset$
3: $\mathsf{St} := (\mathsf{gpk}, \mathsf{gmsk})$
4: $\mathsf{St} \leftarrow_{\$} \mathcal{A}^{\mathsf{GSign}(\mathsf{gsk}[\cdot],\cdot),\mathcal{O}_{\mathsf{C}}(\cdot)}(\mathsf{choose}, \mathsf{St})$
5: $(m, \sigma) \leftarrow_{\$} \mathcal{A}^{\mathsf{GSign}(\mathsf{gsk}[\cdot],\cdot)}(\mathsf{guess}, \mathsf{St})$
6: **if** $\mathsf{GVf}(\mathsf{gpk}, m, \sigma) = 0$
7: **return** 0
8: **if** $\mathsf{Open}(\mathsf{gmsk}, m, \sigma) = \perp$
9: **return** 1
10: **if** $\exists i \in [n] : \mathsf{Open}(\mathsf{gmsk}, m, \sigma) = i \wedge i \notin \mathcal{C} \wedge i, m$ *not queried by* $\mathcal{A}$ **to** *it's oracle*
11: **return** 1
12: **else**
13: **return** 0

with $\mathcal{O}_{\mathsf{C}}(\cdot)$ being an oracle as of the following:

$\mathcal{O}_{\mathsf{C}}(j)$

1: $\mathsf{C} := \mathsf{C} \cup \{j\}$
2: **return** $\mathsf{gsk}[\mathsf{j}]$

We define the advantage of $\mathcal{A}$ in breaking full-traceability of GS as

$$Adv^{\mathsf{trace}}_{GS,\mathcal{A}}(\lambda, n) := \Pr[Exp^{\mathsf{trace}}_{GS,\mathcal{A}}(\lambda, n) = 1] \tag{3.13}$$

We say that GS is fully-traceable *if for all PPT-adversaries $\mathcal{A}$ $Adv^{\mathsf{trace}}_{GS,\mathcal{A}}(\lambda, n)$ is negligible in λ.*

St is a variable that stores state information over the course of the game, i.e. collusion and signing queries made by $\mathcal{A}$. Above game is split in two phases: the choose-phase (or collusion phase) and the guess-phase. In the collusion phase the adversary can query identities $j \in [n]$ to the *collusion oracle* $\mathcal{O}_{\mathsf{C}}(\cdot)$ which responds with the personal secret key $\mathsf{gsk}[\mathsf{j}]$ of group member j. We will call these queries *collusion queries*. This models

the adversary building up it's set C of colluding group members by gaining access to these group members' secret keys. We will shortly refer to this set as the *collusion set* of $\mathcal{A}$. In the choose-phase, $\mathcal{A}$ also has access to a group sign oracle which on input an identity j and a message m returns a signature of that message created by the j-th group member. In the guess-phase, $\mathcal{A}$ computes his candidate forgery while having access to the above-mentioned group sign oracle only. $\mathcal{A}$ wins the game if and only if he forged a valid signature which opens to no identity contained in the collusion set C. Above definition of full-traceability is basically identical to the one given by Bellare et al. in [5], the only difference being how the process of the adversary constructing its collusion set is described. In this thesis, we chose to introduce a collusion oracle which reveals the corresponding secret key when queried with an identity. Despite this is only a notational change, it allows for much easier description of reduction algorithms in Chapter 5. For completeness, the original definition by Bellare et al. can be found in Sect. 3.7.1. Apart from notational changes, Backes et al. [3] use the definition for full-traceability that was established by Bellare et al. in [5].

3.4 Signatures with flexible public keys

In this section, the formal definitions of syntax and security of signatures with flexible public key (SFPK), the first central building block of the group signature scheme *gFPK-GS*, and its security properties will be given. The original definitions were given by Backes et al. in [3]. The basic idea behind SFPK is to divide the public key space into equivalence classes of an equivalence relation R and make it possible to quickly obtain a new valid key pair $(\mathsf{pk}', \mathsf{sk}')$ from a given pair $(\mathsf{pk}, \mathsf{sk})$ such that $\mathsf{pk}' \in [\mathsf{pk}]_R$. The relation R of a given SFPK scheme is implicitly defined by definition of that scheme, the key space of an SFPK scheme is also defined implicitly by the output range of the key generation algorithm $\mathsf{KGen}_{\mathsf{SFPK}}$. An important security property of SFPK schemes is the notion of class-hiding which means that it shall be computationally infeasible to tell if two given SFPK public keys are related via relation R or not. We will start this chapter with defining the syntax of an SFPK scheme.

Definition 3.14 (**SFPK scheme**) *Let R be an equivalence relation. A* signature scheme with flexible public keys with equivalence relation R over the key space *is a tuple* $\Pi_{\mathsf{SFPK}}=($$\mathsf{KGen}_{\mathsf{SFPK}}$, $\mathsf{TKGen}_{\mathsf{SFPK}}$, $\mathsf{Sign}_{\mathsf{SFPK}}$, $\mathsf{ChkRep}_{\mathsf{SFPK}}$, $\mathsf{ChgPK}_{\mathsf{SFPK}}$, $\mathsf{ChgSK}_{\mathsf{SFPK}}$, $\mathsf{Vf}_{\mathsf{SFPK}}$*) of PPT-algorithms as follows:*

$\mathsf{KGen}_{\mathsf{SFPK}}(\lambda, \omega)$ *probablistic key generation algorithm, takes as input the security parameter λ and randomness $\omega \leftarrow_{\$} \mathsf{coin}$ and outputs a key pair $(\mathsf{pk}, \mathsf{sk})$ consisting of a public and a secret key*

$\mathsf{TKGen}_{\mathsf{SFPK}}(\lambda, \omega)$ *probablistic key generation algorithm, takes as input the security parameter λ and randomness $\omega \leftarrow_{\$} \mathsf{coin}$ and outputs a key pair $(\mathsf{pk}, \mathsf{sk})$ consisting of a public and a secret key and a* trapdoor τ *for the class $[\mathsf{pk}]_R$ of pk*

$\mathsf{Sign}_{\mathsf{SFPK}}(\mathsf{sk}, m)$ *probablistic signing algorithm, takes as input a secret key* sk *and a message* m*, outputs a signature* σ *for* m

$\mathsf{ChkRep}_{\mathsf{SFPK}}(\tau, \mathsf{pk})$ *on input a trapdoor* τ *for the equivalence class w.r.t* R *of a public key* pk' *and a public key* pk*, this algorithm returns* 0 *or* 1

$\mathsf{ChgPK}_{\mathsf{SFPK}}(\mathsf{pk}, r)$ *on input a public key* pk *and randomness* $r \leftarrow_{\$} \mathsf{coin}$*, this algorithm returns a public key* pk'*. We call* $\mathsf{pk}' \leftarrow \mathsf{ChgPK}_{\mathsf{SFPK}}(\mathsf{pk}, r)$ *a* randomized version of pk*.*

$\mathsf{ChgSK}_{\mathsf{SFPK}}(\mathsf{sk}, r)$ *on input a secret key* sk *and randomness* r*, this algorithm returns a randomized secret key* sk'

$\mathsf{Vf}_{\mathsf{SFPK}}(\mathsf{pk}, m, \sigma)$ *on input a public key* pk*, a message* m *and a candidate signature* σ *for* m*, this algorithm returns* 0 *or* 1

For an SFPK scheme to work correctly, the following constraints must be fulfilled:

- key pairs generated using $\mathsf{TKGen}_{\mathsf{SFPK}}$ are computationally indistinguishable from those generated using $\mathsf{KGen}_{\mathsf{SFPK}}$
- randomizing both the public key and the secret key of a key pair using the same randomness in $\mathsf{ChgPK}_{\mathsf{SFPK}}, \mathsf{ChgSK}_{\mathsf{SFPK}}$ respectively always yields a valid key pair
- $\mathsf{ChkRep}_{\mathsf{SFPK}}$ tells whether a given public key pk' is in the equivalence class of a public key pk (using a trapdoor τ for the class $[\mathsf{pk}]_R$)
- $\mathsf{ChgPK}_{\mathsf{SFPK}}$ returns a different representative of the equivalence class of the input public key

Above conditions are formalized in the following definition.

Definition 3.15 (**correct SFPK scheme**) *A signature scheme with flexible public keys* $\Pi_{\mathsf{SFPK}} = (\mathsf{KGen}_{\mathsf{SFPK}}, \mathsf{TKGen}_{\mathsf{SFPK}}, \mathsf{Sign}_{\mathsf{SFPK}}, \mathsf{ChkRep}_{\mathsf{SFPK}}, \mathsf{ChgPK}_{\mathsf{SFPK}}, \mathsf{ChgSK}_{\mathsf{SFPK}}, \mathsf{Vf}_{\mathsf{SFPK}})$ *with equivalence relation* R *over the key space is* correct *if the following conditions are fulfilled for any* $\lambda \in \mathbb{N}$*, key-changing randomness* $r \leftarrow_{\$} \mathsf{coin}$*, key-generation randomness* $\omega \leftarrow_{\$} \mathsf{coin}$*:*

- *common distribution of first two entries of* $(\mathsf{pk}, sk) \leftarrow \mathsf{KGen}_{\mathsf{SFPK}}(\lambda, \omega)$, $(\mathsf{pk}, \mathsf{sk}, \tau) \leftarrow \mathsf{TKGen}_{\mathsf{SFPK}}(\lambda, \omega)$ *is identical*
- *for all key pairs* $(\mathsf{pk}, \mathsf{sk}) \leftarrow_{\$} \mathsf{KGen}_{\mathsf{SFPK}}(\lambda, \omega)$, $\mathsf{pk}' = \mathsf{ChgPK}_{\mathsf{SFPK}}(pk, r)$, $\mathsf{sk}' = \mathsf{ChgSK}_{\mathsf{SFPK}}(sk, r)$ *we have*

$$\Pr[\mathsf{Vf}_{\mathsf{SFPK}}(\mathsf{pk}, m, \mathsf{Sign}_{\mathsf{SFPK}}(\mathsf{sk}, m)) = 1] = 1 \tag{3.14}$$

and

$$\Pr[\mathsf{Vf}_{\mathsf{SFPK}}(\mathsf{pk}', m, \mathsf{Sign}_{\mathsf{SFPK}}(\mathsf{sk}', m)) = 1] = 1 \tag{3.15}$$

- *for all* $(\mathsf{sk}, \mathsf{pk}, \tau) \leftarrow_{\$} \mathsf{TKGen}_{\mathsf{SFPK}}(\lambda, \omega)$ *and all public keys* pk' *we have*

$$\mathsf{ChkRep}_{\mathsf{SFPK}}(\tau, \mathsf{pk}') = 1 \Leftrightarrow \mathsf{pk}' \in [\mathsf{pk}]_R \tag{3.16}$$

- *for all public keys* pk *we have*

$$\mathsf{ChgPK}_{\mathsf{SFPK}}(\mathsf{pk}, r) =: \mathsf{pk}' \in [\mathsf{pk}]_R \tag{3.17}$$

  with $\mathsf{pk}' \neq \mathsf{pk}$.

Note that even in a correct SFPK scheme, not every other representative of the equivalence class of a public key pk has to be obtainable via $\mathsf{ChgPK}_{\mathsf{SFPK}}$, so in formulas we have

$$\{\mathsf{pk}' \mid \exists r \in \mathsf{coin} : \mathsf{pk}' = \mathsf{ChgPK}_{\mathsf{SFPK}}(\mathsf{pk}, r)\} \subseteq [\mathsf{pk}]_R \tag{3.18}$$

As far as we know, all existing instantiations of SFPK can be instantiated using bilinear groups of type 3. The equivalence relation which Π_{SFPK} is defined over is left arbitrary in above definition. A discussion about restrictions to the used equivalence relation when it comes to practical use of SFPK schemes can be found in Sect. 3.6.

In the following, we will formally define security for SFPK schemes. The first important security notion for SFPK schemes is the notion of class-hiding. As stated in the introduction, the basic idea behind this security property is that without the respective trapdoor, there should be no efficient way to tell whether two given public keys are related or not. This is formalized as a bit guessing game where the adversary has to tell which of two public keys is related to the challenge public key. For that, it is given access to the two public keys potentially related to the challenge key and the corresponding secret keys. The secret key corresponding to the challenge public key cannot be accessed directly but the adversary can query messages to a signing oracle which uses that secret key. In the game in below definition, which was introduced by the inventors of SFPK in [3], the adversary is given access to the randomness ω_0, ω_1 which was used to generate the two key pairs. Depending on the particular SFPK, the adversary might learn additional information about the key pairs from this, for example the discrete logarithm of their components in a scheme which uses powers of a certain generator of a prime order group as keys.

Definition 3.16 (**class-hiding SFPK**) *Let* $\lambda \in \mathbb{N}$ *be the security parameter. For an SFPK scheme*
$\Pi_{\mathsf{SFPK}} = (\mathsf{KGen}_{\mathsf{SFPK}}, \mathsf{TKGen}_{\mathsf{SFPK}}, \mathsf{Sign}_{\mathsf{SFPK}}, \mathsf{ChkRep}_{\mathsf{SFPK}}, \mathsf{ChgPK}_{\mathsf{SFPK}}, \mathsf{ChgSK}_{\mathsf{SFPK}}, \mathsf{Vf}_{\mathsf{SFPK}})$
and an adversary $\mathcal{A}$ *we define the following security experiment:*

$Exp^{\mathsf{c-h-sfpk}^*}_{\Pi_{\mathsf{SFPK}},\mathcal{A}}(\lambda)$

$\omega_0, \omega_1 \leftarrow\!\!\$\ \mathsf{coin}$
for $i \in \{0,1\}$
  $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow\!\!\$\ \mathsf{KGen}_{\mathsf{SFPK}}(\lambda, \omega_i)$
$b \leftarrow\!\!\$\ \{0,1\}$
$r \leftarrow\!\!\$\ \mathsf{coin}$
$\mathsf{sk}' \leftarrow \mathsf{ChgSK}_{\mathsf{SFPK}}(\mathsf{sk}_b, r)$
$\mathsf{pk}' \leftarrow \mathsf{ChgPK}_{\mathsf{SFPK}}(\mathsf{pk}_b, r)$
$\hat{b} \leftarrow\!\!\$\ \mathcal{A}^{\mathcal{O}_1(\mathsf{sk}', \cdot)}(\omega_0, \omega_1, \mathsf{pk}')$
return $b = \hat{b}$

with $\mathcal{O}_1(\mathsf{sk}', \cdot)$ *being an oracle as of the following:*

$\mathcal{O}_1(\mathsf{sk}', m)$

return $\mathsf{Sign}_{\mathsf{SFPK}}(\mathsf{sk}', m)$

We define the advantage of $\mathcal{A}$ *in breaking the class-hiding property of* Π_{SFPK} *as*

$$Adv^{\mathsf{c\text{-}h\text{-}sfpk}^*}_{\Pi_{\mathsf{SFPK}},\mathcal{A}}(\lambda) := |\Pr[Exp^{\mathsf{c-h-sfpk}^*}_{\Pi_{\mathsf{SFPK}},\mathcal{A}}(\lambda) = 1] - \frac{1}{2}| \tag{3.19}$$

We call Π_{SFPK} class-hiding *if for all PPT adversaries* $\mathcal{A}$ $Adv^{\mathsf{c\text{-}h\text{-}sfpk}^*}_{\Pi_{\mathsf{SFPK}},\mathcal{A}}(\lambda)$ *is negligible in* λ.

Analogous to the issue with our full-anonymity definition (Def. 3.10), there is the possibility of defining class-hiding via a distinguishing game. This leads to a completely equivalent understanding of class-hiding but at the same time is way more convenient (see Sect. 3.7) for the proof of the full-anonymity of *gFPK-GS* (Thm. 5.3). For completeness, the formal definition of class-hiding as a distinguishing game is given in the following. Proving equivalence of the two definitions can be done analogous to the respective proof for full-anonymity (Thm. 3.12 in this thesis).

Definition 3.17 (**class-hiding SFPK (distinguishing variant)**) *Let* $\lambda \in \mathbb{N}$ *be the security parameter. For an SFPK scheme*
$\Pi_{\mathsf{SFPK}} = (\mathsf{KGen}_{\mathsf{SFPK}}, \mathsf{TKGen}_{\mathsf{SFPK}}, \mathsf{Sign}_{\mathsf{SFPK}}, \mathsf{ChkRep}_{\mathsf{SFPK}}, \mathsf{ChgPK}_{\mathsf{SFPK}}, \mathsf{ChgSK}_{\mathsf{SFPK}}, \mathsf{Vf}_{\mathsf{SFPK}})$
and an adversary $\mathcal{A}$ *we define the following security experiment:*

$Exp^{\mathsf{c-h-sfpk-b}}_{\Pi_{\mathsf{SFPK}},\mathcal{A}}(\lambda)$

$\omega_0, \omega_1 \leftarrow\!\!\$\ \mathsf{coin}$
for $i \in \{0,1\}$
  $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow\!\!\$\ \mathsf{KGen}_{\mathsf{SFPK}}(\lambda, \omega_i)$
$r \leftarrow\!\!\$\ \mathsf{coin}$
$\mathsf{sk}' \leftarrow \mathsf{ChgSK}_{\mathsf{SFPK}}(\mathsf{sk}_b, r)$
$\mathsf{pk}' \leftarrow \mathsf{ChgPK}_{\mathsf{SFPK}}(\mathsf{pk}_b, r)$
$Q := \emptyset$
$\hat{b} \leftarrow\!\!\$\ \mathcal{A}^{\mathcal{O}_1(\mathsf{sk}',\cdot)}(\omega_0, \omega_1, \mathsf{pk}')$
return $\hat{b}$

with $\mathcal{O}_1(\mathsf{sk}', m)$ *being an oracle as of the following:*

$\mathcal{O}_1(\mathsf{sk}', m)$

return $\mathsf{Sign}_{\mathsf{SFPK}}(\mathsf{sk}', m)$

We define the distinguishing advantage of $\mathcal{A}$ *in breaking the class-hiding property of* Π_{SFPK} *as*

$$Adv^{\mathsf{c\text{-}h\text{-}sfpk}}_{\Pi_{\mathsf{SFPK}},\mathcal{A}}(\lambda) := |\Pr[Exp^{\mathsf{c-h-sfpk-0}}_{\Pi_{\mathsf{SFPK}},\mathcal{A}}(\lambda) = 1] - \Pr[Exp^{\mathsf{c-h-sfpk-1}}_{\Pi_{\mathsf{SFPK}},\mathcal{A}}(\lambda) = 1]| \tag{3.20}$$

We call Π_{SFPK} class-hiding (in the distinguishing variant) *if for all PPT adversaries* $\mathcal{A}$ $Adv^{\mathsf{c\text{-}h\text{-}sfpk}}_{\Pi_{\mathsf{SFPK}},\mathcal{A}}(\lambda)$ *is negligible in* λ*.*

Analogous to the bit guessing variant of class-hiding, it is important that $\mathcal{A}$ is given access to the randomness used to create the two key pairs instead of the key pairs only since the randomness might reveal additional information about the key pair generated from it, depending on the SFPK. Next we formally state the equivalence of the two class-hiding definitions for completeness.

Theorem 3.18 *Let* Π_{SFPK} *be an SFPK scheme,* $\lambda \in \mathbb{N}$ *security parameter. For a PPT adversary* $\mathcal{A}$ *we have*

$$Adv^{\mathsf{c\text{-}h\text{-}sfpk}}_{\Pi_{\mathsf{SFPK}},\mathcal{A}}(\lambda) = 2 \cdot Adv^{\mathsf{c\text{-}h\text{-}sfpk}^*}_{\Pi_{\mathsf{SFPK}},\mathcal{A}}(\lambda) \tag{3.21}$$

This yields that Π_{SFPK} *is class-hiding if and only if GS is class-hiding in the distinguishing variant.*

Proof. The proof is analogous to the proof of Thm. 3.12. □

Existential unforgeability under chosen-message attacks (in short EUF-CMA) is a standard security requirement for digital signature schemes, basically meaning that if not in possession of the secret key sk of a key pair $(\mathsf{pk}, \mathsf{sk})$, no one should be able to create a signature that is valid under pk. This classic version of EUF-CMA needs an adaption to allow for meaningful statements about the security of an SFPK scheme, because an adversary who forges a signature which is valid under a different public key $\mathsf{pk}' \in [\mathsf{ChgPK}_{\mathsf{SFPK}}(\mathsf{pk}, r)]$ (a different representative of $[\mathsf{pk}]$) might also be a threat

since pk' can be found related to pk using $\mathsf{ChkRep}_{\mathsf{SFPK}}$. These reasonings result in the following two unforgeability definitions for SFPK. We will start with defining the strong unforgeability variant.

Definition 3.19 (**sEUF-CMA for SFPK**) *Let $\lambda \in \mathbb{N}$ be a security parameter. For an SFPK scheme*
$\Pi_{\mathsf{SFPK}} = (\mathsf{KGen}_{\mathsf{SFPK}}, \mathsf{TKGen}_{\mathsf{SFPK}}, \mathsf{Sign}_{\mathsf{SFPK}}, \mathsf{ChkRep}_{\mathsf{SFPK}}, \mathsf{ChgPK}_{\mathsf{SFPK}}, \mathsf{ChgSK}_{\mathsf{SFPK}}, \mathsf{Vf}_{\mathsf{SFPK}})$
and an adversary $\mathcal{A}$ we define the following security experiment:

$Exp^{\mathsf{seuf-sfpk}}_{\Pi_{\mathsf{SFPK}},\mathcal{A}}(\lambda)$

$\omega \leftarrow\!\!\$\ \mathsf{coin}$
$(\mathsf{pk}, \mathsf{sk}, \tau) \leftarrow\!\!\$\ \mathsf{TKGen}_{\mathsf{SFPK}}(\lambda, \omega)$
$Q := \emptyset$
$(\mathsf{pk}', m^*, \sigma^*) \leftarrow\!\!\$\ \mathcal{A}^{\mathcal{O}_1(\mathsf{sk},\cdot),\mathcal{O}_2(\mathsf{sk},\cdot,\cdot)}(\mathsf{pk}, \tau)$
if $(m^*, \sigma^*) \in Q$
  return 0
return $\mathsf{ChkRep}_{\mathsf{SFPK}}(\tau, \mathsf{pk}') = 1 \wedge \mathsf{Vf}(\mathsf{pk}', m^*, \sigma^*) = 1$

with $\mathcal{O}_1$ and $\mathcal{O}_2$ being oracles as of the following:

$\mathcal{O}_1(\mathsf{sk}, m)$

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

$\mathcal{O}_2(\mathsf{sk}, m, r)$

$\mathsf{sk}' \leftarrow\!\!\$\ \mathsf{ChgSK}_{\mathsf{SFPK}}(\mathsf{sk}, r)$
$\sigma \leftarrow\!\!\$\ \mathsf{Sign}_{\mathsf{SFPK}}(\mathsf{sk}', m)$
$Q := Q \cup \{(m, \sigma)\}$
return σ

We define the advantage of $\mathcal{A}$ in breaking the strong existential unforgeability under chosen message attacks of Π_{SFPK} as

$$Adv^{\mathsf{seuf-sfpk}}_{\Pi_{\mathsf{SFPK}},\mathcal{A}}(\lambda) := \Pr[Exp^{\mathsf{seuf-sfpk}}_{\Pi_{\mathsf{SFPK}},\mathcal{A}}(\lambda) = 1] \tag{3.22}$$

We say that Π_{SFPK} is strongly existentially unforgeable under chosen-message attacks *(or in short* sEUF-CMA*) if for all PPT adversaries $\mathcal{A}$ $Adv^{\mathsf{seuf-sfpk}}_{\Pi_{\mathsf{SFPK}},\mathcal{A}}(\lambda)$ is negligible in λ.*

$\mathcal{O}_1$ is a standard signing oracle, returning a signature of the submitted message under the secret key sk it is instantiated with. $\mathcal{O}_2$ is what one might call a randomize-then-sign oracle which first randomizes its secret key with a given randomness before producing a signature for the submitted message. So an adversary against unforgeability of an SFPK scheme is given the possibility to observe signatures made using any secret key related to the challenge secret key.

Note that according to above definition, an adversary $\mathcal{A}$ who forges a different signature for a message submitted to one of the oracles $\mathcal{O}_1$ and $\mathcal{O}_2$ is considered to be an equally severe threat to the unforgeability of an SFPK scheme Π_{SFPK} as an adversary who manages to sign a new message. However, there might be situations where it suffices to exclude the existence of the second type of adversaries. This motivates the following weaker unforgeability definition for SFPK.

Definition 3.20 (**(weak) EUF-CMA for SFPK**) *Let $\lambda \in \mathbb{N}$ be a security parameter. For an SFPK scheme*
$\Pi_{\mathsf{SFPK}} = (\mathsf{KGen}_{\mathsf{SFPK}}, \mathsf{TKGen}_{\mathsf{SFPK}}, \mathsf{Sign}_{\mathsf{SFPK}}, \mathsf{ChkRep}_{\mathsf{SFPK}}, \mathsf{ChgPK}_{\mathsf{SFPK}}, \mathsf{ChgSK}_{\mathsf{SFPK}}, \mathsf{Vf}_{\mathsf{SFPK}})$
and an adversary $\mathcal{A}$ we define the security experiment $Exp^{\mathsf{euf-sfpk}}_{\Pi_{\mathsf{SFPK}},\mathcal{A}}(\lambda)$ which proceeds exactly like $Exp^{\mathsf{seuf-sfpk}}_{\Pi_{\mathsf{SFPK}},\mathcal{A}}(\lambda)$ from Def. 3.19, only line 5 is replaced by

if $\exists(m,\cdot) \in Q : m = m^*$

We define the advantage of $\mathcal{A}$ in breaking the existential unforgeability under chosen message attacks of Π_{SFPK} as

$$Adv^{\mathsf{euf-sfpk}}_{\Pi_{\mathsf{SFPK}},\mathcal{A}}(\lambda) := \Pr[Exp^{\mathsf{euf-sfpk}}_{\Pi_{\mathsf{SFPK}},\mathcal{A}}(\lambda) = 1] \tag{3.23}$$

We say that Π_{SFPK} is existentially unforgeable under chosen-message attacks *or in short* EUF-CMA *if for all PPT adversaries $\mathcal{A}$ $Adv^{\mathsf{euf-sfpk}}_{\Pi_{\mathsf{SFPK}},\mathcal{A}}(\lambda)$ is negligible in λ.*

Next we will briefly cover how SFPK works in the multi-user setting. Here it is possible to have a setup phase before the actual key generation in which a *common reference string* is generated from the security parameter as $\rho \leftarrow_{\$} \mathsf{CRSGen}(\lambda)$. This is a value which is part of every user's public key and implicitly input to all algorithms of the SFPK scheme. The common reference string usually contains public parameters that should be the same for all users. Therefore the exact definition of it and the common reference string generation algorithm CRSGen is dependent on the concrete SFPK scheme. Whether the common reference string is generated by a trusted third party generally plays a significant role when it comes to defining security of SFPK schemes. The definitions of class-hiding (Def. 3.17) and weak (Def. 3.20) and strong unforgeability (Def. 3.19) used in this thesis (which are based on the definitions given by the inventors of SFPK in [3]) are all made under the implicit assumption that the common reference string is generated by a trusted third party. However, as pointed out by Backes et al. in [3], it is possible to define those three security notions under the assumption that the common reference string is generated by an untrusted party (those security notions get the prefix "malicious", i.e. *malicious class-hiding*). Despite this, common reference strings will be ignored in this thesis for simplicity since they are not used in the reductions in the proofs of full-traceability and full-anonymity of *gFPK-GS* (see Thm. 5.2, Thm. 5.3) in which we prerequisite security properties of SFPK schemes in the non-malicious setup from below definitions. This follows from the fact that the respective proofs for full-traceability and full-anonymity of *gFPK-GS* will be completely analogous in the case that the setup of the SFPK used to instantiate *gFPK-GS* is not trusted, the only difference being that the common reference string is generated by the

adversary attacking an SFPK security property during the reduction (instead of being generated by the challenger of the security game for this security property like in the trusted version).

3.5 Structure-preserving signatures on equivalence classes

In this section we formally define the second major building block of the group signature *gFPK-GS* (Sect. 4.1), which is structure-preserving signatures on equivalence classes (in short SPS-EQ). An SPS-EQ scheme is defined over a fixed equivalence relation R which partitions its message space into equivalence classes. Given a valid message-signature pair (m, σ) created using such a scheme one can easily obtain a valid signature σ' on another representative m' of the equivalence class $[m]_R$ of m with respect to relation R. So we see that the general idea of SPS-EQ is similar to the one behind SFPK with the difference that instead of the key space the message space is divided into equivalence classes. The following definitions for its syntax and unforgeability were originally presented by Fuchsbauer et al. in [11]. The definition of perfect adaptation of signatures (Def. 3.24) used here was given by Backes et al. and can be found as Definition 12 in [3]. We will start by giving the definition of the syntax of SPS-EQ schemes.

Definition 3.21 (**SPS-EQ scheme**) *Let $l \in \mathbb{N}, l > 1$, R be an equivalence relation. A* structure preserving signature scheme on equivalence classes with vector length l and equivalence relation R over the message space *(in short* SPS-EQ *with vector length l and relation R) is a tuple $\Pi_{\mathsf{SPS}} = (\mathsf{BGGen}_{\mathsf{SPS}}, \mathsf{KGen}_{\mathsf{SPS}}, \mathsf{Sign}_{\mathsf{SPS}}, \mathsf{ChgRep}_{\mathsf{SPS}}, \mathsf{Vfy}_{\mathsf{SPS}}, \mathsf{VKey}_{\mathsf{SPS}})$ of PPT algorithms as follows:*

$\mathsf{BGGen}_{\mathsf{SPS}}(\lambda)$ *bilinear group generator (see Def. 3.6). The message space of Π_{SPS} is either defined as G_1^l or G_2^l for the prime order groups G_1, G_2 in the output $BG := (p, G_1, G_2, G_T, e, g_1, g_2)$ of $\mathsf{BGGen}_{\mathsf{SPS}}$.*

$\mathsf{KGen}_{\mathsf{SPS}}(BG, l)$ *probablistic key generation algorithm, on input a bilinear group BG and the vector length l it outputs a key pair $(\mathsf{pk}, \mathsf{sk})$*

$\mathsf{Sign}_{\mathsf{SPS}}(\mathsf{sk}, m)$ *probablistic signing algorithm, takes as input a secret key sk and a message m and outputs a signature σ for m*

$\mathsf{ChgRep}_{\mathsf{SPS}}(m, \sigma, r, \mathsf{pk})$ *probablistic algorithm which on input a message $m \in [m]_R \subseteq G_i^l$, a signature σ for m, randomness $r \leftarrow_{\$} \mathsf{coin}$ and a public key pk outputs an updated message-signature pair (m', σ') such that $m' \in [m]_R$ and (m', σ') is valid under pk. (m', σ') is called a* randomized version of *(m, σ)*

$\mathsf{Vfy}_{\mathsf{SPS}}(\mathsf{pk}, m, \sigma)$ *deterministic algorithm, takes as input a message $m \in [m]_R \subseteq G_i^l$, a signature σ and a public key pk and outputs 0 or 1*

$\mathsf{VKey}_{\mathsf{SPS}}(\mathsf{pk}, \mathsf{sk})$ *deterministic algorithm, takes as input a public key pk and a secret key sk and returns 1 if $(\mathsf{pk}, \mathsf{sk})$ is a valid key pair, 0 otherwise*

For an SPS-EQ scheme to work correctly, all key pairs output by $\mathsf{KGen}_{\mathsf{SPS}}$ and all signatures generated by $\mathsf{Sign}_{\mathsf{SPS}}$ must be valid. Furthermore, if we randomize a message-signature pair (obtained using $\mathsf{Sign}_{\mathsf{SPS}}$) using $\mathsf{ChgRep}_{\mathsf{SPS}}$ we must be sure to get another valid message-signature pair. These requirements result in the following definition.

Definition 3.22 (**correct SPS-EQ scheme**) *Let* $l \in \mathbb{N}, l > 1, \lambda \in \mathbb{N}$, $BG = (p, G_1, G_2, G_T, e, g_1, g_2) \leftarrow_{\$} \mathsf{BGGen}_{\mathsf{SPS}}(\lambda)$. *Let* R *be an equivalence relation. An SPS-EQ scheme* $\Pi_{\mathsf{SPS}} = (\mathsf{BGGen}_{\mathsf{SPS}}, \mathsf{KGen}_{\mathsf{SPS}}, \mathsf{Sign}_{\mathsf{SPS}}, \mathsf{ChgRep}_{\mathsf{SPS}}, \mathsf{Vfy}_{\mathsf{SPS}}, \mathsf{VKey}_{\mathsf{SPS}})$ *with relation* R *and vector length* l *is* correct *if the following conditions are fulfilled:*

- *for all* $(\mathsf{pk}, \mathsf{sk}) \leftarrow_{\$} \mathsf{KGen}_{\mathsf{SPS}}(BG, l)$ *we have*

$$\mathsf{VKey}_{\mathsf{SPS}}(\mathsf{pk}, \mathsf{sk}) = 1 \tag{3.24}$$

- *for all* $(\mathsf{pk}, \mathsf{sk}) \leftarrow_{\$} \mathsf{KGen}_{\mathsf{SPS}}(BG, l)$, *messages* $m \in G_i^l$

$$\Pr[\mathsf{Vfy}_{\mathsf{SPS}}(m, \mathsf{Sign}_{\mathsf{SPS}}(\mathsf{sk}, m), \mathsf{pk}) = 1] = 1 \tag{3.25}$$

  holds

- *for all* $(\mathsf{pk}, \mathsf{sk}) \leftarrow_{\$} \mathsf{KGen}_{\mathsf{SPS}}(BG, l)$, *messages* $m \in G_i^l, r \in \mathsf{coin}$ *we have*

$$\Pr[\mathsf{Vfy}_{\mathsf{SPS}}(\mathsf{ChgRep}_{\mathsf{SPS}}(m, \mathsf{Sign}_{\mathsf{SPS}}(\mathsf{sk}, m), r, \mathsf{pk}), \mathsf{pk}) = 1] = 1 \tag{3.26}$$

As far as we know, all existing instantiations of SPS-EQ are based on bilinear groups of type 3. The equivalence relation which Π_{SPS} is defined over is left arbitrary in above definition. A discussion about restrictions to the used equivalence relation when it comes to practical use of SPS-EQ schemes can be found in Sect. 3.6. If the relation R is obvious or not important we omit it in the name and simply call this primitive SPS-EQ.

Next we will define the relevant security properties for SPS-EQ, starting with existential unforgeability. Similar to SFPK, an adaptation of the classic unforgeability notion of digital signatures is needed for SPS-EQ schemes. This follows from the fact that using $\mathsf{ChgRep}_{\mathsf{SPS}}$, everyone can obtain new valid message signature pairs from a given one. We see that for a correct SPS-EQ scheme and any message-signature pair (m', σ') obtained from a given pair (m, σ) using $\mathsf{ChgRep}_{\mathsf{SPS}}$ we have $m' \in [m]$. So the only task that can still be computationally infeasible is signing a message from a class that has not been signed before. These considerations are formalized in the following definition.

Definition 3.23 (**EUF-CMA for SPS-EQ**) *Let* $l \in \mathbb{N}, l > 1, \lambda \in \mathbb{N}$. *For an SPS-EQ scheme* $\Pi_{\mathsf{SPS}} = (\mathsf{BGGen}_{\mathsf{SPS}}, \mathsf{KGen}_{\mathsf{SPS}}, \mathsf{Sign}_{\mathsf{SPS}}, \mathsf{ChgRep}_{\mathsf{SPS}}, \mathsf{Vfy}_{\mathsf{SPS}}, \mathsf{VKey}_{\mathsf{SPS}})$ *with relation* R *and vector length* l *and an adversary* $\mathcal{A}$ *we define the following security experiment:*

$Exp^{\mathsf{euf-sps}}_{\Pi_{\mathsf{SPS}},\mathcal{A}}(\lambda)$

1 : $BG := \mathsf{BGGen}_{\mathsf{SPS}}(\lambda)$
2 : $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{KGen}_{\mathsf{SPS}}(BG, l)$
3 : $Q := \emptyset$
4 : $(m^*, \sigma^*) \leftarrow\!\!\$\ \mathcal{A}^{\mathcal{O}(\mathsf{sk},\cdot)}(\mathsf{pk})$
5 : **return** $(\forall m \in Q : [m^*]_R \neq [m]_R) \wedge \mathsf{Vf}(m^*, \sigma^*, \mathsf{pk}) = 1$

with $\mathcal{O}$ being an oracle as of the following:

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

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

We define the advantage of $\mathcal{A}$ in breaking the EUF-CMA of Π_{SPS} as

$$Adv^{\mathsf{euf-sps}}_{\Pi_{\mathsf{SPS}},\mathcal{A}}(\lambda) = \Pr[Exp^{\mathsf{euf-sps}}_{\Pi_{\mathsf{SPS}},\mathcal{A}}(\lambda) = 1] \tag{3.27}$$

We say that Π_{SPS} is existentially unforgeable under chosen-message attacks *(in short* EUF-CMA*) if for all PPT adversaries $\mathcal{A}$ $Exp^{\mathsf{euf-sps}}_{\Pi_{\mathsf{SPS}},\mathcal{A}}(\lambda)$ is negligible in λ.*

In the above security experiment the adversary is given access to a straightforward signing oracle $\mathcal{O}$ which produces signatures on messages it is queried for using the challenge secret key sk. As already stated before Def. 3.23, two ways of obtaining a new SPS-EQ message-signature pair exist: signing a message using $\mathsf{Sign}_{\mathsf{SPS}}$ and randomizing an existing pair using $\mathsf{ChgRep}_{\mathsf{SPS}}$. Whether the results of these computations can be distinguished efficiently cannot be answered from the SPS-EQ definition alone which justifies the following definition:

Definition 3.24 (perfect adaptation of signatures) *Let $l \in \mathbb{N}, l > 1, \lambda \in \mathbb{N}, BG = (p, G_1, G_2, G_T, e, g_1, g_2) \leftarrow \mathsf{BGGen}_{\mathsf{SPS}}(\lambda)$. An SPS-EQ scheme $\Pi_{\mathsf{SPS}} = (\mathsf{BGGen}_{\mathsf{SPS}}, \mathsf{KGen}_{\mathsf{SPS}}, \mathsf{Sign}_{\mathsf{SPS}}, \mathsf{ChgRep}_{\mathsf{SPS}}, \mathsf{Vfy}_{\mathsf{SPS}}, \mathsf{VKey}_{\mathsf{SPS}})$ with relation R and vector length l* perfectly adapts signatures *if for all $(\mathsf{pk}, \mathsf{sk}, m, \sigma, r)$ where $\mathsf{VKey}_{\mathsf{SPS}}(\mathsf{pk}, \mathsf{sk}) = 1, m \in G_i^l, r \in \mathsf{coin}$ and $\mathsf{Vfy}_{\mathsf{SPS}}(m, \sigma, \mathsf{pk}) = 1$ the distributions of*

$$(m^r, \mathsf{Sign}(\mathsf{sk}, m^r)) \text{ and } \mathsf{ChgRep}_{\mathsf{SPS}}(m, \sigma, r, \mathsf{pk})$$

are identical. m^r in above expression denotes another representative of the equivalence class of m which was obtained using randomness r.

3.6 An important equivalence relation

According to Backes et al. [3], existing implementations of SPS-EQ schemes are over the following relation R_{exp}. It does not become clear in [3] whether there exist SFPK schemes over a different relation than R_{exp}, since both SFPK schemes the authors present

in Chapter 5 of their paper are clearly over R_{exp} (this can be seen easily in the respective $\mathsf{ChgPK}_{\mathsf{SFPK}}$ algorithms) but it is not explicitly stated in [3] that this has to be the case for all SFPK schemes. So whether there exist relations $R \neq R_{\mathsf{exp}}$ to instantiate SFPK or SPS-EQ schemes with remains an open question. This is the reason why the definitions of SFPK (Def. 3.14), SPS-EQ (Def. 3.21) and *gFPK-GS* (Def. 4.1) are given for arbitrary equivalence relations in this thesis. In the following we will give the formal definition of R_{exp} and convince ourselves that it in fact is an equivalence relation.

Definition 3.25 *Let G be a group of prime order p, let $l \in \mathbb{N}$. We define the relation R_{exp} on G^l as*

$$R_{\mathsf{exp}} := \{(M, N) \in (G^l)^2 \mid \exists r \in \mathbb{Z}_p^* : M^r = N\}$$

The group operation in above definition is component-wise, i.e.

$$M^r := (M_1^r, \ldots, M_l^r)$$

for $M := (M_1, \ldots, M_l)$. If R_{exp} is used to instantiate an SFPK or SPS-EQ scheme, the set of the random values influencing the behavior of the probablistic algorithms $\mathsf{ChgPK}_{\mathsf{SFPK}}$, $\mathsf{ChgSK}_{\mathsf{SFPK}}$ (for SFPK) and *chgrep* (for SPS-EQ) is implicitly set to $\mathbb{Z}_p^*$. The proof of R_{exp} being an equivalence relation is straight application of definitions and therefore only sketched in this thesis.

Lemma 3.26 *R_{exp} from Def. 3.25 is an equivalence relation.*

Proof. Reflexivity follows from the fact that an identity element $1_{\mathbb{Z}_p^*} \in \mathbb{Z}_p^*$ exists. Existence of inverse elements for every $r \in \mathbb{Z}_p^*$ yields symmetry of R_{exp}, using the fact that

$$g^p = g^{|G|} = 1_G$$

for any $g \in G$. Since $\mathbb{Z}_p^*$ is complete with respect to multiplication, transitivity of R_{exp} is also fulfilled. □

3.7 Changes to original definitions

In this section, all differences regarding the definitions and findings in this thesis with respect to the original work of the creators of SFPK [3] and SPS-EQ [11] and group signature security [5] will be listed and discussed. To start off, we list minor additions to the above mentioned definitions:

- When defining (perfect) correctness of a group signature (done in Def. 3.8 in this thesis) like in [5], we must require that the probabilities that a signature created with GSign is valid is 1 (instead of requiring validity of $\mathsf{GSign}(\mathsf{gsk}[\mathsf{i}], m)$ like done in [5]). This is because GSign is a probablistic algorithm and therefore $\mathsf{GSign}(\mathsf{gsk}[\mathsf{i}], m)$ is not a well-defined value, since the exact randomness used in this call of the algorithm is not fixed. The same holds for the second requirement for perfect correctness of group signature schemes (see Eq. (3.7)), so the probability of Open

retrieving the identity of the signer who created a given true signature must as well be 1.

- For reasons of convenience in the proof of *gFPK-GS*'s full-anonymity (Thm. 5.3), full-anonymity is defined via a bit guessing game in Def. 3.11 in this thesis (the original definition by Bellare et al. in [5] requires the adversary to distinguish experiments). This change is helpful since the computational indistinguishability of the games in the game sequence is easier to prove with them being bit guessing games. Also for convenience in the proof of Thm. 5.3 we defined class-hiding via distinguishing experiments (see Def. 3.17), in contrast to [3] where it was defined as a bit guessing game. This helps proving that the full-anonymity game from Def. 3.10 can be made independent of the hidden bit using a game hop.

3.7.1 Full-traceability definition by Bellare et al.

Next we will give the original definition of full-traceability by Bellare et al. [5] for completeness. The definition used for the proof of the full-traceability of *gFPK-GS* in this thesis (see Thm. 5.2) can be found under Def. 3.13. Def. 3.13 is preferred over below definition of full-traceability since the way the construction of the collusion set C is described is more compact.

Definition 3.27 *Let $\lambda \in \mathbb{N}$ be the security parameter, $n \in \mathbb{N}$ group size. We define the following security experiment for a group signature scheme $GS = (\mathsf{GKGen}, \mathsf{GSign}, \mathsf{GVf}, \mathsf{Open})$ and an adversary $\mathcal{A}$:*

$Exp^{\mathsf{trace}}_{GS,\mathcal{A}}(\lambda, n)$

1 : $(\mathsf{gpk}, \mathsf{gmsk}, \mathsf{gsk}) \leftarrow\!\!\$\ \mathsf{GKGen}(\lambda, n)$
2 : $\mathsf{St} := (\mathsf{gmsk}, \mathsf{gpk})$
3 : $\mathsf{C} = \emptyset, K := \varepsilon, \mathsf{Cont} = \mathbf{true}$
4 : **while** $\mathsf{Cont} = \mathbf{true}$ **do**
5 : $\quad(\mathsf{Cont}, \mathsf{St}, j) \leftarrow\!\!\$\ \mathcal{A}^{\mathsf{GSign}(\mathsf{gsk}[\cdot],\cdot)}(\mathsf{choose}, \mathsf{St}, K)$
6 : $\quad$**if** Cont
7 : $\quad\quad\mathcal{C} := \mathcal{C} \cup \{j\}$
8 : $\quad\quad K := \mathsf{gsk}[\mathsf{j}]$
9 : $(m, \sigma) \leftarrow\!\!\$\ \mathcal{A}^{\mathsf{GSign}(\mathsf{gsk}[\cdot],\cdot)}(\mathsf{guess}, \mathsf{St})$
10 : **if** $\mathsf{GVf}(\mathsf{gpk}, m, \sigma) = 0$
11 : $\quad$**return** 0
12 : **if** $\mathsf{Open}(\mathsf{gmsk}, m, \sigma) = \perp$
13 : $\quad$**return** 1
14 : **if** $\exists i \in [n] : \mathsf{Open}(\mathsf{gmsk}, m, \sigma) = i \wedge i \notin \mathcal{C} \wedge i, m$ *not queried by* $\mathcal{A}$ **to** *it's oracle*
15 : $\quad$**return** 1
16 : **else**
17 : $\quad$**return** 0

We define the advantage of $\mathcal{A}$ in breaking full-traceability of GS as

$$Adv^{\mathsf{trace}}_{GS,\mathcal{A}}(\lambda, n) := \Pr[Exp^{\mathsf{trace}}_{GS,\mathcal{A}}(\lambda, n) = 1] \tag{3.28}$$

We say that GS is fully-traceable *if for all PPT-adversaries $\mathcal{A}$ $Adv^{\mathsf{trace}}_{GS,\mathcal{A}}(\lambda, n)$ is negligible in λ.*

We see that the while-loop in lines 4 to 8 in above security game implements giving $\mathcal{A}$ access to an oracle answering identities with the respective personal secret keys like the oracle $\mathcal{O}_{\mathsf{C}}(\cdot)$ in Def. 3.13. While the adversary indicates that it will submit more collusion queries (Cont = true), the answer $\mathsf{gsk}[\mathsf{j}]$ to the queried identity j output by $\mathcal{A}$ is saved in the variable K which is input to $\mathcal{A}$ before waiting for $\mathcal{A}$'s next query. When modeling the collusion oracle access via a while-loop (as in above Def. 3.27), the tuple of random variables $\mathcal{A}$ sees throughout the experiment becomes unnecessarily complex since the variables K and St are potentially rewritten several times throughout the experiment. This is why we defined an additional collusion oracle $\mathcal{O}_{\mathsf{C}}(\cdot)$ in the full-traceability definition in this thesis (see Def. 3.13). We see that this obviously gives the adversary the freedom to interleave signing and collusion queries in any way it wants to, however, since both oracles $\mathcal{O}_{\mathsf{C}}(\cdot)$ and GSign from our full-traceability definition (Def. 3.13) are non-interactive algorithms, any interleaving of collusion and signing oracle queries possible in Def. 3.13 can be simulated by the while-loop in above definition, meaning these two full-traceability definitions are equivalent.

3.7.2 Full-anonymity definition by Backes et al.

In this section, we are going to take a look at the definition of full anonymity that was used by Backes et al. in [3]. More precisely, we will point out that the combination of how the security game and advantage are defined in this definition basically renders the definition useless for meaningful statements about a schemes' security. To start off, the formal definition of full-anonymity used by Backes et al. will be given. It can be found as Definition 19 in [3].

Definition 3.28 (**(broken) full-anonymity**) *Let $\lambda \in \mathbb{N}$ be a security parameter, $n \in \mathbb{N}$ group size. We define the following security experiment for a group signature scheme $GS = (\mathsf{GKGen}, \mathsf{GSign}, \mathsf{GVf}, \mathsf{Open})$ and an adversary $\mathcal{A}$:*

$Exp^{\mathsf{anon}-b}_{GS,\mathcal{A}}(\lambda, n)$

1 : $(\mathsf{gpk}, \mathsf{gmsk}, \mathsf{gsk}) \leftarrow\!\!\$\ \mathsf{GKGen}(\lambda, n)$
2 : $Q := \emptyset$
3 : $(St, i_0, i_1, m) \leftarrow\!\!\$\ \mathcal{A}^{\mathsf{Open}(\mathsf{gmsk},\cdot,\cdot)}(\mathsf{gpk}, \mathsf{gsk})$
4 : $\sigma \leftarrow\!\!\$\ \mathsf{GSign}(\mathsf{gsk}[\mathsf{i_b}], m)$
5 : $\hat{b} \leftarrow\!\!\$\ \mathcal{A}^{\mathcal{O}(\mathsf{gmsk},\cdot,\cdot)}(St, \sigma)$
6 : **if** $(m, \sigma) \in Q$ **then return** 0
7 : **else return** $\hat{b} = b$

with $\mathcal{O}$ being an oracle as of the following:

$\mathcal{O}(\mathsf{gmsk}, m, \sigma)$

1 : $Q := Q \cup \{(m, \sigma)\}$
2 : **return** $\mathsf{Open}(\mathsf{gmsk}, m, \sigma)$

We define the advantage of $\mathcal{A}$ in breaking full-anonymity of GS as

$$Adv^{\mathsf{anon}}_{GS,\mathcal{A}}(\lambda, n) = |\Pr[Exp^{\mathsf{anon}-0}_{GS,\mathcal{A}}(\lambda, n) = 1] - \Pr[Exp^{\mathsf{anon}-1}_{GS,\mathcal{A}}(\lambda, n) = 1]| \tag{3.29}$$

We say that GS is fully-anonymous *if for all PPT-adversaries $\mathcal{A}$ $Adv^{\mathsf{anon}}_{GS,\mathcal{A}}(\lambda, n)$ is negligible in λ.*

The boolean value the experiment $Exp^{\mathsf{anon}-b}_{GS,\mathcal{A}}(\lambda, n)$ returns shall be interpreted as a bit for convenience, with 1 corresponding to true, 0 corresponding to false. As well as in the adapted version of the full-anonymity definition by Bellare et al. [5] which can be found as Def. 3.11 in this thesis, the adversary is given access to the personal secret keys of all group members and an open oracle. This oracle will answer message-signature pairs with the identity of the signer which created the signature. The problem that basically renders above definition entirely useless emerges from the combination of how the adversaries advantage and the return value of the above experiment $Exp^{\mathsf{anon}-b}_{GS,\mathcal{A}}(\lambda, n)$ are defined. We see that the experiment returns 1 if and only if $\mathcal{A}$ guesses the hidden bit b correctly, therefore wins the experiment. The adversary's advantage is furthermore defined as the absolute difference between it's winning probabilities in the cases $b = 0, b = 1$. So this means that an adversary $\mathcal{A}$ that always wins the game has the smallest possible advantage, which corresponds to considering this adversary the least possible threat to the full-anonymity of GS. It is plain to see that such an adversary in fact is the most severe threat to the full-anonymity of GS one can come up with in a theoretical setting, so this conveys a major flaw in the full-anonymity definition from [3].

4 Group signature by Backes et al.

In this chapter we will introduce the static group signature scheme *gFPK-GS* presented by Backes et al. in [3] which we refer to as *gFPK-GS*. It can be found as Scheme 2 in the original paper [3]. After the formal definition of *gFPK-GS* we will discuss suitable prerequisites for *gFPK-GS* to be correct and then prove it's correctness. We will also examine to what extent signatures produced with *gFPK-GS* are randomizable. At the end of the chapter, a list of all changes regarding the original definition of *gFPK-GS* [3] which we build on will be given.

4.1 Definition

First we will provide the formal definition of *gFPK-GS*. The signatures created with *gFPK-GS* all contain an SFPK signature valid under a fresh randomized version of an SFPK public key belonging to the originator of the signature. This randomized SFPK public key is also included in the *gFPK-GS*-signature. The third part is an SPS-EQ certificate for that SFPK public key. Before continuing with the formal definition, a more detailed explanation of the in-and outputs of the four algorithms making up *gFPK-GS* will be given. In *gFPK-GS*, the key generation comprises the generation of an SFPK key pair for each user and SPS-EQ certificates for the public key in each pair. The public key needed to check these certificates for validity will be part of the group public key. Above SFPK key pairs are generated using $\mathsf{TKGen}_{\mathsf{SFPK}}$, the trapdoors generated in the process are needed to open group signatures and therefore part of the group manager secret key. For signing a message a user first randomizes its SFPK key pair and the certificate, then he concatenates the new public key and the new certificate with the message to sign and signs this string using the new SFPK secret key. The resulting signature consists of that SFPK signature as well as the new public key and the SPS-EQ certificate for it. Therefore a group signature's validity is checked by validating the contained certificate and SFPK signature. Hereby the SFPK signature is checked for validity under the public key contained in the group signature. Opening of true signatures (Def. 3.7) is done by simply checking whether there is a group member's SFPK public key whose equivalence class contains the public key which is part of the signature to open. Therefore it is very important that $\mathsf{ChgPK}_{\mathsf{SFPK}}$ is required to really output a different representative of a class, otherwise there might be signatures which can be opened trivially by just comparing the public key in the signature to all in the $\mathsf{gsk}[\mathsf{i}]$. For invalid group signatures, Open will always fail and return the error symbol $\perp$.

If the particular SFPK and SPS-EQ Π_{SFPK} and Π_{SPS} used to instantiate $gFPK\text{-}GS_{\Pi_{\mathsf{SFPK}},\Pi_{\mathsf{SPS}}}$

are arbitrary and not important, we will omit them in the identifier and simply write *gFPK-GS*, as done in the first chapters. Note that for *gFPK-GS* to work correctly, the equivalence relations of the used SFPK and SPS-EQ must fulfill a certain constraint which we will call compatibility of equivalence relations. A formal definition of this property can be found in Def. 4.2 below the following definition of *gFPK-GS*.

Definition 4.1 ($\boldsymbol{gFPK\text{-}GS}_{\Pi_{\mathsf{SFPK}},\Pi_{\mathsf{SPS}}}$) *Let $l \in \mathbb{N}, l > 1$, R, R' be compatible equivalence relations, $\lambda \in \mathbb{N}$ security parameter, $n \in \mathbb{N}$,*
$\Pi_{\mathsf{SPS}} = (\mathsf{BGGen}_{\mathsf{SPS}}, \mathsf{KGen}_{\mathsf{SPS}}, \mathsf{Sign}_{\mathsf{SPS}}, \mathsf{ChgRep}_{\mathsf{SPS}}, \mathsf{Vfy}_{\mathsf{SPS}}, \mathsf{VKey}_{\mathsf{SPS}})$ *be an SPS-EQ scheme with relation R and vector length l,*
$\Pi_{\mathsf{SFPK}} = (\mathsf{KGen}_{\mathsf{SFPK}}, \mathsf{TKGen}_{\mathsf{SFPK}}, \mathsf{Sign}_{\mathsf{SFPK}}, \mathsf{ChkRep}_{\mathsf{SFPK}}, \mathsf{ChgPK}_{\mathsf{SFPK}}, \mathsf{ChgSK}_{\mathsf{SFPK}}, \mathsf{Vf}_{\mathsf{SFPK}})$ *be an SFPK scheme with relation R' over the key space. We define the group signature scheme* $\text{gFPK-GS}_{\Pi_{\mathsf{SFPK}},\Pi_{\mathsf{SPS}}} = (\mathsf{GKGen}, \mathsf{Sign}, \mathsf{GVf}, \mathsf{Open})$ *as of the following:*

$\mathsf{GKGen}(\lambda, n)$

1: $BG := \mathsf{BGGen}_{\mathsf{SPS}}(\lambda)$
2: $(\mathsf{pk}_{\mathsf{SPS}}, \mathsf{sk}_{\mathsf{SPS}}) \leftarrow\!\!\$\ \mathsf{KGen}_{\mathsf{SPS}}(BG, l)$
3: **for** $i \in [n]$
4: $\quad \omega_i \leftarrow\!\!\$\ \mathsf{coin}$
5: $\quad (\mathsf{pk}_i, \mathsf{sk}_i, \tau_i) \leftarrow\!\!\$\ \mathsf{TKGen}_{\mathsf{SFPK}}(\lambda, \omega_i)$
6: $\quad \sigma_{\mathsf{SPS}}^{(i)} \leftarrow\!\!\$\ \mathsf{Sign}_{\mathsf{SPS}}(\mathsf{sk}_{\mathsf{SPS}}, \mathsf{pk}_i)$
7: $\mathsf{gpk} := (BG, \mathsf{pk}_{\mathsf{SPS}})$
8: $\mathsf{gmsk} := ((\mathsf{pk}_i, \tau_i))_{i=1}^n$
9: **for** $i \in [n]$
10: $\quad \mathsf{gsk[i]} := (\mathsf{pk}_i, \mathsf{sk}_i, \sigma_{\mathsf{SPS}}^{(i)})$
11: $\mathsf{gsk} := (\mathsf{gsk[i]})_{i=1}^n$
12: **return** $(\mathsf{gpk}, \mathsf{gmsk}, \mathsf{gsk})$

$\mathsf{GSign}(\mathsf{gsk[i]}, m)$

1: *parse* $\mathsf{gsk[i]} := (\mathsf{pk}_i, \mathsf{sk}_i, \sigma_{\mathsf{SPS}}^{(i)})$
2: $r \leftarrow\!\!\$\ \mathsf{coin}$
3: $\mathsf{pk}_i' \leftarrow\!\!\$\ \mathsf{ChgPK}_{\mathsf{SFPK}}(\mathsf{pk}_i, r)$
4: $\mathsf{sk}_i' \leftarrow\!\!\$\ \mathsf{ChgSK}_{\mathsf{SFPK}}(\mathsf{sk}_i, r)$
5: $(\mathsf{pk}_i', {\sigma_{\mathsf{SPS}}^{(i)}}') \leftarrow\!\!\$\ \mathsf{ChgRep}_{\mathsf{SPS}}(\mathsf{pk}_i, \sigma_{\mathsf{SPS}}^{(i)}, r, \mathsf{pk}_{\mathsf{SPS}})$
6: $M := m || {\sigma_{\mathsf{SPS}}^{(i)}}' || \mathsf{pk}_i'$
7: $\sigma_0 \leftarrow\!\!\$\ \mathsf{Sign}_{\mathsf{SFPK}}(\mathsf{sk}_i', M)$
8: **return** $\sigma_{\mathsf{GS}} = (\mathsf{pk}_i', \sigma_0, {\sigma_{\mathsf{SPS}}^{(i)}}')$

$\mathsf{GVf}(\mathsf{gpk}, m, \sigma_{\mathsf{GS}})$

1: *parse* $\sigma_{\mathsf{GS}} = (\mathsf{pk}', \sigma_0, \sigma'_{\mathsf{SPS}})$
2: *parse* $\mathsf{gpk} = (BG, \mathsf{pk}_{\mathsf{SPS}})$
3: **if** $\mathsf{Vfy}_{\mathsf{SPS}}(\mathsf{pk}', \sigma'_{\mathsf{SPS}}, \mathsf{pk}_{\mathsf{SPS}}) = 0$
4: **return** 0
5: $M := m||\sigma'_{\mathsf{SPS}}||\mathsf{pk}'$
6: **return** $\mathsf{Vf}_{\mathsf{SFPK}}(\mathsf{pk}', M, \sigma_0)$

$\mathsf{Open}(\mathsf{gmsk}, m, \sigma_{\mathsf{GS}})$

1: *parse* $\sigma_{\mathsf{GS}} = (\mathsf{pk}', \sigma_0, \sigma'_{\mathsf{SPS}})$
2: *parse* $\mathsf{gmsk} = ((\mathsf{pk}_i, \tau_i))_{i=1}^n$
3: **if** $\mathsf{GVf}(\mathsf{gpk}, m, \sigma_{\mathsf{GS}}) = 0$
4: **return** $\perp$
5: **if** $\nexists i \in [n] : \mathsf{ChkRep}_{\mathsf{SFPK}}(\tau_i, \mathsf{pk}') = 1$
6: **return** $\perp$
7: **else**
8: **return** i

For *gFPK-GS* to work, the key space of the SFPK must fit the message space of the SPS-EQ. It clearly is sufficient if the two spaces are equal but it is an open question whether the existence of a bijective mapping fulfilling certain constraints (e.g. an isomorphism) between them is also sufficient. Furthermore, compatibility of the relations which the SFPK and SPS-EQ are defined over is needed in order to ensure that when using $\mathsf{ChgRep}_{\mathsf{SPS}}$ in GSign to adapt the SFPK public key and the SPS-EQ certificate for it, the obtained public key is the same as the one obtained earlier from the explicit public key randomization using $\mathsf{ChgPK}_{\mathsf{SFPK}}$. We will formalize this requirement in the following definition.

Definition 4.2 (**compatible equivalence relations**) *Let R, R' be equivalence relations, $\lambda \in \mathbb{N}$ security parameter,*
$\Pi_{\mathsf{SPS}} = (\mathsf{BGGen}_{\mathsf{SPS}}, \mathsf{KGen}_{\mathsf{SPS}}, \mathsf{Sign}_{\mathsf{SPS}}, \mathsf{ChgRep}_{\mathsf{SPS}}, \mathsf{Vfy}_{\mathsf{SPS}}, \mathsf{VKey}_{\mathsf{SPS}})$ *be an SPS-EQ scheme with relation R and vector length $l \in \mathbb{N}, l > 1$,*
$\Pi_{\mathsf{SFPK}} = (\mathsf{KGen}_{\mathsf{SFPK}}, \mathsf{TKGen}_{\mathsf{SFPK}}, \mathsf{Sign}_{\mathsf{SFPK}}, \mathsf{ChkRep}_{\mathsf{SFPK}}, \mathsf{ChgPK}_{\mathsf{SFPK}}, \mathsf{ChgSK}_{\mathsf{SFPK}}, \mathsf{Vf}_{\mathsf{SFPK}})$ *be an SFPK scheme with relation R' over the key space. We call R, R'* compatible *if for all*

- $\lambda \in \mathbb{N}$
- $BG \leftarrow_{\$} \mathsf{BGGen}_{\mathsf{SPS}}(\lambda)$
- $(\mathsf{pk}, \mathsf{sk}) \leftarrow_{\$} \mathsf{KGen}_{\mathsf{SFPK}}(\lambda, \omega)$
- $(\mathsf{pk}_{\mathsf{SPS}}, \mathsf{sk}_{\mathsf{SPS}}) \leftarrow_{\$} \mathsf{KGen}_{\mathsf{SPS}}(BG, l)$
- $\sigma_{\mathsf{SPS}} \leftarrow_{\$} \mathsf{Sign}_{\mathsf{SPS}}(\mathsf{pk}, \mathsf{sk}_{\mathsf{SPS}})$

- $r \in \mathsf{coin}$

we have

$$\forall \mathsf{pk}_1 \leftarrow\!\!\$\ \mathsf{ChgPK}_{\mathsf{SFPK}}(\mathsf{pk}, r), (\mathsf{pk}_2, \sigma'_{\mathsf{SPS}}) \leftarrow\!\!\$\ \mathsf{ChgRep}_{\mathsf{SPS}}(\mathsf{pk}, \sigma_{\mathsf{SPS}}, r, \mathsf{pk}_{\mathsf{SPS}}) : \mathsf{pk}_1 = \mathsf{pk}_2 \tag{4.1}$$

Note that for any equivalence relation R it is possible to define $\mathsf{ChgPK}_{\mathsf{SFPK}}$ and $\mathsf{ChgRep}_{\mathsf{SPS}}$ in a way that R is compatible with itself according to Def. 4.2.

Next we will address an important issue concerning the relation of the SFPK public keys generated during GKGen. For *gFPK-GS* to work correctly, all SFPK public keys that are part of the personal secret keys of the group members need to be in different equivalence classes, therefore pairwise unrelated. More precisely, if this was not the case, *gFPK-GS* cannot be perfectly correct. This is because if there exist $i, j \in [n]$ such that $i \neq j$ and pk_i is related to pk_j (w.L.o.G. $i < j$), Open would trace back all signatures created using gsk[j] back to user i since Open stops once it found a public key pk related to the one in the signature and returns the identity of the user whose personal secret key contains this public key pk. This behaviour violates the second requirement of group signature correctness (Eq. (3.7) in Def. 3.8). A formalization of this remark is the following lemma:

Lemma 4.3 *Let $l \in \mathbb{N}, l > 1$, R, R' be compatible equivalence relations, $\lambda \in \mathbb{N}$ security parameter, $n \in \mathbb{N}, \Pi_{\mathsf{SPS}} = (\mathsf{BGGen}_{\mathsf{SPS}}, \mathsf{KGen}_{\mathsf{SPS}}, \mathsf{Sign}_{\mathsf{SPS}}, \mathsf{ChgRep}_{\mathsf{SPS}}, \mathsf{Vfy}_{\mathsf{SPS}}, \mathsf{VKey}_{\mathsf{SPS}})$ be an SPS-EQ scheme with relation R and vector length l,*
$\Pi_{\mathsf{SFPK}} = (\mathsf{KGen}_{\mathsf{SFPK}}, \mathsf{TKGen}_{\mathsf{SFPK}}, \mathsf{Sign}_{\mathsf{SFPK}}, \mathsf{ChkRep}_{\mathsf{SFPK}}, \mathsf{ChgPK}_{\mathsf{SFPK}}, \mathsf{ChgSK}_{\mathsf{SFPK}}, \mathsf{Vf}_{\mathsf{SFPK}})$ be an SFPK scheme with relation R' over the key space. Let
$(\mathsf{gpk}, \mathsf{gmsk}, \mathsf{gsk}) \leftarrow\!\!\$\ \mathsf{GKGen}(\lambda, n), \mathsf{gsk} = ((\mathsf{pk}_d, \mathsf{sk}_d, \sigma^{(d)}_{\mathsf{SPS}}))_{d=1}^{n}$. If there exist $i, j \in [n]$ such that

$$i \neq j \wedge \mathsf{pk}_i\ R'\ \mathsf{pk}_j$$

then gFPK-$\mathrm{GS}_{\Pi_{\mathsf{SFPK}}, \Pi_{\mathsf{SPS}}}$ *is not perfectly correct.*

Proof. Proof by contradiction, assume that *gFPK-*$GS_{\Pi_{\mathsf{SFPK}}, \Pi_{\mathsf{SPS}}}$ is perfectly correct. Without loss of generality we can assume $i < j$ (proof is analogous with roles of i and j flipped otherwise). Let m message, $\sigma = (\mathsf{pk}^*, \sigma^*_0, \sigma^*_{\mathsf{SPS}}) \leftarrow\!\!\$\ \mathsf{GSign}(\mathsf{gsk}[\mathsf{j}], m)$. Since *gFPK-*$GS_{\Pi_{\mathsf{SFPK}}, \Pi_{\mathsf{SPS}}}$ is perfectly correct, $\mathsf{Open}(\mathsf{gmsk}, m, \sigma) = j$ must hold. But since Open checks the public keys in the personal secret key vector in ascending order, it will check for $\mathsf{pk}^* \in [\mathsf{pk}_i]_{R'}$ before checking $\mathsf{pk}^* \in [\mathsf{pk}_j]_{R'}$. $\mathsf{pk}_i\ R'\ \mathsf{pk}_j$ yields $[\mathsf{pk}_i]_{R'} = [\mathsf{pk}_j]_{R'}$, so since $\mathsf{pk}^* \in [\mathsf{pk}_j]_{R'}$ by correctness of Π_{SFPK} (Eq. (3.17)) and definition of GSign (Def. 4.1), we get $\mathsf{pk}^* \in [\mathsf{pk}_i]_{R'}$ and therefore $\mathsf{Open}(\mathsf{gmsk}, m, \sigma) = i$. All in all we get

$$i = \mathsf{Open}(\mathsf{gmsk}, m, \sigma) = j$$

which is a contradiction to the assumption that $i \neq j$. □

So we see that if two users' SFPK public keys are related, *gFPK-GS* cannot be perfectly correct. This means that if we want to prove perfect correctness of *gFPK-GS*, we

need to assume that all the SFPK public keys in the personal secret keys are pairwise unrelated. However, an open question is whether computational correctness (Def. 3.9) of $gFPK\text{-}GS_{\Pi_{\mathsf{SFPK}},\Pi_{\mathsf{SPS}}}$ can be proven without any further assumptions on the relation R Π_{SFPK} is defined over as well as the key generation algorithm $\mathsf{KGen}_{\mathsf{SFPK}}$ of Π_{SFPK}. The crucial task here is to estimate the likelihood of two SFPK public keys from different personal secret keys being related when generating the key material during setup. This question will be answered in Sect. 4.2.

Next we discuss how the internal randomness of GSign influences the creation of signatures with *gFPK-GS*. As an outcome of this discussion we obtain a formal definition of a connection between different group signatures for the same message created by the same user. This will be very useful in the formal proof of full-anonymity of *gFPK-GS* in Thm. 5.3. We see that a group signature for a message m created by user i using *gFPK-GS* is fully determined by

- m
- the randomness r explicitly drawn in GSign to randomize the public key in the personal secret key of user i
- the internal randomness used in the $\mathsf{Sign}_{\mathsf{SFPK}}$ call inside GSign

So even if we fix a message m and user i, there can still be different outcomes of GSign when this user signs this message. Yet all the signatures that can possibly be produced when GSign is called on this input contain public keys from the same equivalence class, each with a valid SPS-EQ certificate and a valid SFPK signature. So we see that randomizing the SFPK public key differently alters all three components of the resulting group signature. But even if the message m, the signer identity i and the randomness r drawn in GSign are all three fixed, there can be different outcomes when calling GSign multiple times, depending on the internal randomness of $\mathsf{Sign}_{\mathsf{SFPK}}$. These considerations are formalized in the following definition.

Definition 4.4 *Let* $\text{gFPK-GS}_{\Pi_{\mathsf{SFPK}},\Pi_{\mathsf{SPS}}}$ *be defined like in Def.* 4.1 *with* R *being the relation* Π_{SFPK} *is defined over. Let* m *be a message,* $i \in [n]$ *a user,*
$\sigma := (\mathsf{pk}, \sigma_0, \sigma_{\mathsf{SPS}}) \leftarrow_{\$} \mathsf{GSign}(\mathsf{gsk}[\mathrm{i}], m)$ *with* pk *being an SFPK public key,*
$\mathsf{Vfy}_{\mathsf{SPS}}(\mathsf{pk}, \sigma_{\mathsf{SPS}}, \mathsf{pk}_{\mathsf{SPS}}) = 1$ *,*$\sigma_0 \leftarrow_{\$} \mathsf{Sign}_{\mathsf{SFPK}}(\mathsf{sk}, m||\sigma_{\mathsf{SPS}}||\mathsf{pk})$ *and* sk *being a corresponding SFPK secret key to* pk*. We call a valid signature* $\sigma^* = (\mathsf{pk}^*, \sigma_0^*, \sigma_{\mathsf{SPS}}^*)$ *a* randomized version of σ *if and only if*

- $\mathsf{pk}\ R\ \mathsf{pk}^*$
- $\sigma \neq \sigma^*$

Note that the two triples σ, σ^* are considered distinct if they differ in at least one component. It is easy to see that all signatures for a fixed message that a fixed user creates are randomized versions of each other (except for the case that the $\mathsf{Sign}_{\mathsf{SFPK}}$ call inside GSign uses the same randomness twice). As it can be seen in Def. 4.1, a triple $\sigma^* = (\mathsf{pk}^*, \sigma_0^*, \sigma_{\mathsf{SPS}}^*)$ is a valid group signature for a message m with respect to *gFPK-GS*

if the GVf-algorithm of *gFPK-GS* returns 1 when called upon input (m, σ^*) and the group public key which is the case if and only if

$$\mathsf{Vfy}_{\mathsf{SPS}}(\mathsf{pk}^*, \sigma^*_{\mathsf{SPS}}, \mathsf{pk}_{\mathsf{SPS}}) = 1$$

and

$$\mathsf{Vf}_{\mathsf{SFPK}}(\mathsf{pk}^*, M^*, \sigma^*_0) = 1$$

hold, with $M^* := m||\sigma^*_{\mathsf{SPS}}||\mathsf{pk}^*$. It is important to notice that in Def. 4.4, the case of

$$\mathsf{pk} = \mathsf{pk}^* \wedge \sigma_{\mathsf{SPS}} = \sigma^*_{\mathsf{SPS}} \wedge \sigma_0 \neq \sigma^*_0$$

can occur, so when randomizing a signature $\sigma := (\mathsf{pk}, \sigma_0, \sigma_{\mathsf{SPS}})$ for a message m made with *gFPK-GS*, it is sufficient to replace the contained SFPK signature σ_0 with a distinct valid SFPK signature for $m||\sigma_{\mathsf{SPS}}||\mathsf{pk}$. This correlates to using the same input and internal randomness twice for GSign but altering the internal randomness of the $\mathsf{Sign}_{\mathsf{SFPK}}$-call. On the other hand, we see that when randomizing a *gFPK-GS*-signature created by user i, it is inevitable to avoid computing an SFPK signature with the SFPK secret key sk_i which is part of the personal secret key $\mathsf{gsk}[\mathsf{i}]$ of user i. This is because the SFPK signature σ_0 needs to be replaced in any case when randomizing a signature σ. So this yields that if the SFPK scheme Π_{SFPK} used to instantiate $gFPK\text{-}GS_{\Pi_{\mathsf{SFPK}},\Pi_{\mathsf{SPS}}}$ is existentially unforgeable, no one except user i itself can randomize a *gFPK-GS*-signature created by user i.

4.2 Correctness

In the following we will discuss under which assumptions *gFPK-GS* can be perfect and computationally correct. Because of the discussion in Sect. 4.1 which resulted in Lem. 4.3, we assume for the proof of perfect correctness that all SFPK public keys in the users' personal secret keys are pairwise unrelated. Despite being a more or less simple computation under this prerequisite, proving its perfect correctness helps a lot with getting familiar with *gFPK-GS*.

Theorem 4.5 *Let $l \in \mathbb{N}$, $l > 1$, R, R' be compatible equivalence relations, $\lambda \in \mathbb{N}$ security parameter, $\Pi_{\mathsf{SPS}} = (\mathsf{BGGen}_{\mathsf{SPS}}, \mathsf{KGen}_{\mathsf{SPS}}, \mathsf{Sign}_{\mathsf{SPS}}, \mathsf{ChgRep}_{\mathsf{SPS}}, \mathsf{Vfy}_{\mathsf{SPS}}, \mathsf{VKey}_{\mathsf{SPS}})$ be a correct SPS-EQ scheme with relation R and vector length l,*
$\Pi_{\mathsf{SFPK}} = (\mathsf{KGen}_{\mathsf{SFPK}}, \mathsf{TKGen}_{\mathsf{SFPK}}, \mathsf{Sign}_{\mathsf{SFPK}}, \mathsf{ChkRep}_{\mathsf{SFPK}}, \mathsf{ChgPK}_{\mathsf{SFPK}}, \mathsf{ChgSK}_{\mathsf{SFPK}}, \mathsf{Vf}_{\mathsf{SFPK}})$ be a correct SFPK scheme with relation R' over the key space. Furthermore we assume that for all $n, i, j \in \mathbb{N}, (\cdot, \cdot, \mathsf{gsk}) \leftarrow\!\!\$\ \mathsf{GKGen}(\lambda, n)$ with $\mathsf{gsk} = ((\mathsf{pk}_d, \mathsf{sk}_d, \sigma^{(d)}_{\mathsf{SPS}}))_{d=1}^n$ we have

$$\mathsf{pk}_i \not R' \mathsf{pk}_j$$

Then $\text{gFPK-GS}_{\Pi_{\mathsf{SFPK}},\Pi_{\mathsf{SPS}}}$ is a perfectly correct group signature scheme.

Proof. Let

- $n \in \mathbb{N}$ group size
- $BG := (p, G_1, G_2, G_T, e, g_1, g_2) := \mathsf{BGGen}_{\mathsf{SPS}}(\lambda)$
- $(\mathsf{pk}_{\mathsf{SPS}}, \mathsf{sk}_{\mathsf{SPS}}) \leftarrow\!\!\$\, \mathsf{KGen}_{\mathsf{SPS}}(BG, l)$
- $\forall i \in [n]$:
  - $\omega_i \leftarrow\!\!\$\, \mathsf{coin}$
  - $(\mathsf{pk}_i, \mathsf{sk}_i, \tau_i) \leftarrow\!\!\$\, \mathsf{TKGen}_{\mathsf{SFPK}}(\lambda, \omega_i)$
  - $\sigma_{\mathsf{SPS}}^{(i)} \leftarrow\!\!\$\, \mathsf{Sign}_{\mathsf{SPS}}(\mathsf{sk}_{\mathsf{SPS}}, \mathsf{pk}_i)$
- $\mathsf{gpk} := (BG, \mathsf{pk}_{\mathsf{SPS}})$ be the group public key
- $\mathsf{gmsk} := ((\tau_i, \mathsf{pk}_i))_{i=1}^n$ the group manager secret key
- $\mathsf{gsk}[\mathsf{i}] := (\mathsf{pk}_i, \mathsf{sk}_i, \sigma_{\mathsf{SPS}}^{(i)})$ as the i-th group member's personal secret key (for all identities $i \in [n]$)

To prove perfect correctness of $gFPK\text{-}GS_{\Pi_{\mathsf{SFPK}}, \Pi_{\mathsf{SPS}}}$ according to Def. 3.8, we need to prove that for all $i \in [n]$, messages m

$$\Pr[\mathsf{GVf}(\mathsf{gpk}, m, \mathsf{GSign}(\mathsf{gsk}[\mathsf{i}])) = 1] = 1$$

and

$$\Pr[\mathsf{Open}(\mathsf{gmsk}, m, \mathsf{GSign}(\mathsf{gsk}[\mathsf{i}], m)) = i] = 1$$

We start with proving the first equation. Let

- $i \in [n]$
- m arbitrary message
- $r \in \mathsf{coin}$ internal randomness of GSign used for key randomization and certificate adaption.
- $\mathsf{pk}'_i \leftarrow\!\!\$\, \mathsf{ChgPK}_{\mathsf{SFPK}}(\mathsf{pk}_i, r)$ randomized public key
- $\mathsf{sk}'_i \leftarrow\!\!\$\, \mathsf{ChgSK}_{\mathsf{SFPK}}(\mathsf{sk}_i, r)$ adapted secret key
- $(\mathsf{pk}''_i, {\sigma_{\mathsf{SPS}}^{(i)}}') \leftarrow\!\!\$\, \mathsf{ChgRep}_{\mathsf{SPS}}(\mathsf{pk}_i, \sigma_{\mathsf{SPS}}^{(i)}, r, \mathsf{pk}_{\mathsf{SPS}}))$ the randomized public key with an adapted certificate (due to compatibility of the equivalence relations, the resulting public key is equal to the randomized one from above, so we get $\mathsf{pk}'_i = \mathsf{pk}''_i$)
- $M := m || {\sigma_{\mathsf{SPS}}^{(i)}}' || \mathsf{pk}'_i$
- $\sigma_0 \leftarrow\!\!\$\, \mathsf{Sign}_{\mathsf{SFPK}}(\mathsf{sk}'_i, M)$ the resulting SFPK-signature
- $\sigma \leftarrow\!\!\$\, \mathsf{GSign}(\mathsf{gsk}[\mathsf{i}], m) := (\mathsf{pk}'_i, \sigma_0, {\sigma_{\mathsf{SPS}}^{(i)}}')$ the resulting group signature

We see that

$$\begin{aligned}
&\Pr[\mathsf{GVf}(\mathsf{gpk}, m, \mathsf{GSign}(\mathsf{gsk}[\mathrm{i}], m)) = 1] && = 1\\
\overset{\mathsf{gpk},\sigma}{\Leftrightarrow} &\Pr[\mathsf{GVf}((BG, \mathsf{pk_{SPS}}), m, (\mathsf{pk}'_i, \sigma_0, {\sigma^{(i)}_{\mathsf{SPS}}}')) = 1] && = 1\\
\overset{\mathsf{GVf}}{\Leftrightarrow} &\Pr[\mathsf{Vf_{SFPK}}(\mathsf{pk}'_i, M, \sigma_0) = 1] = 1 \wedge \Pr[\mathsf{Vfy_{SPS}}(\mathsf{pk}'_i, {\sigma^{(i)}_{\mathsf{SPS}}}', \mathsf{pk_{SPS}}) = 1] && = 1\\
\Leftrightarrow &\Pr[\mathsf{Vfy_{SPS}}(\mathsf{pk}'_i, {\sigma^{(i)}_{\mathsf{SPS}}}', \mathsf{pk_{SPS}}) = 1] && = 1\\
\Leftrightarrow &1 && = 1
\end{aligned}$$

which proves the first equation (Eq. (3.6)). The last equivalence comes from the fact that $(\mathsf{pk}'_i, {\sigma^{(i)}_{\mathsf{SPS}}}') \leftarrow_\$ \mathsf{ChgRep_{SPS}}(\mathsf{pk}_i, \sigma^{(i)}_{\mathsf{SPS}}, r, \mathsf{pk_{SPS}})$ so because Π_{SPS} is a correct SPS-EQ scheme, Eq. (3.26) holds which yields that $(\mathsf{pk}'_i, {\sigma^{(i)}_{\mathsf{SPS}}}')$ is a valid message-signature pair under $\mathsf{pk_{SPS}}$ so we get above equivalence. The second last equivalence holds because of the correctness of Π_{SFPK}, since σ_0 is a signature created using sk'_i and $(\mathsf{pk}'_i, \mathsf{sk}'_i)$ is a key pair created by randomizing both parts of the valid key pair $(\mathsf{pk}_i, \mathsf{sk}_i)$ using the same randomness r. So in formulas we have $\sigma_0 \leftarrow_\$ \mathsf{Sign_{SFPK}}(\mathsf{sk}'_i, M), \mathsf{pk}'_i \leftarrow_\$ \mathsf{ChgPK_{SFPK}}(\mathsf{pk}_i, r)$, $\mathsf{sk}'_i \leftarrow_\$ \mathsf{ChgSK_{SFPK}}(\mathsf{sk}_i, r)$, since Π_{SFPK} is correct we can use Eq. (3.15) to argue $\mathsf{Vf_{SFPK}}(\mathsf{pk}'_i, M, \sigma_0) = 1$ which yields above equivalence. We continue to prove the second equation (Eq. (3.7)). We see that

$$\begin{aligned}
&\Pr[\mathsf{Open}(\mathsf{gmsk}, m, \mathsf{GSign}(\mathsf{gsk}[\mathrm{i}], m)) = i]\\
=&\Pr[\mathsf{Open}(((\tau_i, \mathsf{pk}_i))_{i=1}^n, m, (\mathsf{pk}'_i, \sigma_0, {\sigma^{(i)}_{\mathsf{SPS}}}')) = i]\\
=&1
\end{aligned}$$

holds, which proves the second equation (Eq. (3.7)). The last equality deserves some words. For Open to return an identity in the first place, $\mathsf{GVf}(\mathsf{gpk}, m, \mathsf{GSign}(\mathsf{gsk}[\mathrm{i}], m)) = 1$ must hold. Because we already proved Eq. (3.6) for $gFPK\text{-}GS_{\Pi_{\mathsf{SFPK}},\Pi_{\mathsf{SPS}}}$, this is fulfilled and the only thing left to argue (by definition of Open in Def. 4.1) is therefore $\mathsf{ChkRep_{SFPK}}(\tau_i, \mathsf{pk}'_i) = 1$.

Since Π_{SFPK} is correct this is equivalent to $\mathsf{pk}'_i \in [\mathsf{pk}_i]_{R'}$ (Eq. (3.16)). By definition we have $\mathsf{pk}'_i \leftarrow_\$ \mathsf{ChgPK_{SFPK}}(\mathsf{pk}_i, r)$, so since Π_{SFPK} is correct (Eq. (3.17)) this means $\mathsf{pk}'_i \in [\mathsf{pk}_i]_{R'}$. Since pk'_i is in no other equivalence class $[\mathsf{pk}_j]_{R'}$ for $j \neq i$ (because we assumed all SFPK public keys in the personal secret keys to be pairwise unrelated), we get the desired equality. Since we proved both Eq. (3.6) and Eq. (3.7) for $gFPK\text{-}GS_{\Pi_{\mathsf{SFPK}},\Pi_{\mathsf{SPS}}}$, this means that $gFPK\text{-}GS_{\Pi_{\mathsf{SFPK}},\Pi_{\mathsf{SPS}}}$ is a perfectly correct group signature scheme according to Def. 3.8 under the assumption that all SFPK public keys in the users' personal secret keys are pairwise unrelated. □

Computational correctness Computational correctness of $gFPK\text{-}GS_{\Pi_{\mathsf{SFPK}},\Pi_{\mathsf{SPS}}}$ (see Def. 3.9) is heavily dependent on the equivalence relation which the SFPK scheme Π_{SFPK} is defined over as well as the key generation algorithm $\mathsf{TKGen_{SFPK}}$ of Π_{SFPK}. To formally

prove computational correctness of *gFPK-GS* according to Def. 3.9, we need to prove

$$\Pr[\mathsf{GVf}(\mathsf{gpk}, m, \mathsf{GSign}(\mathsf{gsk}[\mathrm{i}])) = 1] = 1 - \kappa_1(\lambda)$$

and

$$\Pr[\mathsf{Open}(\mathsf{gmsk}, m, \mathsf{GSign}(\mathsf{gsk}[\mathrm{i}], m)) = i] = 1 - \kappa_2(\lambda)$$

for κ_1, κ_2 being negligible functions in the security parameter λ and all other variables being defined as in the theorem for perfect correctness of *gFPK-GS* (see Thm. 4.5). We see that since *gFPK-GS* fulfills the first requirement for group signature correctness (Eq. (3.6)) without the assumption that all SFPK public keys in the personal secret keys are pairwise unrelated, it also fulfills the first requirement of computational correctness (Eq. (3.8)) with $\kappa_1 = 0$ (zero function, mapping all inputs to 0). However, it cannot be formally proven or ruled out that $gFPK\text{-}GS_{\Pi_{\mathsf{SFPK}},\Pi_{\mathsf{SPS}}}$ fulfills the second of above requirements for computational correctness. This is because the probability $p_{\mathsf{keys-rel}}^{gFPK\text{-}GS_{\Pi_{\mathsf{SFPK}},\Pi_{\mathsf{SPS}}}}$ of GKGen outputting two personal secret keys containing related SFPK public keys can vary dependent on the concrete SFPK scheme Π_{SFPK} used for the instantiation of $gFPK\text{-}GS_{\Pi_{\mathsf{SFPK}},\Pi_{\mathsf{SPS}}}$, more precisely, depending on the key generation algorithm $\mathsf{TKGen}_{\mathsf{SFPK}}$ of Π_{SFPK} as well as the equivalence relation R that Π_{SFPK} is defined over. So $p_{\mathsf{keys-rel}}^{gFPK\text{-}GS_{\Pi_{\mathsf{SFPK}},\Pi_{\mathsf{SPS}}}}$ cannot be computed in the general case, even if we fix the equivalence relation Π_{SFPK} is defined over to R_{exp}.

A trivial approach to fix the issue with the possibly related SFPK public keys is to execute $\mathsf{TKGen}_{\mathsf{SFPK}}$ during GKGen of *gFPK-GS* until it has output n SFPK key pairs containing public keys that are pairwise unrelated. However, if GKGen was changed like this, its running time could not be estimated properly, since $p_{\mathsf{keys-rel}}^{gFPK\text{-}GS_{\Pi_{\mathsf{SFPK}},\Pi_{\mathsf{SPS}}}}$ is still not computable in the general case and therefore we cannot estimate the number of times $\mathsf{TKGen}_{\mathsf{SFPK}}$ will be executed during GKGen. It might not even be guaranteed that this changed version of GKGen terminates since the key space of Π_{SFPK} and the equivalence relation R Π_{SFPK} is defined over could be defined in a way that n pairwise unrelated SFPK public keys do not even exist. However, the probability of GKGen exceeding polynomial running time must be negligible by definition (see Def. 3.7). This means that if we changed GKGen of $gFPK\text{-}GS_{\Pi_{\mathsf{SFPK}},\Pi_{\mathsf{SPS}}}$ (Def. 4.1) in the way sketched above, $gFPK\text{-}GS_{\Pi_{\mathsf{SFPK}},\Pi_{\mathsf{SPS}}}$ would no longer be a group signature scheme according to Def. 3.7.

So all in all, in Sect. 4.2 we proved perfect correctness of *gFPK-GS* under the assumption that all SFPK public keys contained in the personal secret keys output by GKGen are pairwise unrelated. Computational correctness according to Def. 3.9 cannot be proven formally for the generic construction by Backes et al. [3] (Def. 4.1 in this thesis) and must therefore be proven individually for concrete instantiations of *gFPK-GS*.

4.3 Changes to original work

In this section, all differences regarding the original definition of *gFPK-GS* that was given in [3] will be listed and discussed. Note that [2] is an extended version of [3] in

which some additions to the original work in [3] are made which we will also cover in this section.

- The correctness proof for *gFPK-GS* was omitted in [3], a high-level proof sketch was added in [2]. A detailed correctness proof for *gFPK-GS* can be found under Thm. 4.5 in this thesis.

- Backes et al. did not mention in [3] that the SFPK public keys in all users' personal secret keys must be pairwise unrelated for *gFPK-GS* to be perfectly correct (see Lem. 4.3 in this thesis). So the version of *gFPK-GS* presented as Scheme 2 in [3] is not perfectly correct without further assumptions. The proof of perfect correctness in this thesis (see Thm. 4.5) therefore additionally assumes that all SFPK public keys in the personal secret keys are in pairwise distinct equivalence classes. Proving computational correctness of the generic construction *gFPK-GS* according to Def. 3.9 is not possible, since the probability $p_{\mathsf{keys-rel}}^{gFPK\text{-}GS_{\Pi_{\mathsf{SFPK}},\Pi_{\mathsf{SPS}}}}$ of GKGen outputting two personal secret keys with related SFPK public keys cannot be estimated without knowledge of the concrete SFPK scheme Π_{SFPK} used to instantiate *gFPK-GS* (see paragraph "Computational correctness" in Sect. 4.2 for details).

- The notion of compatibility of equivalence relations (Def. 4.2 in this thesis) was not formally introduced in [3]. In the introduction of their paper, the authors mentioned that the relations the schemes $\Pi_{\mathsf{SFPK}}, \Pi_{\mathsf{SPS}}$ used to instantiate *gFPK-GS* are defined over need to be compatible, but what this term exactly means was not explained. We cannot assume without loss of generality that $\mathsf{ChgPK}_{\mathsf{SFPK}}$ and $\mathsf{ChgRep}_{\mathsf{SPS}}$ will always yield the same public key when called with the same randomness and SFPK public key (which is seen as a message by $\mathsf{ChgRep}_{\mathsf{SPS}}$ in this use case) or do so with high probability. This behaviour of $\mathsf{ChgPK}_{\mathsf{SFPK}}$ and $\mathsf{ChgRep}_{\mathsf{SPS}}$ is a requirement for *gFPK-GS* to be perfectly correct. For *gFPK-GS* to be computationally correct according to Def. 3.9, above behaviour must at least be achieved with high probability. One might call this requirement *computational compatibility*. However, since computational correctness of $gFPK\text{-}GS_{\Pi_{\mathsf{SFPK}},\Pi_{\mathsf{SPS}}}$ cannot be proven without specifying Π_{SFPK} and Π_{SPS} (and is therefore not proven in this work, see paragraph on computational correctness in Sect. 4.2 for details), computational compatibility is not defined formally in this thesis.

  A formal definition of compatibility of equivalence relations similar to the one in this thesis (Def. 4.2) is still missing in [2], although in the sketch of the correctness proof for *gFPK-GS* (Theorem 3 in [2]), the equivalence relations of the SFPK and SPS-EQ are said to fulfill the property which was called compatibility in this thesis. However, the authors argue this using the implicit prerequisite that both schemes are instantiated with the same equivalence relation while the compatibility definition in this thesis also captures the possibility that their relations are different.

- In the group key generation algorithm of Scheme 2 in [3] (which we refer to as *gFPK-GS*), it is denoted that the execution of BGGen is probablistic with internal randomness. But according to the authors' definition of bilinear group generators (Definition 2 in [3]), BGGen is a deterministic algorithm.

5 Security of group signature by Backes et al.

In this section, full proof of the full-traceability and full-anonymity of *gFPK-GS* under the prerequisites named by Backes et al. [3] will be given. The authors gave the game sequences they used in their proofs as well as a high-level view on the arguments needed for a formal proof. This can both be found in the appendix of their paper. A discussion of their approaches can be found in Sect. 5.3. Before beginning with the actual proofs, we are going to define two terms that will help us a lot when it comes to arguing that a specific security game is simulated correctly to an adversary in a reduction.

Definition 5.1 (**view, correctly distributed**) *The tuple that contains all random variables an adversary is input during a security game in order of appearance is called the* view *of that adversary. We say that a random variable that is input to an adversary in a simulation of a security game during a reduction is* correctly distributed *if it is identically distributed as in the respective security game.*

Note that according to above terminology the view of an adversary is a random variable itself. So when simulating security games (as done in a reduction, for example), we always need to prove that the view of the adversary the game is simulated for is correctly distributed.

5.1 Full-traceability

We will start with proving full-traceability of *gFPK-GS*. The following theorem can be found as Theorem 3 in the original paper [3]. In the following we will adjust it to the notation used in this thesis. Before the formal proof a short intuition of the general idea of it will be given. Note that the approach of the proof in this thesis is different from the one chosen in [3] which is based on a sequence of games.

Theorem 5.2 *Let R, R' be compatible equivalence relations (Def. 4.2), $\lambda \in \mathbb{N}$ security parameter, $n \in \mathbb{N}$ group size, $l \in \mathbb{N}, l > 1$. Let*
$\Pi_{\mathsf{SPS}} = (\mathsf{BGGen}_{\mathsf{SPS}}, \mathsf{KGen}_{\mathsf{SPS}}, \mathsf{Sign}_{\mathsf{SPS}}, \mathsf{ChgRep}_{\mathsf{SPS}}, \mathsf{Vfy}_{\mathsf{SPS}}, \mathsf{VKey}_{\mathsf{SPS}})$ *be an SPS-EQ scheme with relation R and vector length l,*
$\Pi_{\mathsf{SFPK}} = (\mathsf{KGen}_{\mathsf{SFPK}}, \mathsf{TKGen}_{\mathsf{SFPK}}, \mathsf{Sign}_{\mathsf{SFPK}}, \mathsf{ChkRep}_{\mathsf{SFPK}}, \mathsf{ChgPK}_{\mathsf{SFPK}}, \mathsf{ChgSK}_{\mathsf{SFPK}}, \mathsf{Vf}_{\mathsf{SFPK}})$ *be an SFPK scheme with relation R' over the key space. Furthermore let Π_{SFPK} be existentially unforgeable (Def. 3.20), Π_{SPS} be existentially unforgeable (Def. 3.23). Then* $\text{gFPK-GS}_{\Pi_{\mathsf{SFPK}},\Pi_{\mathsf{SPS}}}$ *(Def. 4.1) is fully-traceable according to Def. 3.13.*

The unforgeability of Π_{SFPK} and Π_{SPS} are seperated in the formulation of theorem 5.1 because despite they share the same name, these are different security properties.

Proof. Let us first outline the proof. Two ways to win the full-traceability experiment from Def. 3.13 exist which directly emerge from the definition: an adversary that wins must either have output an unopenable signature or a signature that opens to an identity which is not part of its collusion set. In case that an adversary manages to forge a valid and unopenable group signature, the public key and the certificate in this group signature can be used to break existential unforgeability of the SPS-EQ scheme. This is because then the public key contained in the group signature is not related to any of the group members public keys while the group signature contains a valid SPS-EQ certificate for it. On the other hand, if the forgery opens to an identity $i \in [n]$, it can be used to create a forgery for the SPFK key pair in the i-th user's personal secret key $\mathsf{gsk}[\mathrm{i}]$, therefore breaking the SFPK unforgeability. So in this case the basic idea of the reduction is to embed the challenge key pair into the group secret key vector as one of the group members key pairs. In the following we do the case distinction sketched above and therefore describe the construction of two adversaries, for every execution of an algorithm breaking full-traceability, exactly one of them has non-negligible advantage. Formally this means we need to prove

$$\begin{aligned} &\exists \text{ PPT adversary } \mathcal{A} : Adv^{\mathsf{trace}}_{gFPK\text{-}GS_{\Pi_{\mathsf{SFPK}},\Pi_{\mathsf{SPS}}},\mathcal{A}}(\lambda, n) \text{ not negligible} \\ \Rightarrow &\exists \text{ PPT adversary } \mathcal{B} : Adv^{\mathsf{euf-sfpk}}_{\Pi_{\mathsf{SFPK}},\mathcal{B}}(\lambda) \text{ not negligible} \\ \vee &\exists \text{ PPT adversary } \mathcal{D} : Adv^{\mathsf{euf-sps}}_{\Pi_{\mathsf{SPS}},\mathcal{D}}(\lambda) \text{ not negligible} \end{aligned}$$

which is equivalent to proving

$$\begin{aligned} &\exists \text{ PPT adversary } \mathcal{A} : Adv^{\mathsf{trace}}_{gFPK\text{-}GS_{\Pi_{\mathsf{SFPK}},\Pi_{\mathsf{SPS}}},\mathcal{A}}(\lambda, n) \text{ not negligible} \\ \Rightarrow &\exists \text{ PPT adversary } \mathcal{B}, \text{ PPT adversary } \mathcal{D} : Adv^{\mathsf{euf-sfpk}}_{\Pi_{\mathsf{SFPK}},\mathcal{B}}(\lambda) \text{ or } Adv^{\mathsf{euf-sps}}_{\Pi_{\mathsf{SPS}},\mathcal{D}}(\lambda) \text{ not negligible} \end{aligned}$$

So in the following, two adversaries $\mathcal{B}, \mathcal{D}$ will be constructed from an adversary $\mathcal{A}$ which has non-negligible advantage in breaking full-traceability of $gFPK\text{-}GS_{\Pi_{\mathsf{SFPK}},\Pi_{\mathsf{SPS}}}$. $\mathcal{B}$ will be able to break the existential unforgeability of Π_{SFPK} (with non-negligible advantage) in the case that the forgery output by $\mathcal{A}$ opens to an identity $k \in [n]$. $\mathcal{D}$ will be able to break the existential unforgeability of Π_{SPS} (also with non-negligible advantage) under the prerequisite that the forgery output by $\mathcal{A}$ is unopenable. Since $\mathcal{A}$ must either output a forgery that opens to an identity or is unopenable, we see that the construction of adversaries $\mathcal{B}, \mathcal{D}$ as above allows for proving the above reformulation of the contraposition of Thm. 5.2.

Next we construct an adversary B breaking unforgeability of Π_{SFPK} from an adversary $\mathcal{A}$ breaking full-traceability of $gFPK\text{-}GS_{\Pi_{\mathsf{SFPK}},\Pi_{\mathsf{SPS}}}$ whose forgery opens to an identity $k \in [n]$. On input (pk, τ) with $(\mathsf{pk}, \mathsf{sk}, \tau) \leftarrow \mathsf{TKGen}_{\mathsf{SFPK}}(\lambda, \omega)$,$\omega \in \mathsf{coin}$ like in the unforgeability game from Def. 3.20, $\mathcal{B}$ works as follows:

1. $\mathcal{B}$ sets up the traceability challenge for $\mathcal{A}$, performing the following steps:
   a) $\mathcal{B}$ samples an identity $i \in [n]$ and initializes the collusion set as $\mathsf{C} := \emptyset$

b) $\mathcal{B}$ generates the bilinear group $BG \leftarrow \mathsf{BGGen}(\lambda)$ and an SPS-EQ key pair $(\mathsf{pk_{SPS}}, \mathsf{sk_{SPS}}) \leftarrow_\$ \mathsf{KGen_{SPS}}(BG, l)$. Note that since BGGen is deterministic (see Def. 3.6) and λ publicly known, BG can be regarded a publicly known value.

c) for all identities $j \in [n], j \neq i$, $\mathcal{B}$ generates the personal secret key and a trapdoor as follows

i. $\mathcal{B}$ draws random coins $\omega_j \leftarrow_\$ \mathsf{coin}$

ii. $\mathcal{B}$ generates an SFPK key pair with a trapdoor by computing $(\mathsf{pk}_j, \mathsf{sk}_j, \tau_j) \leftarrow_\$ \mathsf{TKGen_{SFPK}}(\lambda, \omega_j)$

iii. $\mathcal{B}$ creates a certificate by signing the generated SFPK public key like $\sigma^{(j)}_{\mathsf{SPS}} \leftarrow_\$ \mathsf{Sign_{SPS}}(\mathsf{sk_{SPS}}, \mathsf{pk}_j)$

iv. $\mathcal{B}$ defines the personal secret key of group member j as $\mathsf{gsk[j]} \leftarrow_\$ (\mathsf{pk}_j, \mathsf{sk}_j, \sigma^{(j)}_{\mathsf{SPS}})$

d) $\mathcal{B}$ defines $\mathsf{pk}_i := \mathsf{pk}, \tau_i := \tau$ and creates a fresh certificate $\sigma^{(i)}_{\mathsf{SPS}} \leftarrow_\$ \mathsf{Sign_{SPS}}(\mathsf{sk_{SPS}}, \mathsf{pk}_i)$ for pk. Note that this implicitly sets $\mathsf{sk}_i := \mathsf{sk}$, where sk is the SFPK secret key from the SFPK unforgeability challenge for $\mathcal{B}$. This means that we can see the personal secret key of group member i as fully defined, although $\mathcal{B}$ cannot access sk

e) $\mathcal{B}$ defines the group public key as $\mathsf{gpk} := (BG, \mathsf{pk_{SPS}})$

f) $\mathcal{B}$ defines the group manager secret key as $\mathsf{gmsk} := ((\tau_j, \mathsf{pk}_j))_{j=1}^n$

g) $\mathcal{B}$ defines the group secret key vector as $\mathsf{gsk} := (\mathsf{gsk[j]})_{j=1}^n$

2. $\mathcal{B}$ starts $\mathcal{A}$ on input $(\mathsf{choose}, \mathsf{St})$ with $\mathsf{St} := (\mathsf{gpk}, \mathsf{gmsk})$ and answers $\mathcal{A}$'s oracle queries in the choose-phase as follows:

collusion query : if $\mathcal{A}$ sends an identity $j \in [n]$, $\mathcal{B}$ adds j to the collusion set C ($\mathsf{C} := \mathsf{C} \cup \{j\}$) and answers with $\mathsf{gsk[j]}$. It is plain to see that in the case $j = i$, $\mathcal{B}$ must abort the game (since $\mathcal{B}$ cannot access $\mathsf{sk}_i = \mathsf{sk}$ which is part of $\mathsf{gsk[i]}$), we will discuss the details below.

signing query : when receiving a signing query from $\mathcal{A}$ (consisting of an identity $j \in [n]$ and a message m), $\mathcal{B}$ performs the following steps:

a) $\mathcal{B}$ draws randomness $r \leftarrow_\$ \mathsf{coin}$

b) $\mathcal{B}$ randomizes pk_j like $\mathsf{pk}'_j \leftarrow_\$ \mathsf{ChgPK_{SFPK}}(\mathsf{pk}_j, r)$ and adapts the corresponding certificate by computing $(\mathsf{pk}'_j, {\sigma^{(j)}_{\mathsf{SPS}}}') \leftarrow_\$ \mathsf{ChgRep_{SPS}}(\mathsf{pk}_j, \sigma^{(j)}_{\mathsf{SPS}}, r, \mathsf{pk_{SPS}})$

c) $\mathcal{B}$ defines $M := m || {\sigma^{(j)}_{\mathsf{SPS}}}' || \mathsf{pk}'_j$

d) for creating the required SFPK signature, two cases exist:

$j = i$ $\mathcal{B}$ uses its own oracle (which uses sk_i) to generate the SFPK signature, more precisely it submits (M, r) to the randomize-then-sign oracle and obtains $\sigma_0 \leftarrow_\$ \mathsf{Sign_{SFPK}}(\mathsf{sk}', m)$ where $\mathsf{sk}' \leftarrow_\$ \mathsf{ChgSK_{SFPK}}(\mathsf{sk}, r)$.

This results in (M, σ_0) being added to the set Q of query-answer tuples of the oracle.

$j \neq i$ $\mathcal{B}$ obtains the required SFPK signature by computing $\sigma_0 \leftarrow_\$ \mathsf{Sign}_{\mathsf{SFPK}}(\mathsf{sk}'_j, M)$ where $\mathsf{sk}'_j \leftarrow_\$ \mathsf{ChgSK}_{\mathsf{SFPK}}(\mathsf{sk}_j, r)$

e) $\mathcal{B}$ puts together the answer to $\mathcal{A}$'s query as $\sigma := (\mathsf{pk}'_j, \sigma_0, \sigma_{\mathsf{SPS}}^{(j)})$ and returns it

Eventually, $\mathcal{A}$ halts and outputs state information St

3. $\mathcal{B}$ starts $\mathcal{A}$ on input $(\mathsf{guess}, \mathsf{St})$ and answers $\mathcal{A}$'s signing queries like in the choose-phase

4. when $\mathcal{A}$ eventually outputs a forgery (m^*, σ^*), $\mathcal{B}$ will create his candidate forgery for Π_{SFPK} as follows:
   a) $\mathcal{B}$ parses $\sigma^* := (\mathsf{pk}^*, \sigma_0^*, \sigma_{\mathsf{SPS}}^*)$
   b) $\mathcal{B}$ defines $M^* := m^* || \sigma_{\mathsf{SPS}}^* || \mathsf{pk}^*$
   c) $\mathcal{B}$ outputs $(\mathsf{pk}^*, M^*, \sigma_0^*)$

The values choose and guess are constants indicating the current phase of the full-traceability game. In the first step, $\mathcal{B}$ basically performs a slightly adapted version of GKGen: generation of $\mathsf{gsk}[i]$ is just rearranging the values input to $\mathcal{B}$, only the SPS-EQ certificate for the public key needs to generated manually. All other personal secret keys are generated according to the definition of GKGen. Answering the collusion queries then is straightforward except for the case $j = i$, which will be dealt with in the later analysis. In short, if $\mathcal{A}$ wins and $\mathcal{B}$ is lucky with guessing i, $\mathcal{A}$ will never query i in the collusion phase. This follows from the fact that $\mathcal{A}$ creates a forgery that opens to an identity $k \in [n]$ so there is at least one identity which he cannot have queried when he wins the game. For answering signing queries for identities $j \neq i$, $\mathcal{B}$ can simply perform the GSign algorithm since it is in possession of all $\mathsf{gsk}[j]$. For $j = i$, $\mathcal{B}$ partly also executes GSign, but since sk_i is not known to $\mathcal{B}$, it must use its own signing oracle to obtain the required SFPK signature. In the last step $\mathcal{B}$ must recover the message the SFPK signature in $\mathcal{A}$'s forgery was made for before it outputs that message and signature as it's own forgery, together with the public key from the group signature output by $\mathcal{A}$.

We now need to argue that $\mathcal{B}$ is a PPT adversary and that the full-traceability experiment is simulated correctly for $\mathcal{A}$ in above reduction. With that done, it is possible to analyze $Adv^{\mathsf{euf-sfpk}}_{\Pi_{\mathsf{SFPK}},\mathcal{B}}(\lambda)$. We see that $\mathcal{B}$ obviously is a PPT adversary since

- $\mathcal{A}$ is a PPT adversary
- all algorithms used in the construction of $\mathcal{B}$ are PPT-algorithms by definition (since they are part of either Π_{SFPK}, Π_{SPS} or $gFPK\text{-}GS_{\Pi_{\mathsf{SFPK}},\Pi_{\mathsf{SPS}}}$ (the only exception is BGGen which is a bilinear group generator and therefore a polynomial time algorithm by definition)), namely
  - $\mathsf{KGen}_{\mathsf{SPS}}$

  - $\mathsf{TKGen}_{\mathsf{SFPK}}$
  - $\mathsf{Sign}_{\mathsf{SPS}}$
  - $\mathsf{ChgPK}_{\mathsf{SFPK}}$
  - $\mathsf{ChgSK}_{\mathsf{SFPK}}$
  - $\mathsf{ChgRep}_{\mathsf{SPS}}$
  - GSign

- sending messages and all other basic operations $\mathcal{B}$ performs can obviously be done using PPT-algorithms

If $\mathcal{A}$ queries i during collusion set construction, $\mathcal{B}$ obviously cannot return $\mathsf{gsk}[\mathsf{i}]$. This is because by defining $\mathsf{pk}_i := \mathsf{pk}$, B implicitly set $\mathsf{sk}_i := \mathsf{sk}$, which is the secret key generated by the challenger, a value B cannot access. So the strongest statement with respect to simulation of the full-traceability experiment for $\mathcal{A}$ we can hope to prove is that $\mathcal{B}$ simulates the full-traceability experiment correctly for $\mathcal{A}$ if $\mathcal{A}$ does not choose user i to be part of it's collusion set C. In the advantage analysis for $\mathcal{B}$, it will turn out that this indeed is already sufficient for $\mathcal{B}$ to win $Exp^{\mathsf{euf-sfpk}}_{\Pi_{\mathsf{SFPK}},\mathcal{B}}(\lambda)$ with non-negligible advantage, therefore breaking existential unforgeability of Π_{SFPK}. This follows from the fact that in order to be able to win the full-traceability game according to Def. 3.13 with a forgery that opens to an identity $k \in [n]$, $\mathcal{A}$ cannot have the collusion set $\mathsf{C} = [n]$ which means that $\mathcal{A}$ cannot choose all users to be part of its collusion set C.

Claim If $\mathcal{A}$ does not query i in collusion phase then $\mathcal{B}$ simulates the full-traceability experiment correctly to $\mathcal{A}$.

When proving this claim in the following, we will make use of the terminology for security game simulations established in Def. 5.1. The view of $\mathcal{A}$ in the full-traceability experiment from Def. 3.13 is a tuple with the following structure:

$$\mathcal{V} = (\mathsf{gmsk}, \mathsf{gpk}, \mathsf{C}, (m_d, j_d, \sigma_d)_{d=1}^{q_1}, \mathsf{St}, (m_d, j_d, \sigma_d)_{d=q_1+1}^{q_2})$$

with

- $\mathsf{gmsk}, \mathsf{gpk}$ being a group manager secret key and a group public key generated using GKGen
- $\mathsf{C} \subseteq [n]$ being the collusion set of the adversary $\mathcal{A}$
- St being state information about the course of the game that $\mathcal{A}$ is input at the beginning of the guess-phase
- for every $d \in [q_2]$, m_d is a message, j_d is an identity and σ_d is a *gFPK-GS*-signature for m_d created by user j_d

- q_1 is the number of signing queries $\mathcal{A}$ submitted during the collusion phase, q_2 is the total number of signing queries $\mathcal{A}$ submitted throughout course of the entire experiment with both q_1, q_2 being polynomials in λ

Note that no variable in the view of $\mathcal{A}$ is directly influenced by other variables in $\mathcal{A}$'s view (meaning that if a variable in $\mathcal{A}$'s view is computed from other random variables $\mathcal{A}$ sees, fresh individual randomness is used in the computation), which means that all random variables in $\mathcal{A}$'s view can be considered independent. This means that for showing that $\mathcal{A}$'s view is correctly distributed, it suffices to show that every individual variable it contains is correctly distributed. So in the following we will take a look at the distribution of all random variables from $\mathcal{A}$'s view in the simulation of the full-traceability game in above reduction and argue that all of them are distributed like in the full-traceability experiment from Def. 3.13:

- gmsk is obviously correctly distributed since for $j \neq i$ the j-th entry of gmsk is generated using $\mathsf{TKGen}_{\mathsf{SFPK}}$ by $\mathcal{B}$ and the i-th entry is generated using $\mathsf{TKGen}_{\mathsf{SFPK}}$ by the challenger.

- gpk is correctly distributed since BG is a bilinear group and $\mathsf{pk}_{\mathsf{SPS}}$ is an SPS-EQ public key generated using $\mathsf{KGen}_{\mathsf{SPS}}$ as required.

- For any $j \neq i$: $\mathsf{gsk}[\mathsf{j}]$ is a personal secret key containing an SFPK key pair $(\mathsf{pk}_j, \mathsf{sk}_j) \leftarrow\!\!\$\ \mathsf{TKGen}_{\mathsf{SFPK}}(\lambda, \omega_j)$ and a SPS-EQ-certificate $\sigma^{(j)}_{\mathsf{SPS}} \leftarrow\!\!\$\ \mathsf{Sign}_{\mathsf{SPS}}(\mathsf{sk}_{\mathsf{SPS}}, \mathsf{pk}_j)$ of pk_j as required (note that by the assumption in above claim, $\mathcal{A}$ must never be given access to $\mathsf{gsk}[\mathsf{i}]$). Furthermore we see that the answers $\mathsf{gsk}[\mathsf{j}]$ are part of the same group secret key gsk which was generated by $\mathcal{B}$ during step 1 of the reduction when it simulated the group key generation algorithm GKGen of *gFPK-GS*. Therefore all of $\mathcal{B}$'s answers to $\mathcal{A}$'s collusion queries are correctly distributed.

- Consider both phases where $\mathcal{A}$ submits queries to the signing oracle (which $\mathcal{B}$ needs to simulate for $\mathcal{A}$). We will take a look at how $\mathcal{B}$ creates its answer σ_d to a signing query (m_d, j_d) from $\mathcal{A}$:

  $j_d \neq i$ $\mathcal{B}$ effectively returns $\mathsf{GSign}(\mathsf{gsk}[\mathsf{j_d}], m_d)$ (clear to see, it does all steps part of GSign for this input) as the oracle's answer, as required in the definition of the full-traceability experiment

  $j_d = i$ $\mathcal{B}$ uses its randomize-then-sign oracle to simulate an execution of GSign for parameters $\mathsf{gsk}[\mathsf{i}]$ and m_d. Note that as stated in the first step of above reduction, sk_i is implicitly set to the secret key sk which the signing oracle uses that $\mathcal{B}$ can access.

  So we see that the answer to the query made by $\mathcal{A}$ is correctly distributed in both cases.

- St obviously is correctly distributed since it is computed by $\mathcal{A}$ as required

Since all random variables from $\mathcal{A}$'s view are independent and correctly distributed (and therefore $\mathcal{A}$'s view in the simulation is also correctly distributed), we proved the above claim, so $\mathcal{B}$ simulates the full-traceability game correctly for $\mathcal{A}$ if i is not queried by $\mathcal{A}$ during the collusion phase.

For the later advantage analysis of $\mathcal{B}$, we will make use of the fact that we can very well analyze $Adv^{\mathsf{euf-sfpk}}_{\Pi_{\mathsf{SFPK}},\mathcal{B}}(\lambda)$ in comparison to $Adv^{\mathsf{trace}}_{gFPK\text{-}GS_{\Pi_{\mathsf{SFPK}},\Pi_{\mathsf{SPS}}},\mathcal{A}}(\lambda,n)$ in case that i is the identity $\mathcal{A}$'s forgery opens to. We will formally define this event in the next paragraphs. Consider the following equivalence relation on the set of adversaries against full-traceability of $gFPK\text{-}GS_{\Pi_{\mathsf{SFPK}},\Pi_{\mathsf{SPS}}}$: two adversaries $\mathcal{A},\mathcal{A}'$ are called equivalent if they output message signature pairs $(m_{\mathcal{A}},\sigma_{\mathcal{A}}),(m_{\mathcal{A}'},\sigma_{\mathcal{A}'})$ such that $\mathsf{Open}(\mathsf{gmsk},m_{\mathcal{A}},\sigma_{\mathcal{A}}) = \mathsf{Open}(\mathsf{gmsk},m_{\mathcal{A}'},\sigma_{\mathcal{A}'})$ on input $(\mathsf{gpk},\mathsf{gmsk})$. We see that this obviously induces an equivalence relation and the set of equivalence classes of which is

$$\{\mathcal{A}_k \mid k \in [n]\}$$

where

$$\mathcal{A}_k := \{\mathcal{A} \mid \mathcal{A} \text{ outputs } (m,\sigma) \text{ with } \mathsf{Open}(\mathsf{gpk},m,\sigma) = k\}$$

When the initial input for $\mathcal{A}$ in above reduction algorithm $\mathcal{B}$ has been computed, the unique k such that $\mathcal{A} \in \mathcal{A}_k$ is determined (k exists because of our assumption for this case, uniqueness is clear to see). We define the event good as the event that $i = k$ holds. Since k can be considered fixed at the time i is drawn, we see that

$$\Pr[\mathsf{good}] = \Pr[i = k] = \frac{1}{n}$$

holds.

We are now going to prove that $Adv^{\mathsf{euf-sfpk}}_{\Pi_{\mathsf{SFPK}},\mathcal{B}}(\lambda)$ is not negligible. We see that

$$\begin{aligned}
&Adv^{\mathsf{euf-sfpk}}_{\Pi_{\mathsf{SFPK}},\mathcal{B}}(\lambda)\\
=&\Pr[Exp^{\mathsf{euf-sfpk}}_{\Pi_{\mathsf{SFPK}},\mathcal{B}}(\lambda)=1]\\
=&\Pr[Exp^{\mathsf{euf-sfpk}}_{\Pi_{\mathsf{SFPK}},\mathcal{B}}(\lambda)=1 \mid \mathsf{good}]\cdot\Pr[\mathsf{good}] + \Pr[Exp^{\mathsf{euf-sfpk}}_{\Pi_{\mathsf{SFPK}},\mathcal{B}}(\lambda)=1 \mid \bar{\mathsf{good}}]\cdot\Pr[\bar{\mathsf{good}}]\\
\geq&\Pr[Exp^{\mathsf{euf-sfpk}}_{\Pi_{\mathsf{SFPK}},\mathcal{B}}(\lambda)=1 \mid \mathsf{good}]\cdot\Pr[\mathsf{good}]\\
=&\Pr[Exp^{\mathsf{euf-sfpk}}_{\Pi_{\mathsf{SFPK}},\mathcal{B}}(\lambda)=1 \mid \mathsf{good}]\cdot\frac{1}{n}\\
\geq&\Pr[Exp^{\mathsf{trace}}_{gFPK\text{-}GS_{\Pi_{\mathsf{SFPK}},\Pi_{\mathsf{SPS}}},\mathcal{A}}(\lambda,n)=1]\cdot\frac{1}{n}\\
=&Adv^{\mathsf{trace}}_{gFPK\text{-}GS_{\Pi_{\mathsf{SFPK}},\Pi_{\mathsf{SPS}}},\mathcal{A}}(\lambda,n)\cdot\frac{1}{n}
\end{aligned}$$

Since $Adv^{\mathsf{trace}}_{gFPK\text{-}GS_{\Pi_{\mathsf{SFPK}},\Pi_{\mathsf{SPS}}},\mathcal{A}}(\lambda,n)$ is not negligible in λ and n by assumption in the contraposition of Thm. 5.2 and $\frac{1}{n}$ is not negligible in λ (if it were, this would mean that n is superpolynomial in λ, but then GKGen from Def. 4.1 could not be a PPT algorithm)

we get that $Adv^{\mathsf{trace}}_{gFPK\text{-}GS_{\Pi_{\mathsf{SFPK}},\Pi_{\mathsf{SPS}}},\mathcal{A}}(\lambda,n)\cdot\frac{1}{n}$ is not negligible in λ. Since

$$Adv^{\mathsf{euf-sfpk}}_{\Pi_{\mathsf{SFPK}},\mathcal{B}}(\lambda) \geq Adv^{\mathsf{trace}}_{gFPK\text{-}GS_{\Pi_{\mathsf{SFPK}},\Pi_{\mathsf{SPS}}},\mathcal{A}}(\lambda,n)\cdot\frac{1}{n}$$

this yields that $Adv^{\mathsf{euf-sfpk}}_{\Pi_{\mathsf{SFPK}},\mathcal{B}}(\lambda)$ is not negligible in λ, so Π_{SFPK} is not EUF-CMA secure.

To prove the last inequality, we prove

$$Exp^{\mathsf{trace}}_{gFPK\text{-}GS_{\Pi_{\mathsf{SFPK}},\Pi_{\mathsf{SPS}}},\mathcal{A}}(\lambda,n)=1\wedge\mathsf{good}\Rightarrow Exp^{\mathsf{euf-sfpk}}_{\Pi_{\mathsf{SFPK}},\mathcal{B}}(\lambda)=1 \tag{5.1}$$

with good denoting the event that $i=k$ holds (for the identity i drawn uniformly at random by $\mathcal{B}$ and the identity k that the forgery which $\mathcal{A}$ outputs opens to). This basically means that if $\mathcal{A}$ wins the full-traceability game for $gFPK\text{-}GS_{\Pi_{\mathsf{SFPK}},\Pi_{\mathsf{SPS}}}$ and the good-event occurs, then $\mathcal{B}$ wins the existential unforgeability game for Π_{SFPK}. Let (m^*,σ^*) be the forgery output by $\mathcal{A}$, with $\sigma^*=(\mathsf{pk}^*,\sigma_0^*,\sigma_{\mathsf{SPS}}^*)$ and $\mathsf{Open}(\mathsf{gmsk},m^*,\sigma^*)=k\in[n],k\notin\mathsf{C}$. Note that $k\notin\mathsf{C}$ holds by definition of traceability because of $Exp^{\mathsf{trace}}_{gFPK\text{-}GS_{\Pi_{\mathsf{SFPK}},\Pi_{\mathsf{SPS}}},\mathcal{A}}(\lambda,n)=1$. Furthermore let $M^*:=m^*||\sigma_{\mathsf{SPS}}^*||\mathsf{pk}^*$. We see that

$$\begin{aligned}
&Exp^{\mathsf{trace}}_{gFPK\text{-}GS_{\Pi_{\mathsf{SFPK}},\Pi_{\mathsf{SPS}}},\mathcal{A}}(\lambda,n)=1\wedge\mathsf{good}\\
\Leftrightarrow&\mathsf{GVf}(\mathsf{gpk},m^*,\sigma^*)=1\wedge\mathsf{Open}(\mathsf{gmsk},m^*,\sigma^*)=k\wedge k\notin\mathsf{C}\wedge\mathsf{good}\\
\Leftrightarrow&\mathsf{GVf}(\mathsf{gpk},m^*,\sigma^*)=1\wedge\mathsf{Open}(\mathsf{gmsk},m^*,\sigma^*)=i\wedge i\notin\mathsf{C}\\
\overset{\mathsf{GVf}}{\Rightarrow}&\mathsf{Vf}_{\mathsf{SFPK}}(\mathsf{pk}^*,M^*,\sigma_0^*)=1\wedge\mathsf{Open}(\mathsf{gmsk},m^*,\sigma^*)=i\wedge i\notin\mathsf{C}\\
\Rightarrow&\mathsf{ChkRep}_{\mathsf{SFPK}}(\tau,\mathsf{pk}^*)=1\wedge\mathsf{Vf}_{\mathsf{SFPK}}(\mathsf{pk}^*,M^*,\sigma_0^*)=1\\
\Rightarrow&Exp^{\mathsf{euf-sfpk}}_{\Pi_{\mathsf{SFPK}},\mathcal{B}}(\lambda)=1
\end{aligned}$$

The second last implication holds since we have

$$\begin{aligned}
&\mathsf{Open}(\mathsf{gmsk},m^*,\sigma^*)=i\\
\Rightarrow&\mathsf{pk}^*\in[\mathsf{pk}_i]_{R'}\overset{\mathsf{pk}_i=\mathsf{pk}}{=}[\mathsf{pk}]_{R'}\\
\Rightarrow&\mathsf{ChkRep}_{\mathsf{SFPK}}(\tau,\mathsf{pk}^*)=1
\end{aligned}$$

Note that in the last step in above computation we use that τ is the trapdoor for the class $[\mathsf{pk}]_{R'}$ of pk. The last implication in the proof of Eq. (5.1) holds by definition of the winning condition of existential unforgeability game $Exp^{\mathsf{euf-sfpk}}_{gFPK\text{-}GS_{\Pi_{\mathsf{SFPK}},\Pi_{\mathsf{SPS}}},\mathcal{B}}(\lambda)$ from Def. 3.20, but we need to clarify that $\mathcal{B}$ indeed outputs a forgery for a message that was not signed before (meaning not queried by $\mathcal{B}$ to its signing or randomize-then-sign oracle). We see that

- $(\mathsf{pk}^*,M^*,\sigma_0^*)$ is the triple output by $\mathcal{B}$
- $M^*\notin Q$ (with Q denoting the set of all messages $\mathcal{B}$ queried to its signing and randomize-then-sign oracle) holds since

- if $M^* = m^* || \sigma^*_{\mathsf{SPS}} || \mathsf{pk}^* \in Q$ we get that $\mathcal{A}$ submitted (i, m^*) to $\mathcal{B}$ as a signing oracle query
- this is because $\mathcal{B}$ only queries its SFPK signing oracle once with $m || {\sigma^{(i)}_{\mathsf{SPS}}}' || \mathsf{pk}'_i$ for each query i, m $\mathcal{A}$ submits to its group signing oracle, with pk'_i being a randomized version of the SFPK public key pk_i from the i-th user's personal secret key $\mathsf{gsk}[\mathsf{i}]$ and i being the identity chosen uniformly at random by $\mathcal{B}$ at the beginning of the reduction.
- so if $M^* = m^* || \sigma^*_{\mathsf{SPS}} || \mathsf{pk}^* \in Q$, we get with $\mathsf{pk}^* \in [\mathsf{pk}_i]_{R'}$ that $\mathcal{B}$ queried $(M^*, \cdot)$ to one of its signing oracles to answer the signing query (i, m^*) made by $\mathcal{A}$
- so since m^* is the message $\mathcal{A}$ outputs a forgery for, the fact that $\mathcal{A}$ queried (i, m^*) to its group signing oracle implies $Exp^{\mathsf{trace}}_{gFPK\text{-}GS_{\Pi_{\mathsf{SFPK}},\Pi_{\mathsf{SPS}}},\mathcal{A}}(\lambda, n) = 0$ contradicting the assumption that $Exp^{\mathsf{trace}}_{gFPK\text{-}GS_{\Pi_{\mathsf{SFPK}},\Pi_{\mathsf{SPS}}},\mathcal{A}}(\lambda, n) = 1$ holds

So we constructed a PPT adversary $\mathcal{B}$ which has non-negligible advantage in breaking existential unforgeability of Π_{SFPK} when it has access to a PPT adversary $\mathcal{A}$ that has non-negligible advantage in breaking full-traceability of $gFPK\text{-}GS_{\Pi_{\mathsf{SFPK}},\Pi_{\mathsf{SPS}}}$ and outputs a forgery that opens to an identity $k \in [n]$.

In the following, a PPT adversary $\mathcal{D}$ which has non-negligible advantage in breaking existential unforgeability of Π_{SPS} will be constructed from an adversary $\mathcal{A}$ that has non-negligible advantage in breaking full-traceability of $gFPK\text{-}GS_{\Pi_{\mathsf{SFPK}},\Pi_{\mathsf{SPS}}}$ and outputs an unopenable forgery.

On input (pk) with $BG \leftarrow\!\!\$\ \mathsf{BGGen}_{\mathsf{SPS}}(\lambda)$, $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{KGen}_{\mathsf{SPS}}(BG, l)$ as in the SPS-EQ unforgeability experiment from Def. 3.23, $\mathcal{D}$ behaves as follows:

1. $\mathcal{D}$ sets up the traceability challenge for $\mathcal{A}$, performing the following steps:
   a) $\mathcal{D}$ computes $BG \leftarrow\!\!\$\ \mathsf{BGGen}_{\mathsf{SPS}}(\lambda)$. Note that since the bilinear group generator $\mathsf{BGGen}_{\mathsf{SPS}}$ is a deterministic algorithm (see Def. 3.6), $\mathcal{D}$ can obtain the same group that was used to generate the input to $\mathcal{D}$.
   b) for all identities $j \in [n]$, $\mathcal{D}$ generates the personal secret key and a trapdoor as follows:
      i. $\mathcal{D}$ draws random coins $\omega_j \in \mathsf{coin}$
      ii. $\mathcal{D}$ generates an SFPK key pair with a trapdoor by $(\mathsf{pk}_j, \mathsf{sk}_j, \tau_j) \leftarrow\!\!\$\ \mathsf{TKGen}_{\mathsf{SFPK}}(\lambda, \omega_j)$
      iii. $\mathcal{D}$ queries pk_j to its signing oracle $\mathcal{O}$, obtaining a certificate $\sigma^{(j)}_{\mathsf{SPS}} \leftarrow\!\!\$\ \mathsf{Sign}_{\mathsf{SPS}}(\mathsf{pk}_j, \mathsf{sk})$ for the j-th public key. This results in pk_j being added to the set Q of messages queried to the oracle.
      iv. $\mathcal{D}$ defines the personal secret key of the j-th group member as $\mathsf{gsk}[\mathsf{j}] := (\mathsf{pk}_j, \mathsf{sk}_j, \sigma^{(j)}_{\mathsf{SPS}})$
   c) $\mathcal{D}$ defines the group public key as $\mathsf{gpk} := (BG, \mathsf{pk})$

d) $\mathcal{D}$ defines the group manager secret key as $\mathsf{gmsk} := ((\tau_j, \mathsf{pk}_j))_{j=1}^n$

e) $\mathcal{D}$ defines the group secret key vector as $\mathsf{gsk} := (\mathsf{gsk[j]})_{j=1}^n$

2. $\mathcal{D}$ starts $\mathcal{A}$ on input $(\mathsf{choose}, \mathsf{St})$ with $\mathsf{St} := (\mathsf{gpk}, \mathsf{gmsk})$ and answers $\mathcal{A}$'s queries in the choose-phase as follows:

   collusion query : if $\mathcal{A}$ sends an identity $j \in [n]$, $\mathcal{D}$ adds j to the collusion set C by $\mathsf{C} := \mathsf{C} \cup \{j\}$ and returns $\mathsf{gsk[j]}$

   signing query : if $\mathcal{A}$ sends an identity j and a message m, $\mathcal{D}$ computes and returns $\sigma \leftarrow_\$ \mathsf{GSign}(\mathsf{gsk[j]}, m)$.

   Eventually $\mathcal{A}$ halts and outputs state information St.

3. $\mathcal{D}$ starts $\mathcal{A}$ on input $(\mathsf{guess}, \mathsf{St})$ and answers $\mathcal{A}$'s signing queries just like in the choose-phase

4. when $\mathcal{A}$ eventually outputs a forgery (m^*, σ^*), $\mathcal{D}$ will create his candidate forgery for Π_{SFPK} as follows:

   a) $\mathcal{D}$ parses $\sigma^* := (\mathsf{pk}^*, \sigma_0^*, \sigma_{\mathsf{SPS}}^*)$

   b) $\mathcal{D}$ outputs $(\mathsf{pk}^*, \sigma_{\mathsf{SPS}}^*)$

The values choose and guess are constants indicating the current phase of the full-traceability game. In the first step, $\mathcal{D}$ basically performs GKGen with the only difference that the certificates for the public keys in the users' personal secret keys are computed using the signing oracle. Answering the collusion and signing queries throughout the experiment is straightforward since $\mathcal{D}$ has all information needed and does not have to query any oracle. Eventually $\mathcal{D}$ outputs the randomized public key and adapted certificate from $\mathcal{A}$'s forgery as his own forgery.

We see that $\mathcal{D}$ obviously is a PPT adversary since

- $\mathcal{A}$ is a PPT adversary
- all algorithms used in the construction of $\mathcal{D}$ are PPT-algorithms by definition (since they are part of either Π_{SFPK} or $gFPK\text{-}GS_{\Pi_{\mathsf{SFPK}},\Pi_{\mathsf{SPS}}}$), namely
  - $\mathsf{BGGen}_{\mathsf{SPS}}$
  - $\mathsf{TKGen}_{\mathsf{SFPK}}$
  - GSign
- sending messages and all other basic operations $\mathcal{D}$ performs can obviously be done using PPT algorithms

When proving that $\mathcal{D}$ simulates the full-traceability experiment correctly to $\mathcal{A}$, we will again make use of the terminology introduced in Def. 5.1. The view of $\mathcal{A}$ is a tuple

$$\mathcal{V} = (\mathsf{gmsk}, \mathsf{gpk}, \mathsf{C}, (m_d, j_d, \sigma_d)_{d=1}^{q_1}, \mathsf{St}, (m_d, j_d, \sigma_d)_{d=q_1+1}^{q_2})$$

with

- $\mathsf{gmsk}, \mathsf{gpk}$ being a group manager secret key and a group public key generated using GKGen
- $\mathsf{C} \subseteq [n]$ being the collusion set of the adversary $\mathcal{A}$
- St being state information about the course of the game that $\mathcal{A}$ is input at the beginning of the guess-phase
- for every $d \in [q_2]$, m_d is a message, j_d is an identity and σ_d is a *gFPK-GS*-signature of m_d created by user j_d
- q_1 is the number of signing queries $\mathcal{A}$ submitted during the collusion phase, q_2 is the total number of signing queries $\mathcal{A}$ submitted throughout course of the entire experiment with both q_1, q_2 being polynomials in λ

For proving that the simulation of the full-traceability game is correct, we will take a look at all random variables that are contained in $\mathcal{A}$'s view and argue that all of them are distributed like in the full-traceability experiment from Def. 3.13. Looking at the individual random variables again is sufficient (like in the first reduction in this proof) as they can be considered independent since fresh randomness is used when computing a random variable in $\mathcal{A}$'s view from others. We see that

- gmsk is obviously correctly distributed since all public keys and trapdoors it contains are generated using $\mathsf{TKGen}_{\mathsf{SFPK}}$ as required
- gpk obviously is correctly distributed since pk is an SPS-EQ public key and BG is a bilinear group
- gsk is correctly distributed since all $\mathsf{pk}_j, \mathsf{sk}_j$ are created using $\mathsf{TKGen}_{\mathsf{SFPK}}$ and all $\sigma^{(j)}_{\mathsf{SPS}}$ are created using the signing oracle which uses the secret key sk corresponding to the public key pk from gpk
- St is obviously correctly distributed since it is computed by $\mathcal{A}$ as required
- $\mathcal{D}$'s responses to $\mathcal{A}$'s signing oracle queries (j_d, m_d) are obviously correctly distributed since they are created using GSign and the respective personal secret key $\mathsf{gsk}[\mathsf{j_d}]$ just as required in the full-traceability experiment (Def. 3.13)
- since gsk is correctly distributed, $\mathcal{D}$'s answers to $\mathcal{A}$'s queries in the collusion phase are also correctly distributed

Since all random variables $\mathcal{A}$ sees are correctly distributed, we proved above claim, so $\mathcal{D}$ simulates the full-traceability game correctly for $\mathcal{A}$.

To analyze $\mathcal{D}$'s advantage in breaking existential unforgeability of Π_{SPS}, we prove the following implication:

$$Exp^{\mathsf{trace}}_{gFPK\text{-}GS_{\Pi_{\mathsf{SFPK}},\Pi_{\mathsf{SPS}}},\mathcal{A}}(\lambda, n) = 1 \Rightarrow Exp^{\mathsf{euf-sps}}_{\Pi_{\mathsf{SPS}},\mathcal{D}}(\lambda) = 1 \tag{5.2}$$

which basically means that if $\mathcal{A}$ wins the full-traceability game for $gFPK\text{-}GS_{\Pi_{\mathsf{SFPK}},\Pi_{\mathsf{SPS}}}$ then $\mathcal{D}$ wins the unforgeability game for Π_{SPS}. Let (m^*, σ^*) be the forgery output by $\mathcal{A}$, with $\sigma^* = (\mathsf{pk}^*, \sigma_0^*, \sigma_{\mathsf{SPS}}^*)$ and $\mathsf{Open}(\mathsf{gmsk}, m^*, \sigma^*) = \perp \notin [n]$. We see that

$$\begin{aligned}
&Exp^{\mathsf{trace}}_{gFPK\text{-}GS_{\Pi_{\mathsf{SFPK}},\Pi_{\mathsf{SPS}}},\mathcal{A}}(\lambda, n) = 1 \\
\Rightarrow &\mathsf{GVf}(\mathsf{gpk}, m^*, \sigma^*) = 1 \wedge \mathsf{Open}(\mathsf{gmsk}, m^*, \sigma^*) = \perp \\
\Rightarrow &\mathsf{Vfy}_{\mathsf{SPS}}(\mathsf{pk}^*, \sigma_{\mathsf{SPS}}^*, \mathsf{pk}) = 1 \wedge \nexists i \in [n] : \mathsf{ChkRep}_{\mathsf{SFPK}}(\tau_i, \mathsf{pk}^*) = 1 \\
\Rightarrow &\mathsf{Vfy}_{\mathsf{SPS}}(\mathsf{pk}^*, \sigma_{\mathsf{SPS}}^*, \mathsf{pk}) = 1 \wedge \forall \tilde{\mathsf{pk}} \in Q : \mathsf{pk}^* \notin [\tilde{\mathsf{pk}}]_R \\
\Rightarrow &Exp^{\mathsf{euf-sps}}_{\Pi_{\mathsf{SPS}},\mathcal{D}}(\lambda) = 1
\end{aligned}$$

with Q denoting the set of all messages $\mathcal{D}$ queried to its SPS-EQ signing oracle over the course of the game. So with Eq. (5.2) we get

$$\begin{aligned}
&Adv^{\mathsf{euf-sps}}_{\Pi_{\mathsf{SPS}},\mathcal{D}}(\lambda) \\
=&\Pr[Exp^{\mathsf{euf-sps}}_{\Pi_{\mathsf{SPS}},\mathcal{D}}(\lambda) = 1] \\
\geq&\Pr[Exp^{\mathsf{trace}}_{gFPK\text{-}GS_{\Pi_{\mathsf{SFPK}},\Pi_{\mathsf{SPS}}},\mathcal{A}}(\lambda, n) = 1] \\
=&Adv^{\mathsf{trace}}_{gFPK\text{-}GS_{\Pi_{\mathsf{SFPK}},\Pi_{\mathsf{SPS}}},\mathcal{A}}(\lambda, n)
\end{aligned}$$

which yields that since $Adv^{\mathsf{trace}}_{gFPK\text{-}GS_{\Pi_{\mathsf{SFPK}},\Pi_{\mathsf{SPS}}},\mathcal{A}}(\lambda, n)$ is not negligible, $Adv^{\mathsf{euf-sps}}_{\Pi_{\mathsf{SPS}},\mathcal{D}}(\lambda)$ is also not negligible, so Π_{SPS} is not EUF-CMA secure.

The third last implication in the proof of Eq. (5.2) holds since (m^*, σ^*) is valid and unopenable, so the certificate for the public key pk^* in σ^* must be valid under the SPS-EQ public key pk; furthermore no personal secret key $\mathsf{gsk}[i]$ which contains a public key related to pk^* exists. The second last implication holds since the public keys pk_j from the personal secret keys of the group members are the only messages $\mathcal{D}$ ever submitted to its signing oracle which yields

$$\forall \tilde{\mathsf{pk}} \in Q : \mathsf{pk}^* \notin [\tilde{\mathsf{pk}}]_R$$

So we constructed a PPT adversary $\mathcal{D}$ which has non-negligible advantage in breaking existential unforgeability of Π_{SPS} when it has access to a PPT adversary $\mathcal{A}$ that has non-negligible advantage in breaking full-traceability of $gFPK\text{-}GS_{\Pi_{\mathsf{SFPK}},\Pi_{\mathsf{SPS}}}$ and outputs an unopenable forgery.

All in all, given a PPT adversary $\mathcal{A}$ against full-traceability of $gFPK\text{-}GS_{\Pi_{\mathsf{SFPK}},\Pi_{\mathsf{SPS}}}$ which has non negligible advantage $Adv^{\mathsf{trace}}_{gFPK\text{-}GS_{\Pi_{\mathsf{SFPK}},\Pi_{\mathsf{SPS}}},\mathcal{A}}(\lambda, n)$, we constructed two PPT adversaries $\mathcal{B}, \mathcal{D}$ such that either $Adv^{\mathsf{euf-sfpk}}_{\Pi_{\mathsf{SFPK}},\mathcal{B}}(\lambda)$ or $Adv^{\mathsf{euf-sps}}_{\Pi_{\mathsf{SPS}},\mathcal{D}}(\lambda)$ is not negligible which means that we proved Thm. 5.2 by contraposition. □

5.2 Full-anonymity

In this section will give proof of full-anonymity of *gFPK-GS* under the prerequisites named by Backes et al. [3]. The following theorem can be found as Theorem 4 in [3]. In the following we will first adjust it to the notation used in this thesis and then give a short sketch of the general idea behind the proof.

Theorem 5.3 *Let R, R' be compatible equivalence relations (Def. 4.2), $\lambda \in \mathbb{N}$ security parameter, $n \in \mathbb{N}$ group size, $l \in \mathbb{N}, l > 1$. Let*
$\Pi_{\mathsf{SPS}} = (\mathsf{BGGen}_{\mathsf{SPS}}, \mathsf{KGen}_{\mathsf{SPS}}, \mathsf{Sign}_{\mathsf{SPS}}, \mathsf{ChgRep}_{\mathsf{SPS}}, \mathsf{Vfy}_{\mathsf{SPS}}, \mathsf{VKey}_{\mathsf{SPS}})$ *be an SPS-EQ scheme with relation R and vector length l,*
$\Pi_{\mathsf{SFPK}} = (\mathsf{KGen}_{\mathsf{SFPK}}, \mathsf{TKGen}_{\mathsf{SFPK}}, \mathsf{Sign}_{\mathsf{SFPK}}, \mathsf{ChkRep}_{\mathsf{SFPK}}, \mathsf{ChgPK}_{\mathsf{SFPK}}, \mathsf{ChgSK}_{\mathsf{SFPK}}, \mathsf{Vf}_{\mathsf{SFPK}})$ *be an SFPK scheme with relation R' over the key space. Furthermore let Π_{SFPK} be strongly existentially unforgeable (Def. 3.19) and class-hiding (Def. 3.17), let Π_{SPS} perfectly adapt signatures (Def. 3.24) and be existentially unforgeable (Def. 3.23). Then* $\text{gFPK-GS}_{\Pi_{\mathsf{SFPK}},\Pi_{\mathsf{SPS}}}$ *is fully-anonymous.*

Proof. We will use an approach based on a sequence of games $\mathsf{G}_0, \ldots, \mathsf{G}_5$ where G_0 is the original full-anonymity experiment from Def. 3.10. To prove full-anonymity of $gFPK\text{-}GS_{\Pi_{\mathsf{SFPK}},\Pi_{\mathsf{SPS}}}$ according to Def. 3.10 we will consider an arbitrary PPT adversary $\mathcal{A}$ and use S_i to denote the event that $\mathcal{A}$ wins in G_i, meaning that its output $\tilde{b}$ is equal to the hidden bit b. It is sufficient to upper-bound $\Pr[\mathsf{S}_0]$ to $\frac{1}{2} + \kappa(\lambda)$ for a negligible function κ, for that we will use and analyze the game sequence $\mathsf{G}_0, \ldots, \mathsf{G}_5$. As a first step, we will find an upper bound for $\Pr[\mathsf{S}_0]$ which is dependent on $\Pr[\mathsf{S}_5]$. The idea behind this is that the sequence of games is constructed in a way that G_5 is independent of the hidden bit b, which makes it possible to precisely upper-bound $\Pr[\mathsf{S}_5]$.

Sequence of games In the following we will define the sequence of games used in this proof. A formal description of all of its games can be found in Appendix A, detailed pseudocode for G_5 with all changes highlighted will also be given later in this proof. To make it easier to understand the changes made in the games when reading through below description, the pseudocode of G_0, which is the full-anonymity bit guessing game from Def. 3.11 for *gFPK-GS*, will be given before describing the game sequence.

$\mathsf{G}_0(\lambda)$

1 : $b \leftarrow\!\!\$\, \{0,1\}$
2 : $BG := \mathsf{BGGen}_{\mathsf{SPS}}(\lambda)$
3 : $(\mathsf{pk}_{\mathsf{SPS}}, \mathsf{sk}_{\mathsf{SPS}}) \leftarrow\!\!\$\, \mathsf{KGen}_{\mathsf{SPS}}(BG, l)$
4 : **for** $j \in [n]$
5 : $\quad \omega_j \leftarrow\!\!\$\, \mathsf{coin}$
6 : $\quad (\mathsf{pk}_j, \mathsf{sk}_j, \tau_j) \leftarrow\!\!\$\, \mathsf{TKGen}_{\mathsf{SFPK}}(\lambda, \omega_j)$
7 : $\quad \sigma^{(j)}_{\mathsf{SPS}} \leftarrow\!\!\$\, \mathsf{Sign}_{\mathsf{SPS}}(\mathsf{sk}_{\mathsf{SPS}}, \mathsf{pk}_j)$
8 : $\mathsf{gpk} := (BG, \mathsf{pk}_{\mathsf{SPS}})$
9 : $\mathsf{gmsk} := ((\mathsf{pk}_j, \tau_j))_{j=1}^n$
10 : **for** $j \in [n]$
11 : $\quad \mathsf{gsk[j]} := (\mathsf{pk}_j, \mathsf{sk}_j, \sigma^{(j)}_{\mathsf{SPS}})$
12 : $\mathsf{gsk} := (\mathsf{gsk[j]})_{j=1}^n$
13 : $(\mathsf{St}, i_0, i_1, m) \leftarrow\!\!\$\, \mathcal{A}^{\mathsf{Open}(\mathsf{gmsk},\cdot,\cdot)}(\mathsf{choose}, \mathsf{gpk}, \mathsf{gsk})$
14 : parse $\mathsf{gsk[i_b]} := (\mathsf{pk}_{i_b}, \mathsf{sk}_{i_b}, \sigma^{(i_b)}_{\mathsf{SPS}})$
15 : $r \leftarrow\!\!\$\, \mathsf{coin}$
16 : $\mathsf{pk}'_{i_b} \leftarrow\!\!\$\, \mathsf{ChgPK}_{\mathsf{SFPK}}(\mathsf{pk}_{i_b}, r)$
17 : $\mathsf{sk}'_{i_b} \leftarrow\!\!\$\, \mathsf{ChgPK}_{\mathsf{SFPK}}(\mathsf{sk}_{i_b}, r)$
18 : $(\mathsf{pk}'_{i_b}, {\sigma^{(i_b)}_{\mathsf{SPS}}}') \leftarrow\!\!\$\, \mathsf{ChgRep}_{\mathsf{SPS}}(\mathsf{pk}_{i_b}, \sigma^{(i_b)}_{\mathsf{SPS}}, r, \mathsf{pk}_{\mathsf{SPS}})$
19 : $M := m || {\sigma^{(i_b)}_{\mathsf{SPS}}}' || \mathsf{pk}'_{i_b}$
20 : $\sigma_0 \leftarrow\!\!\$\, \mathsf{Sign}_{\mathsf{SFPK}}(\mathsf{sk}'_{i_b}, M)$
21 : $\sigma_{ch} := (\mathsf{pk}'_{i_b}, \sigma_0, {\sigma^{(i_b)}_{\mathsf{SPS}}}')$
22 : $\tilde{b} \leftarrow\!\!\$\, \mathcal{A}^{\mathsf{Open}(\mathsf{gmsk},\cdot,\cdot)}(\mathsf{guess}, \mathsf{St}, \sigma_{ch})$
23 : **if** $\mathcal{A}$ did not query Open-oracle with (m, σ_{ch}) in guess phase
24 : $\quad$ **then** return $\tilde{b} = b$
25 : **else** return 0

Next we will informally define the rest of the game sequence (for a formal definition with all changes marked in the pseudocode see Appendix A).

G_0**:** regular full-anonymity game (bit-guessing variant from Def. 3.11)

G_1**:** G_0 but when creating the challenge signature σ_{ch}, the contained public key $\mathsf{pk}' \leftarrow\!\!\$\, \mathsf{ChgPK}_{\mathsf{SFPK}}(\mathsf{pk}, r)$ is manually signed using $\mathsf{Sign}_{\mathsf{SPS}}$ to create the required certificate σ'_{SPS} instead of using $\mathsf{ChgRep}_{\mathsf{SPS}}$

G_2**:** G_1 but an identity $i \in [n]$ is chosen uniformly at random and the game is aborted if $i \neq i_b$

G_3**:** G_2 but the game is aborted if $\mathcal{A}$ submits an unopenable query (means Open returns $\perp$ for that pair) to the opening oracle

G_4: G_3 but when generating the personal secret key $\mathsf{gsk}[i]$ for user i, $\mathsf{KGen}_{\mathsf{SFPK}}$ is used instead of $\mathsf{TKGen}_{\mathsf{SFPK}}$ to generate the required SFPK key pair $(\mathsf{pk}_i, \mathsf{sk}_i)$. Furthermore, the Open-algorithm of $gFPK\text{-}GS_{\Pi_{\mathsf{SFPK}},\Pi_{\mathsf{SPS}}}$ is adapted as seen below.

G_5: G_4 but the public key pk' (and therefore also certificate σ'_{SPS}) in the challenge signature σ_{ch} is replaced by a random one.

Aborting in the case that $i \neq i_b$ from G_2 onwards is not necessary and only makes the proof more convenient. This is because from G_3 onwards, the identity which must be used in the creation of the challenge signature is now known before $\mathcal{A}$ submits two identities and a message. So we can make changes on the personal secret key generation for a identity i without having to analyze the cases that $\mathcal{A}$'s choice of i_0, i_1 forces us to create the challenge signature for a different identity than i. The change between G_3 and G_4 is needed to finally be able to do the last game hop which makes the game independent of the hidden bit b. We will point out where we make use of this change when we later reduce distinguishing G_4 and G_5 to breaking class-hiding of Π_{SFPK}. Because of the change in G_4, no trapdoor is generated for the SFPK public key pk_i in the personal secret key $\mathsf{gsk}[i]$ of user i. Therefore, opening of signatures must be done differently from G_4 onwards since Open from $gFPK\text{-}GS_{\Pi_{\mathsf{SFPK}},\Pi_{\mathsf{SPS}}}$ requires trapdoors for all the SFPK in the users personal secret keys to work correctly. To open a signature queried to the opening oracle in G_4 and G_5, we will make use of the fact that every queried signature must open to an identity $k \in [n]$ and cannot be unopenable (due to the change in G_3). So if the public key pk' in a specific group signature $(\mathsf{pk}', \cdot, \cdot)$ cannot be found related to an SFPK public key in any users personal secret key using the $n-1$ trapdoors in gmsk, this signature must therefore open to the identity i for which the personal secret key was generated differently. Formally we can define a modified Open-algorithm Open' as follows and prove that Open' will return the same result as the original Open for every input triple $(\mathsf{gmsk}, m, \sigma)$.

$\mathsf{Open}'(\mathsf{gmsk}, m, \sigma_{\mathsf{GS}})$

1 : parse $\sigma_{\mathsf{GS}} = (\mathsf{pk}', \sigma_0, \sigma'_{\mathsf{SPS}})$
2 : parse $\mathsf{gmsk} = ((\mathsf{pk}_j, \tau_j))_{j=1}^n$
3 : **if** $\mathsf{GVf}(\mathsf{gpk}, m, \sigma_{\mathsf{GS}}) = 0$
4 : **return** 0
5 : **if** $\nexists j \in [n] \setminus \{i\} : \mathsf{ChkRep}_{\mathsf{SFPK}}(\tau_j, \mathsf{pk}') = 1$
6 : **return** i
7 : **else**
8 : **return** j

We see that for every $j \in [n] \setminus \{i\}$, we get

$$\mathsf{Open}(\mathsf{gmsk}, m, \sigma_{\mathsf{GS}}) = \mathsf{Open}'(\mathsf{gmsk}, m, \sigma_{\mathsf{GS}})$$

trivially for every group manager secret key gmsk generated using GKGen and message-signature pair $(m, \sigma_{\mathsf{GS}})$. If $\mathsf{Open}(\mathsf{gmsk}, m, \sigma_{\mathsf{GS}}) = i$ holds (with $\sigma_{\mathsf{GS}} = (\mathsf{pk}', \sigma_0, \sigma'_{\mathsf{SPS}})$), we see that by construction of Open

$$\begin{aligned}
&\mathsf{Open}(\mathsf{gmsk}, m, \sigma_{\mathsf{GS}}) = i \\
\Rightarrow &\mathsf{ChkRep}_{\mathsf{SFPK}}(\tau_i, \mathsf{pk}') = 1 \\
\Rightarrow &\nexists j \in [n] \setminus \{i\} : \mathsf{ChkRep}_{\mathsf{SFPK}}(\tau_j, \mathsf{pk}') = 1 \\
\Rightarrow &\mathsf{Open}'(\mathsf{gmsk}, m, \sigma_{\mathsf{GS}}) = i
\end{aligned}$$

holds, the third implication comes from the fact that for $gFPK\text{-}GS_{\Pi_{\mathsf{SFPK}},\Pi_{\mathsf{SPS}}}$ to be correct, all SFPK public keys contained in the users personal secret keys must be pairwise unrelated (see Lem. 4.3). So since Open and Open' always return the same result when called with the same input, they can be considered equal as functions. We will therefore always write Open instead of Open' for simplicity.

Two things are important to note about the change in G_5: First, random public key does not necessarily mean uniformly random but that we can choose the distribution of pk'. So this basically the freedom of choosing any distribution for pk' that is convenient in a certain context. The change in G_1 is a crucial requirement to do so, since it allows us to create a fresh SPS-EQ certificate for pk' (which is needed since pk' is not related to any key generated during GKGen, therefore we cannot obtain a SPS-EQ certificate for pk' using $\mathsf{ChgRep}_{\mathsf{SPS}}$). Second, the challenge signature in G_5 is independent of the bit b drawn at the beginning of the experiment. We will make use of this important fact later in the proof. Despite all games from above game sequence can be found in Appendix A, we will give a formal definition of G_5 here for convenience and mark all changes with the number of the game they first occur in.

$\mathsf{G}_5(\lambda)$

1 : $b \leftarrow\!\!\$\ \{0,1\}$
2 : $i \leftarrow\!\!\$\ [n]$ // G_2-change
3 : $BG := \mathsf{BGGen}_{\mathsf{SPS}}(\lambda)$
4 : $(\mathsf{pk}_{\mathsf{SPS}}, \mathsf{sk}_{\mathsf{SPS}}) \leftarrow\!\!\$\ \mathsf{KGen}_{\mathsf{SPS}}(BG, l)$
5 : **for** $j \in [n]$
6 : $\quad \omega_j \leftarrow\!\!\$\ \mathsf{coin}$
7 : $\quad$ **if** $j \neq i$ // G_4-change
8 : $\quad\quad (\mathsf{pk}_j, \mathsf{sk}_j, \tau_j) \leftarrow\!\!\$\ \mathsf{TKGen}_{\mathsf{SFPK}}(\lambda, \omega_j)$
9 : $\quad$ **else** // G_4-change
10 : $\quad\quad (\mathsf{pk}_j, \mathsf{sk}_j) \leftarrow\!\!\$\ \mathsf{TKGen}_{\mathsf{SFPK}}(\lambda, \omega_j)$ // G_4-change
11 : $\quad \sigma^{(j)}_{\mathsf{SPS}} \leftarrow\!\!\$\ \mathsf{Sign}_{\mathsf{SPS}}(\mathsf{sk}_{\mathsf{SPS}}, \mathsf{pk}_j)$
12 : $\mathsf{gpk} := (BG, \mathsf{pk}_{\mathsf{SPS}})$
13 : $\mathsf{gmsk} := ((\mathsf{pk}_j, \tau_j))_{j=1}^n$
14 : **for** $j \in [n]$
15 : $\quad \mathsf{gsk[j]} := (\mathsf{pk}_j, \mathsf{sk}_j, \sigma^{(j)}_{\mathsf{SPS}})$
16 : $\mathsf{gsk} := (\mathsf{gsk[j]})_{j=1}^n$
17 : $(\mathsf{St}, i_0, i_1, m) \leftarrow\!\!\$\ \mathcal{A}^{\mathsf{Open}(\mathsf{gmsk},\cdot,\cdot)}(\mathsf{choose}, \mathsf{gpk}, \mathsf{gsk})$
18 : **if** $i \neq i_b$ // G_2-change
19 : $\quad$ abort // G_2-change
20 : $\omega \leftarrow\!\!\$\ \mathsf{coin}$ // G_5-change
21 : $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{KGen}_{\mathsf{SFPK}}(\lambda, \omega)$ // G_5-change
22 : $r \leftarrow\!\!\$\ \mathsf{coin}$
23 : $\mathsf{pk}' \leftarrow\!\!\$\ \mathsf{ChgPK}_{\mathsf{SFPK}}(\mathsf{pk}, r)$ // G_5-change
24 : $\mathsf{sk}' \leftarrow\!\!\$\ \mathsf{ChgPK}_{\mathsf{SFPK}}(\mathsf{sk}, r)$ // G_5-change
25 : $\sigma'_{\mathsf{SPS}} \leftarrow\!\!\$\ \mathsf{Sign}_{\mathsf{SPS}}(\mathsf{pk}', \mathsf{sk}_{\mathsf{SPS}})$ // G_1-change, G_5-change
26 : $M := m||\sigma'_{\mathsf{SPS}}||\mathsf{pk}'$ // G_5-change
27 : $\sigma_0 \leftarrow\!\!\$\ \mathsf{Sign}_{\mathsf{SFPK}}(\mathsf{sk}', M)$
28 : $\sigma_{ch} := (\mathsf{pk}', \sigma_0, \sigma'_{\mathsf{SPS}})$ // G_5-change
29 : $\tilde{b} \leftarrow\!\!\$\ \mathcal{A}^{\mathsf{Open}(\mathsf{gmsk},\cdot,\cdot)}(\mathsf{guess}, \mathsf{St}, \sigma_{ch})$
30 : **if** $\mathcal{A}$ queried (m^*, σ^*) with $\mathsf{Open}(\mathsf{gmsk}, m^*, \sigma^*) = \perp$ // G_3-change
31 : $\quad$ abort // G_3-change
32 : **if** $\mathcal{A}$ did not query Open-oracle with (m, σ_{ch}) in guess phase
33 : $\quad$ **then** return $\tilde{b} = b$
34 : **else** return 0

Next we establish a relation between $\Pr[\mathsf{S}_j], \Pr[\mathsf{S}_{j+1}]$ for $j \in \{0, \ldots, 4\}$. S_j denotes the event that $\mathcal{A}$ wins in G_j, which means that $\tilde{b} = b$ holds, so the adversary guesses the hidden bit correctly.

From game 0 to game 1 In G_1, the way to compute SPS-EQ in the challenge signature is changed. We see that all random variables computed throughout the course of $\mathsf{G}_0, \mathsf{G}_1$ are computed the same way in both $\mathsf{G}_0, \mathsf{G}_1$, except for the certificate σ'_{SPS} for the randomized public key in the challenge signature (see pseudocode of the two games in Appendix A). It is computed as $(\cdot, c_0) \leftarrow_{\$} \mathsf{ChgRep}_{\mathsf{SPS}}(\mathsf{pk}_{i_b}, \sigma^{(i_b)}_{\mathsf{SPS}}, r, \mathsf{pk}_{\mathsf{SPS}})$ in G_0 and as $c_1 \leftarrow_{\$} \mathsf{Sign}_{\mathsf{SPS}}(\mathsf{pk}'_{i_b}, \mathsf{sk}_{\mathsf{SPS}})$ in G_1. However, since Π_{SPS} perfectly adapts signatures we have that c_0, c_1 are identically distributed which proves our claim. So we see that the output $\tilde{b}$ of $\mathcal{A}$ is therefore also identically distributed in $\mathsf{G}_0, \mathsf{G}_1$, which yields

$$\Pr[\mathsf{S}_0] = \Pr[\mathsf{S}_1]$$

From game 1 to game 2 We will continue with $\mathsf{G}_1, \mathsf{G}_2$, so we need to relate $\Pr[\mathsf{S}_1]$ and $\Pr[\mathsf{S}_2]$. In G_2, an identity i is drawn uniformly at random and the game is aborted if i does not match the challenge identity i_b. We see that all random variables needed for determining i_b are independent of i, so i_b also is independent of i. Therefore we get

$$\Pr[i = i_b] = \frac{1}{n}$$

Furthermore it is important to note that the events S_1 that $\mathcal{A}$ wins G_1 and the event $i = i_b$ that the challenge identity is guessed correctly are obviously independent. With this we get

$$\begin{aligned}\Pr[\mathsf{S}_2] &= \Pr[\mathsf{S}_1 \wedge i = i_b] \\ &= \Pr[\mathsf{S}_1] \cdot \Pr[i = i_b] \\ &= \Pr[\mathsf{S}_1] \cdot \frac{1}{n}\end{aligned}$$

From game 2 to game 3 In G_3, we abort if $\mathcal{A}$ submits a message-signature pair to the opening oracle for which Open returns $\perp$. Let F denote the event that $\mathcal{A}$ submits such an unopenable oracle query, this means that over the course of G_2 or G_3, $\mathcal{A}$ submits $(\tilde{m}, \tilde{\sigma})$ such that $\mathsf{Open}(\mathsf{gmsk}, \tilde{m}, \tilde{\sigma}) = \perp$. Let $\tilde{\sigma} = (\tilde{\mathsf{pk}}, \tilde{\sigma_0}, \sigma_{\tilde{\mathsf{SPS}}})$. If $(\tilde{m}, \tilde{\sigma})$ is unopenable, this means that

$$\forall j \in [n] : \tilde{\mathsf{pk}} \notin [\mathsf{pk}_j]_{R'}$$

where pk_j is the SFPK public key contained in the personal secret key of user j. Since these pk_j are the only messages that $\mathcal{A}$ has observed SPS-EQ signatures for and $\sigma_{\tilde{\mathsf{SPS}}}$ is a valid SPS-EQ for $\tilde{\mathsf{pk}}$, $(\tilde{\mathsf{pk}}, \sigma_{\tilde{\mathsf{SPS}}})$ is a valid forgery for the SPS-EQ key pair $(\mathsf{pk}_{\mathsf{SPS}}, \mathsf{sk}_{\mathsf{SPS}})$. Thus $\mathcal{A}$ can be seen as an adversary breaking existential unforgeability of Π_{SPS} since

- the tuple $(\tilde{\mathsf{pk}}, \sigma_{\tilde{\mathsf{SPS}}})$ allows $\mathcal{A}$ to win the existential unforgeability game for Π_{SPS} for the SPS-EQ key pair $(\mathsf{pk}_{\mathsf{SPS}}, \mathsf{sk}_{\mathsf{SPS}})$ that is distributed like in the unforgeability game for SPS-EQ from Def. 3.23
- $(\tilde{\mathsf{pk}}, \sigma_{\tilde{\mathsf{SPS}}})$ is easy to construct from the unopenable query $(\tilde{m}, \tilde{\sigma} = (\tilde{\mathsf{pk}}, \tilde{\sigma_0}, \sigma_{\tilde{\mathsf{SPS}}}))$ which $\mathcal{A}$ outputs throughout the course of the game

Furthermore F can be seen as the event that $\mathcal{A}$ wins the existential unforgeability game for Π_{SPS} and challenge $(\mathsf{pk}_{\mathsf{SPS}})$. Together with the prerequisited existential unforgeability of Π_{SPS} and the Difference Lemma (Lem. 5.4) we get

$$\begin{aligned} |\Pr[\mathsf{S}_2] - \Pr[\mathsf{S}_3]| &\leq \Pr[\mathsf{F}] \\ &= Adv^{\mathsf{euf-sps}}_{\Pi_{\mathsf{SPS}},\mathcal{A}}(\lambda) \\ &\leq \kappa(\lambda) \end{aligned}$$

for a negligible function κ. The first inequality comes directly from the Difference Lemma since

$$\mathsf{S}_2 \wedge \overline{\mathsf{F}} \Leftrightarrow \mathsf{S}_3 \wedge \overline{\mathsf{F}}$$

is plain to see (G_2 and G_3 proceed identically if F does not occur so all random variables computed throughout G_2, G_3 are computed identically in both of the games, this means that the distribution of $\mathcal{A}$'s output and $\mathcal{A}$'s winning probability must also be the same in both games). The second equality holds since $\mathcal{A}$ wins the unforgeability game for Π_{SPS} if and only if F occurs. The last inequality holds since Π_{SPS} in existentially unforgeable.

From game 3 to game 4 Next we analyze G_3 and G_4 and prove that they are perfectly indistinguishable. So we need to prove that all random variables computed throughout the course of the two games are identically distributed in both G_3 and G_4. G_4 changes the way the SFPK public key in the personal secret key $\mathsf{gsk}[\mathrm{i}]$ for the identity i drawn uniformly at random is computed. We easily see that all random variables computed throughout the course of the two games are computed the same way except for the SFPK key pair in the personal secret key of user i (see pseudocode of the two games in Appendix A), which is computed as

$$(\mathsf{pk}_3, \mathsf{sk}_3, \cdot) \leftarrow_{\$} \mathsf{TKGen}_{\mathsf{SFPK}}(\lambda, \omega_i)$$

in G_3 and as

$$(\mathsf{pk}_4, \mathsf{sk}_4) \leftarrow_{\$} \mathsf{KGen}_{\mathsf{SFPK}}(\lambda, \omega_i)$$

in G_4. Correctness of Π_{SFPK} (Def. 3.15) yields that the distribution of $(\mathsf{pk}_3, \mathsf{sk}_3)$, $(\mathsf{pk}_4, \mathsf{sk}_4)$ is identical which proves our claim. So we see that the output $\tilde{b}$ of $\mathcal{A}$ is also identically distributed in G_3 and G_4, which yields

$$\Pr[\mathsf{S}_3] = \Pr[\mathsf{S}_4]$$

From game 4 to game 5 Next we analyze G_4 and G_5 and prove that they are also computationally indistinguishable. In G_5, the SFPK public key pk'_{i_b} in the challenge signature σ_{ch} is replaced by a random SFPK public key pk'. For that we prove the distribution of the output $\tilde{b}$ of any PPT adversary $\mathcal{D}$ does not differ significantly between G_4 and G_5. Let W_j denote the event that $\mathcal{D}$ outputs 1 in G_j for $j \in \{4, 5\}$. To prove our claim by contrapositon, we assume that $\mathcal{D}$ can distinguish the two games, so $|\Pr[\mathsf{W}_4] -$

$\Pr[\mathsf{W}_5]|$ is not negligible. From $\mathcal{D}$ we construct an adversary $\mathcal{E}$ breaking class-hiding of Π_{SFPK}. The basic idea of the reduction of distinguishing G_4 and G_5 to class-hiding is that if $\mathcal{E}$ is in class-hiding experiment 0, it simulates G_4 to $\mathcal{D}$, if $\mathcal{E}$ is in class-hiding experiment 1, it simulates G_5. The critical point is that $\mathcal{E}$ does not need to know whether it is in case $b = 0$ or $b = 1$ to do so. First we need to define the input for $\mathcal{E}$ in the class-hiding experiment (Def. 3.17). Let $\omega_0, \omega_1 \leftarrow_\$ \mathsf{coin}$, $(\mathsf{pk}_j, \mathsf{sk}_j) \leftarrow_\$ \mathsf{KGen}_{\mathsf{SFPK}}(\lambda, \omega_j)$ for $j \in \{0,1\}$, $r \in \mathsf{coin}$, $\mathsf{sk}' \leftarrow \mathsf{ChgSK}_{\mathsf{SFPK}}(\mathsf{sk}, r)$, $\mathsf{pk}' \leftarrow \mathsf{ChgPK}_{\mathsf{SFPK}}(\mathsf{pk}, r)$ as in the class-hiding experiment from Def. 3.17. On input $\omega_0, \omega_1, \mathsf{pk}'$, $\mathcal{E}$ works as follows:

1. $\mathcal{E}$ draws $i \leftarrow_\$ [n]$ and sets up the game for $\mathcal{D}$ as follows:
   a) $\mathcal{E}$ generates an SPS-EQ key pair $(\mathsf{pk}_{\mathsf{SPS}}, \mathsf{sk}_{\mathsf{SPS}}) \leftarrow_\$ \mathsf{KGen}_{\mathsf{SPS}}(BG, l)$
   b) for $j \in [n]$, $\mathcal{E}$ generates the personal secret key of user j as follows:
      i. $\omega_j \leftarrow_\$ \mathsf{coin}$
      ii. $\mathcal{E}$ generates $(\mathsf{pk}_j, \mathsf{sk}_j, \tau_j) \leftarrow_\$ \mathsf{TKGen}_{\mathsf{SFPK}}(\lambda, \omega_j)$ if $j \neq i$, otherwise it generates $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow_\$ \mathsf{KGen}_{\mathsf{SFPK}}(\lambda, \omega_0)$
      iii. $\mathcal{E}$ creates a certificate for pk_j as $\sigma^{(j)}_{\mathsf{SPS}} \leftarrow_\$ \mathsf{Sign}_{\mathsf{SPS}}(\mathsf{sk}_{\mathsf{SPS}}, \mathsf{pk}_j)$
      iv. $\mathcal{E}$ defines $\mathsf{gsk[j]} := (\mathsf{pk}_j, \mathsf{sk}_j, \sigma^{(j)}_{\mathsf{SPS}})$
   c) $\mathcal{E}$ defines $\mathsf{gpk} := (\mathsf{pk}_{\mathsf{SPS}}, BG)$ with $BG \leftarrow \mathsf{BGGen}(\lambda)$
   d) $\mathcal{E}$ defines $\mathsf{gmsk} := ((\mathsf{pk}_j, \tau_j))^n_{j=1, j\neq i}$
   e) $\mathcal{E}$ defines $\mathsf{gsk} := (\mathsf{gsk[j]})^n_{j=1}$
2. $\mathcal{E}$ starts $\mathcal{D}$ on input $(\mathsf{choose}, \mathsf{gpk}, \mathsf{gsk})$. When $\mathcal{D}$ submits state information St, two identities i_0, i_1 and a message m, $\mathcal{E}$ proceeds as follows to create the challenge signature σ_{ch} for $\mathcal{D}$:
   a) $\mathcal{E}$ computes $\sigma'_{\mathsf{SPS}} \leftarrow_\$ \mathsf{Sign}_{\mathsf{SPS}}(\mathsf{sk}_{\mathsf{SPS}}, \mathsf{pk}')$
   b) $\mathcal{E}$ defines $M := m||\sigma'_{\mathsf{SPS}}||\mathsf{pk}'$ and submits M to it's own SFPK signing oracle, which uses the secret key sk'. Eventually $\mathcal{E}$ obtains $\sigma_0 \leftarrow_\$ \mathsf{Sign}_{\mathsf{SFPK}}(\mathsf{sk}', M)$.
   c) $\mathcal{E}$ defines $\sigma_{ch} := (\mathsf{pk}', \sigma_0, \sigma'_{\mathsf{SPS}})$ and starts $\mathcal{D}$ on input $(\mathsf{guess}, \mathsf{St}, \sigma_{ch})$
3. $\mathcal{E}$ answers an opening query $(\tilde{m}, \tilde{\sigma})$ submitted by $\mathcal{D}$ as follows:
   a) $\mathcal{E}$ tries to open the signature by computing $o \leftarrow \mathsf{Open}(\mathsf{gmsk}, \tilde{m}, \tilde{\sigma})$
   b) if it obtains $o = \perp$, it returns i to $\mathcal{D}$, otherwise it returns o
4. When $\mathcal{D}$ eventually outputs a guess $\tilde{b} \in \{0,1\}$, $\mathcal{E}$ outputs $\tilde{b}$

The values choose and guess are constants indicating the current phase of the game. It is important to note that the group manager secret key gmsk is not created as required in the full-anonymity game, since no trapdoor for pk_i was generated. But the value o obtained by the opening try in the third step of the reduction is distributed as prescribed by the original game nevertheless. We already mentioned this in the discussion about

the G_4 change after the definition of the game sequence for this proof. Moreover we see that the change made in G_4 is of great importance for this reduction to work, since $\mathcal{E}$ can generate the first SFPK key pair from the class-hiding challenge by computing $(\mathsf{pk}_0, \mathsf{sk}_0) \leftarrow\!\!\$\, \mathsf{KGen}_{\mathsf{SFPK}}(\lambda, \omega_0)$ and reusing randomness ω_0 but cannot create a trapdoor for the class $[\mathsf{pk}_0]_{R'}$. So for $\mathcal{E}$ to be able to blend a key pair from its own class-hiding challenge into the personal secret keys, one of the key pairs in the personal secret keys must be allowed to have no corresponding trapdoor in gmsk. Furthermore we see that $\mathcal{E}$ obviously is a PPT adversary since

- $\mathcal{D}$ is a PPT adversary
- all algorithms used in the construction of $\mathcal{E}$ are PPT algorithms by definition (since they are part of either Π_{SFPK}, Π_{SPS} or $gFPK\text{-}GS_{\Pi_{\mathsf{SFPK}},\Pi_{\mathsf{SPS}}}$), namely
  - $\mathsf{KGen}_{\mathsf{SPS}}$
  - $\mathsf{TKGen}_{\mathsf{SFPK}}$
  - $\mathsf{KGen}_{\mathsf{SFPK}}$
  - $\mathsf{Sign}_{\mathsf{SPS}}$
  - Open
- sending messages and all other basic operations $\mathcal{E}$ performs can obviously be done by PPT algorithms

Next we prove that if $b = 0$, $\mathcal{E}$ simulates G_4 to $\mathcal{D}$, if $b = 1$, $\mathcal{E}$ simulates G_5 to $\mathcal{D}$. For that, we use the terminology established in Def. 5.1. We will start with the case $b = 0$ and prove that in that case $\mathcal{B}$ correctly simulates G_4 to $\mathcal{D}$. The view of $\mathcal{D}$ in G_4 is a tuple with the following structure:

$$\mathcal{V}_4 = (\mathsf{gpk}, \mathsf{gsk}, (m_d, \sigma_d, o_d)_{d=1}^{q_1}, \mathsf{St}, \sigma_{ch}, (m_d, \sigma_d, o_d)_{d=q_1+1}^{q_2})$$

with

- $\mathsf{gpk}, \mathsf{gsk}$ being a group public key and group secret key vector generated using GKGen
- For every $d \in [q_2]$, m_d is a message, σ_d is a signature, o_d is the response of the opening oracle to the query (m_d, σ_d). For reasons of convenience we define the term $\mathsf{query}_d := (m_d, \sigma_d, o_d)$ for every $d \in [q_2]$.
- St being a variable containing state information about the experiment
- σ_{ch} is the challenge signature, containing a randomized version pk'_{i_b} of pk_{i_b}, a certificate for this randomized public key and an SFPK signature created using a randomized version of the secret key sk_{i_b} corresponding to pk_{i_b}
- q_1 is the number of opening queries $\mathcal{A}$ submitted during the choose phase, q_2 is the total number of opening queries $\mathcal{A}$ submitted throughout course of the entire experiment with both q_1, q_2 being polynomials in λ

So next we iterate through the view of $\mathcal{D}$ in the case that $b = 0$ and prove that every random variable is distributed like in G_4 if we condition on arbitrary fixed values of all random variables prior in the view. Looking at the individual random variables in the view is sufficient since they can all be considered independent since none of them is computed directly from others without using fresh randomness in the computation. For that it is important that we look at each query query_d as a single random variable consisting of three values since $o_d := \mathsf{Open}(\mathsf{gmsk}, m_d, \sigma_d)$ is entirely determined by m_d, σ_d. We see that

- gpk is correctly distributed since BG is a bilinear group and $\mathsf{pk}_{\mathsf{SPS}}$ is an SPS-EQ public key generated using $\mathsf{KGen}_{\mathsf{SPS}}$ as required.

- gsk is correctly distributed since
  - the j-th entry $(\mathsf{pk}_j, \mathsf{sk}_j, \sigma_{\mathsf{SPS}}^{(j)})$ of gsk for $j \neq i$ is an SFPK key pair $(\mathsf{pk}_j, \mathsf{sk}_j)$ generated like $(\mathsf{pk}_j, \mathsf{sk}_j, \cdot) \leftarrow\!\!\$\ \mathsf{TKGen}_{\mathsf{SFPK}}(\lambda, \omega_j)$ as required using randomness $\omega_j \leftarrow\!\!\$\ \mathsf{coin}$ and an SPS-EQ certificate $\sigma_{\mathsf{SPS}}^{(j)} \leftarrow\!\!\$\ \mathsf{Sign}_{\mathsf{SPS}}(\mathsf{sk}_{\mathsf{SPS}}, \mathsf{pk}_j)$ for pk_j generated using $\mathsf{Sign}_{\mathsf{SPS}}$ and the SPS-EQ secret key $\mathsf{sk}_{\mathsf{SPS}}$ corresponding to $\mathsf{pk}_{\mathsf{SPS}}$ from the group public key gpk
  - the i-th entry $(\mathsf{pk}_i, \mathsf{sk}_i, \sigma_{\mathsf{SPS}}^{(i)})$ of gsk is an SFPK key pair $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow\!\!\$\ \mathsf{KGen}_{\mathsf{SFPK}}(\lambda, \omega_0)$ as required with $\omega_0 \leftarrow\!\!\$\ \mathsf{coin}$ being first randomness from the class-hiding challenge $\mathcal{E}$ was given. Furthermore $\sigma_{\mathsf{SPS}}^{(i)} \leftarrow\!\!\$\ \mathsf{Sign}_{\mathsf{SPS}}(\mathsf{sk}_{\mathsf{SPS}}, \mathsf{pk}_i)$ is an SPS-EQ certificate for pk_i generated using $\mathsf{Sign}_{\mathsf{SPS}}$ as required

- St obviously is correctly distributed since it is computed by $\mathcal{D}$ as required

- σ_{ch} is a *gFPK-GS*-signature containing a randomized version pk' of pk_i as its first component (note that due to the change in G_2, $i = i_b$ holds, so $\mathcal{B}$ has to set up the challenge signature as a signature created by user i). This is because if $b = 0$, $\mathsf{pk}' \in [\mathsf{pk}_i]$ holds due to the definition of the class-hiding game from Def. 3.17 (since pk_i is the first candidate public key pk_0 from the class-hiding challenge that was given to $\mathcal{E}$). Furthermore $\sigma'_{\mathsf{SPS}} \leftarrow\!\!\$\ \mathsf{Sign}_{\mathsf{SPS}}(\mathsf{sk}_{\mathsf{SPS}}, \mathsf{pk}')$ is an SPS-EQ certificate for pk' generated using $\mathsf{Sign}_{\mathsf{SPS}}$ as required, moreover $\sigma_0 \leftarrow\!\!\$\ \mathsf{Sign}_{\mathsf{SFPK}}(\mathsf{sk}', M)$ with $M := m||\sigma'_{\mathsf{SPS}}||\mathsf{pk}'$ is an SFPK signature as required. So all in all, $\sigma_{ch} := (\mathsf{pk}', \sigma_0, \sigma'_{\mathsf{SPS}})$ is a correctly distributed challenge signature.

- $\mathsf{query}_d := (m_d, \sigma_d, o_d)$ for $d \in [q_2]$ obviously is correctly distributed in the simulation since
  - if the originator behind the signature σ_d for the message m_d is $j \neq i$, then by construction of $\mathcal{E}$, $\mathcal{E}$'s opening try will retrieve its identity and $\mathcal{E}$ will return $o_d := j$ (since the trapdoor τ_j for the SFPK public key pk_j in the personal secret key $\mathsf{gsk}[\mathsf{j}]$ of this user j is contained in gmsk)

– if the originator behind the signature σ_d for the message m_d is i, $\mathcal{E}$ will fail to open (m_d, σ_d) (since no trapdoor was generated for the SFPK public key pk_i in the personal secret key $\mathsf{gsk}[i]$) of user i and will then return i by construction

So we see that in any case, $\mathcal{E}$ correctly retrieves the signer of a message-signature pair (m_d, σ_d) part of a query $\mathsf{query}_d := (m_d, \sigma_d, o_d)$ from $\mathcal{D}$.

So all in all we proved that if $b = 0$ holds, $\mathcal{E}$ correctly simulates G_4 to $\mathcal{D}$. Next we prove that if $b = 1$ holds, $\mathcal{E}$ correctly simulates G_5 to $\mathcal{D}$. The view of $\mathcal{D}$ in G_5 is a tuple with the following structure:

$$\mathcal{V}_5 = (\mathsf{gpk}, \mathsf{gsk}, (m_d, \sigma_d, o_d)_{d=1}^{q_1}, \mathsf{St}, \sigma_{ch}, (m_d, \sigma_d, o_d)_{d=q_1+1}^{q_2})$$

with

- $\mathsf{gpk}, \mathsf{gsk}$ being a group public key and group secret key vector generated using GKGen
- For every $d \in [q_2]$, m_d is a message, σ_d is a signature, o_d is the response of the opening oracle to the query (m_d, σ_d). For reasons of convenience we define the term $\mathsf{query}_d := (m_d, \sigma_d, o_d)$ for every $d \in [q_2]$.
- St being a variable containing state information about the experiment
- σ_{ch} is the challenge signature, containing a random public key pk', a certificate for this random public key and an SFPK signature created using a randomized version of the secret key sk' corresponding to pk'
- q_1 is the number of opening queries $\mathcal{A}$ submitted during the choose phase, q_2 is the total number of opening queries $\mathcal{A}$ submitted throughout course of the entire experiment with both q_1, q_2 being polynomials in λ

We see that apart from the challenge signature σ_{ch}, $\mathcal{D}$'s view in G_5 is the same as $\mathcal{D}$'s view in G_4 (this is not surprising since the only change from G_4 to G_5 is the way the challenge signature is computed). In the following, we analyze the distribution of every random variable in the view of $\mathcal{D}$ in the simulation in above reduction (with $b = 1$) and prove that it is identical to the distribution of the respective random variable in G_5. Looking at the individual random variables in the view is sufficient since they can all be considered independent since none of them is computed directly from others without using fresh randomness in the computation. For that it is important that we look at each query query_d as a single random variable consisting of three values since $o_d := \mathsf{Open}(\mathsf{gmsk}, m_d, \sigma_d)$ is entirely determined by m_d, σ_d. We see that:

- gpk is correctly distributed since BG is a bilinear group and $\mathsf{pk}_{\mathsf{SPS}}$ is an SPS-EQ public key generated using $\mathsf{KGen}_{\mathsf{SPS}}$ as required.
- gsk is correctly distributed since

  - the j-th entry $(\mathsf{pk}_j, \mathsf{sk}_j, \sigma_{\mathsf{SPS}}^{(j)})$ of gsk for $j \neq i$ is an SFPK key pair $(\mathsf{pk}_j, \mathsf{sk}_j)$ generated like $(\mathsf{pk}_j, \mathsf{sk}_j, \cdot) \leftarrow_\$ \mathsf{TKGen}_{\mathsf{SFPK}}(\lambda, \omega_j)$ as required using randomness $\omega_j \leftarrow_\$ \mathsf{coin}$ and an SPS-EQ certificate $\sigma_{\mathsf{SPS}}^{(j)} \leftarrow_\$ \mathsf{Sign}_{\mathsf{SPS}}(\mathsf{sk}_{\mathsf{SPS}}, \mathsf{pk}_j)$ for pk_j generated using $\mathsf{Sign}_{\mathsf{SPS}}$ and the SPS-EQ secret key $\mathsf{sk}_{\mathsf{SPS}}$ corresponding to $\mathsf{pk}_{\mathsf{SPS}}$ from the group public key gpk
  - the i-th entry $(\mathsf{pk}_i, \mathsf{sk}_i, \sigma_{\mathsf{SPS}}^{(i)})$ of gsk is an SFPK key pair $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow_\$ \mathsf{KGen}_{\mathsf{SFPK}}(\lambda, \omega_0)$ as required with $\omega_0 \leftarrow_\$ \mathsf{coin}$ being first randomness from the class-hiding challenge $\mathcal{E}$ was given. Furthermore $\sigma_{\mathsf{SPS}}^{(i)} \leftarrow_\$ \mathsf{Sign}_{\mathsf{SPS}}(\mathsf{sk}_{\mathsf{SPS}}, \mathsf{pk}_i)$ is an SPS-EQ certificate for pk_i generated using $\mathsf{Sign}_{\mathsf{SPS}}$ as required

- St obviously is correctly distributed since it is computed by $\mathcal{D}$ as required

- σ_{ch} is a *gFPK-GS*-signature containing a random public key pk' which is most likely not related to any SFPK public key pk_j contained in any $\mathsf{gsk}[\mathrm{j}]$ as its first component. This is because if $b = 1$, $\mathsf{pk}' \notin [\mathsf{pk}_i]$ most likely holds due to the definition of the class-hiding game from Def. 3.17 (since pk_i is the first candidate public key pk_0 from the class-hiding challenge that was given to $\mathcal{E}$ and in case $b = 1$, $\mathsf{pk}' \in [\mathsf{pk}_1]$ holds (pk_1 is the second candidate public key from the class-hiding challenge given to $\mathcal{E}$)). Furthermore $\sigma'_{\mathsf{SPS}} \leftarrow_\$ \mathsf{Sign}_{\mathsf{SPS}}(\mathsf{sk}_{\mathsf{SPS}}, \mathsf{pk}')$ is an SPS-EQ certificate for pk' generated using $\mathsf{Sign}_{\mathsf{SPS}}$ as required, moreover $\sigma_0 \leftarrow_\$ \mathsf{Sign}_{\mathsf{SFPK}}(\mathsf{sk}', M)$ with $M := m||\sigma'_{\mathsf{SPS}}||\mathsf{pk}'$ is an SFPK signature as required. So all in all, $\sigma_{ch} := (\mathsf{pk}', \sigma_0, \sigma'_{\mathsf{SPS}})$ is a correctly distributed challenge signature.

- $\mathsf{query}_d := (m_d, \sigma_d, o_d)$ for $d \in [q_2]$ obviously is correctly distributed in the simulation since

  - if the originator behind the signature σ_d for the message m_d is $j \neq i$, then by construction of $\mathcal{E}$, $\mathcal{E}$'s opening try will retrieve its identity and $\mathcal{E}$ will return $o_d := j$ (since the trapdoor τ_j for the SFPK public key pk_j in the personal secret key $\mathsf{gsk}[\mathrm{j}]$ of this user j is contained in gmsk)
  - if the originator behind the signature σ_d for the message m_d is i, $\mathcal{E}$ will fail to open (m_d, σ_d) (since no trapdoor was generated for the SFPK public key pk_i in the personal secret key $\mathsf{gsk}[\mathrm{i}]$) of user i and will then return i by construction

  So we see that in any case, $\mathcal{E}$ correctly retrieves the signer of a message-signature pair (m_d, σ_d) part of a query $\mathsf{query}_d := (m_d, \sigma_d, o_d)$ from $\mathcal{D}$.

So all in all we proved that if $b = 1$ holds, $\mathcal{E}$ correctly simulates G_5 to $\mathcal{D}$. We already proved above that if $b = 0$ holds, $\mathcal{E}$ correctly simulates G_4 to $\mathcal{D}$, so now we are done with analyzing the distribution of $\mathcal{D}$'s view in the simulation.

Next we prove that if $\mathcal{D}$ distinguishes G_4 and G_5, then $\mathcal{E}$ breaks the class-hiding property of Π_{SFPK}. For that we denote by W_j the event that $\mathcal{D}$ outputs 1 in G_j for $j \in \{4, 5\}$ and by $\mathsf{W}_b^{\mathsf{c-h}}$ the event that $\mathcal{E}$ outputs 1 in the class-hiding experiment

$Exp^{\mathsf{c-h-sfpk-b}}_{\Pi_{\mathsf{SFPK}},\mathcal{E}}(\lambda)$. The fact that $\mathcal{D}$ distinguishes G_4 and G_5 is formalized by $|\Pr[\mathsf{W}_4] - \Pr[\mathsf{W}_5|]$ being not negligible. We get

$$\begin{aligned} Adv^{\mathsf{c-h-sfpk}}_{\Pi_{\mathsf{SFPK}},\mathcal{A}}(\lambda) &= |\Pr[\mathsf{W}_0^{\mathsf{c-h}}] - \Pr[\mathsf{W}_1^{\mathsf{c-h}}]| \\ &= |\Pr[\mathsf{W}_4] - \Pr[\mathsf{W}_5]| \end{aligned}$$

which is not negligible since $\mathcal{D}$ distinguishes G_4 and G_5. The last equation sign holds since

$$\mathsf{W}_4 \text{ occurs } \Leftrightarrow \mathsf{W}_0^{\mathsf{c-h}} \text{ occurs}$$

and

$$\mathsf{W}_5 \text{ occurs } \Leftrightarrow \mathsf{W}_1^{\mathsf{c-h}} \text{ occurs}$$

since if $b = 0$ holds, $\mathcal{E}$ correctly simulates G_4 to $\mathcal{D}$, if $b = 1$ holds, $\mathcal{E}$ correctly simulates G_5 to $\mathcal{D}$ and $\mathcal{E}$ outputs the same bit as $\mathcal{D}$. So since we constructed an adversary $\mathcal{E}$ breaking class-hiding of Π_{SFPK} from an adversary $\mathcal{D}$ distinguishing G_4 and G_5, this means we proved that if G_4 and G_5 are computationally distinguishable, Π_{SFPK} is not class-hiding, which in contraposition means that if Π_{SFPK} is class-hiding, G_4 and G_5 are computationally indistinguishable, meaning for every PPT adversary $\mathcal{D}$ we have that $|\Pr[\mathsf{W}_4 - \mathsf{W}_5]|$ is negligible. We now need to use this fact to establish a relation between $\Pr[\mathsf{S}_4]$ and $\Pr[\mathsf{S}_5]$ with S_j denoting the event that a specific adversary $\mathcal{A}$ wins the full-anonymity game in G_j. In fact, we will prove that for every PPT adversary $\mathcal{A}$, $|\Pr[\mathsf{S}_4] - \Pr[\mathsf{S}_5]|$ is negligible in λ. We see that there exist negligible functions $\kappa_1, \kappa_2, \kappa_3$ such that

$$\begin{aligned} &|\Pr[\mathsf{S}_4] - \Pr[\mathsf{S}_5]| \\ =&|\Pr[\overline{\mathsf{W}_4} \wedge b = 0] + \Pr[\mathsf{W}_4 \wedge b = 1] - (\Pr[\overline{\mathsf{W}_5} \wedge b = 0] + \Pr[\mathsf{W}_5 \wedge b = 1])| \\ =&|1 - \Pr[\mathsf{W}_4 \wedge b = 0] + \Pr[\mathsf{W}_4 \wedge b = 1] - (1 - \Pr[\mathsf{W}_5 \wedge b = 0] + \Pr[\mathsf{W}_5 \wedge b = 1])| \\ =&|\Pr[\mathsf{W}_4 \wedge b = 1] - \Pr[\mathsf{W}_4 \wedge b = 0] + \Pr[\mathsf{W}_5 \wedge b = 0] - \Pr[\mathsf{W}_5 \wedge b = 1]| \\ =&|\Pr[\mathsf{W}_4 \wedge b = 1] - \Pr[\mathsf{W}_5 \wedge b = 1] + \Pr[\mathsf{W}_5 \wedge b = 0] - \Pr[\mathsf{W}_4 \wedge b = 0]| \\ =&\frac{1}{2}|\Pr[\mathsf{W}_4 \mid b = 1] - \Pr[\mathsf{W}_5 \mid b = 1] + \Pr[\mathsf{W}_5 \mid b = 0] - \Pr[\mathsf{W}_4 \mid b = 0]| \\ \leq&|\Pr[\mathsf{W}_4 \mid b = 1] - \Pr[\mathsf{W}_5 \mid b = 1]| + |\Pr[\mathsf{W}_5 \mid b = 0] - \Pr[\mathsf{W}_4 \mid b = 0]| \\ \leq&\kappa_1(\lambda) + \kappa_2(\lambda) \\ =&\kappa_3(\lambda) \end{aligned}$$

holds. In this computation we used computation rules for (conditional) probabilities as well as the triangle inequality. The second last inequality comes from the fact that G_4 and G_5 are computationally indistinguishable. With that we proved that $|\Pr[\mathsf{S}_4] - \Pr[\mathsf{S}_5]|$ is in fact negligible in λ.

Analysis of anonymity advantage In the last paragraphs, we established the following relations between $\Pr[\mathsf{S}_i]$, $\Pr[\mathsf{S}_{i+1}]$ for every $i \in \{0, \ldots, 4\}$:

- $\Pr[\mathsf{S}_0] = \Pr[\mathsf{S}_1]$
- $\Pr[\mathsf{S}_1] = n \cdot \Pr[\mathsf{S}_2]$, which is equivalent to $\frac{1}{n}\Pr[\mathsf{S}_1] = \Pr[\mathsf{S}_2]$
- $|\Pr[\mathsf{S}_2] - \Pr[\mathsf{S}_3]|$ is negligible in λ
- $\Pr[\mathsf{S}_3] = \Pr[\mathsf{S}_4]$
- $|\Pr[\mathsf{S}_4] - \Pr[\mathsf{S}_5]|$ is negligible in λ

This will now help us to upper-bound the winning probability $\Pr[\mathsf{S}_0]$ of an adversary $\mathcal{A}$ in the regular full-anonymity bit guessing game for $gFPK\text{-}GS_{\Pi_{\mathsf{SFPK}},\Pi_{\mathsf{SPS}}}$ from Def. 3.11 using the winning probability $\Pr[\mathsf{S}_5]$ of an adversary $\mathcal{A}$ in the bit-independent game G_5. After that, we only need to find a suitable upper bound to $\Pr[\mathsf{S}_5]$ to finish our proof of $gFPK\text{-}GS_{\Pi_{\mathsf{SFPK}},\Pi_{\mathsf{SPS}}}$'s full-anonymity. Using the triangle inequation it directly follows from above equations that there exist negligible functions κ_1, κ_2, κ_3 such that

$$
\begin{aligned}
&\Pr[\mathsf{S}_0] \\
=&|\Pr[\mathsf{S}_0]| \\
=&|\Pr[\mathsf{S}_0] - \Pr[\mathsf{S}_1] + \Pr[\mathsf{S}_1] - \Pr[\mathsf{S}_2] + \Pr[\mathsf{S}_2] - \Pr[\mathsf{S}_3] \\
&+ \Pr[\mathsf{S}_3] - \Pr[\mathsf{S}_4] + \Pr[\mathsf{S}_4] - \Pr[\mathsf{S}_5] + \Pr[\mathsf{S}_5]| \\
\leq&|\Pr[\mathsf{S}_0] - \Pr[\mathsf{S}_1]| + |\Pr[\mathsf{S}_1] - \Pr[\mathsf{S}_2]| + |\Pr[\mathsf{S}_2] - \Pr[\mathsf{S}_3]| \\
&+|\Pr[\mathsf{S}_3] - \Pr[\mathsf{S}_4]| + |\Pr[\mathsf{S}_4] - \Pr[\mathsf{S}_5]| + |\Pr[\mathsf{S}_5]| \\
=&0 + |\Pr[\mathsf{S}_1] - \Pr[\mathsf{S}_2]| + |\Pr[\mathsf{S}_2] - \Pr[\mathsf{S}_3]| + 0 + |\Pr[\mathsf{S}_4] - \Pr[\mathsf{S}_5]| + \Pr[\mathsf{S}_5] \\
=&0 + |(1 - \frac{1}{n}) \cdot \Pr[\mathsf{S}_1]| + |\Pr[\mathsf{S}_2] - \Pr[\mathsf{S}_3]| + 0 + |\Pr[\mathsf{S}_4] - \Pr[\mathsf{S}_5]| + \Pr[\mathsf{S}_5] \\
\leq&0 + (1 - \frac{1}{n}) \cdot \Pr[\mathsf{S}_1] + \kappa_1(\lambda) + 0 + \kappa_2(\lambda) + \Pr[\mathsf{S}_5] \\
=&(1 - \frac{1}{n}) \cdot \Pr[\mathsf{S}_0] + \kappa_1(\lambda) + \kappa_2(\lambda) + \Pr[\mathsf{S}_5] \\
=&(1 - \frac{1}{n}) \cdot \Pr[\mathsf{S}_0] + \kappa_3(\lambda) + \Pr[\mathsf{S}_5]
\end{aligned}
$$

which is equivalent to

$$\frac{1}{n} \cdot \Pr[\mathsf{S}_0] - \kappa_3(\lambda) \leq \Pr[\mathsf{S}_5] \tag{5.3}$$

Rearranging this yields the equivalent inequality

$$\Pr[\mathsf{S}_0] \leq n \cdot (\Pr[\mathsf{S}_5] + \kappa_3(\lambda)) \tag{5.4}$$

In the following we are going to prove that the view (for terminology see Def. 5.1) of $\mathcal{A}$ in G_5 is distributed identically no matter what the value of the hidden bit b is. So in that case, $\mathcal{A}$'s view in G_5 is not depending on b, which yields that its output $\tilde{b}$ is also identically distributed in both cases $b = 0$, $b = 1$. The view of $\mathcal{A}$ in G_5 is a tuple with

the following structure:

$$\mathcal{V}_5 = (\mathsf{gpk}, \mathsf{gsk}, (m_d, \sigma_d, o_d)_{d=1}^{q_1}, \mathsf{St}, \sigma_{ch}, (m_d, \sigma_d, o_d)_{d=q_1+1}^{q_2})$$

with

- $\mathsf{gpk}, \mathsf{gsk}$ being a group public key and group secret key vector generated using GKGen
- For every $d \in [q_2]$, m_d is a message, σ_d is a signature, o_d is the response of the opening oracle to the query (m_d, σ_d). For reasons of convenience we define the term $\mathsf{query}_d := (m_d, \sigma_d, o_d)$ for every $d \in [q_2]$.
- St being a variable containing state information about the experiment
- σ_{ch} is the challenge signature, containing a random public key pk', a certificate for this random public key and an SFPK signature created using a randomized version of the secret key sk' corresponding to pk'
- q_1 is the number of opening queries $\mathcal{A}$ submitted during the choose phase, q_2 is the total number of opening queries $\mathcal{A}$ submitted throughout course of the entire experiment with both q_1, q_2 being polynomials in λ

To prove that $\mathcal{V}_5$ is independent of b, we prove that every random variable contained in $\mathcal{V}_5$ is distributed identically in both cases $b = 0$ and $b = 1$. Hereby looking at the individual variables is sufficient to make the required statement about their common distribution (the distribution of $\mathcal{V}_5$) since all of them can be considered independent because none of them is computed directly from others without using fresh randomness in the computation. For that is important that we look at each query query_d as a single random variable consisting of three values since $o_d := \mathsf{Open}(\mathsf{gmsk}, m_d, \sigma_d)$ is entirely determined by m_d, σ_d. We see that

- gpk obviously is identically distributed in both cases with BG being a bilinear group and $\mathsf{pk}_{\mathsf{SPS}}$ being an SPS-EQ public key generated using $\mathsf{KGen}_{\mathsf{SPS}}$
- gsk is identically distributed in both cases since the j-th entry $(\mathsf{pk}_j, \mathsf{sk}_j, \sigma_{\mathsf{SPS}}^{(j)})$ of gsk for $j \in [n]$ is an SFPK key pair $(\mathsf{pk}_j, \mathsf{sk}_j)$ generated like $(\mathsf{pk}_j, \mathsf{sk}_j, \cdot) \leftarrow_{\$} \mathsf{TKGen}_{\mathsf{SFPK}}(\lambda, \omega_j)$ using randomness $\omega_j \leftarrow_{\$} \mathsf{coin}$ and an SPS-EQ certificate $\sigma_{\mathsf{SPS}}^{(j)} \leftarrow_{\$} \mathsf{Sign}_{\mathsf{SPS}}(\mathsf{sk}_{\mathsf{SPS}}, \mathsf{pk}_j)$ for pk_j generated using $\mathsf{Sign}_{\mathsf{SPS}}$ and the SPS-EQ secret key $\mathsf{sk}_{\mathsf{SPS}}$ corresponding to $\mathsf{pk}_{\mathsf{SPS}}$ from the group public key gpk in both cases
- St obviously is identically distributed in both cases since it is computed by $\mathcal{D}$ in both cases
- in both cases the public key pk which is part of the strong unforgeability challenge given to $\mathcal{B}$ clearly is independent of b, so the same holds for it's randomized version

pk' which is part of the challenge signature σ_{ch}. This yields that $\sigma_{\$}$ is also independent of b. Since the challenge secret key sk from the above unforgeability challenge and it's randomized version sk' used in the creation of the challenge signature are clearly also independent of b, we get that the SFPK signature σ_0 contained in σ_{ch} is independent of b as well. All in all this yields that the challenge signature σ_{ch} is independent of b and therefore identically distributed in both cases $b = 0$ and $b = 1$.

- it is clear to see that each query (m_d, σ_d) that $\mathcal{A}$ submits to its opening oracle throughout the course of G_5 is independent of b since $\mathcal{A}$ cannot access b. Since $\mathcal{B}$'s answer to that query is already determined by the query itself (since Open is a deterministic algorithm), o_d therefore also is independent of b. With that we get that for all $d \in [q_2]$, $\mathsf{query}_d := (m_d, \sigma_d, o_d)$ is independent of b and therefore identically distributed in both cases $b = 0$, $b = 1$.

All in all we get that the view of $\mathcal{A}$ in G_5 is independent of b since it is identically distributed in both cases $b = 0$, $b = 1$ since all of its components are independent and identically distributed in both cases $b = 0$, $b = 1$.

However, if the adversary $\mathcal{A}$ somehow manages to create a randomized version $\sigma^*_{ch} = (\mathsf{pk}^*, \sigma^*_0, \sigma^*_{\$})$ of the challenge signature $\sigma_{ch} = (\mathsf{pk}', \sigma_0, \sigma_{\$})$ (according to Def. 4.4) and submits it to the opening oracle, the opening oracle will fail to open σ^*_{ch} (since most likely no $j \in [n]$ with $\mathsf{pk}^* \in [\mathsf{pk}_j]_{R'}$ exists) which means by the change in G_3, that i will be returned as the answer to the query. With that the adversary now has access to a pk^* related to the public key pk' in the challenge signature and gets the information that a valid $gFPK\text{-}GS_{\Pi_{\mathsf{SFPK}},\Pi_{\mathsf{SPS}}}$-signature containing pk^* opens to i. By the construction of Open in $gFPK\text{-}GS_{\Pi_{\mathsf{SFPK}},\Pi_{\mathsf{SPS}}}$, $\mathcal{A}$ can now come to the conclusion that σ_{ch} also opens to i and output $\tilde{b}$ such that $i_{\tilde{b}} = i$ which makes him win the game. Consequently we see that if $\mathcal{A}$ manages to randomize the challenge signature σ_{ch}, it has a chance to win G_5. Let R denote the event of $\mathcal{A}$ submitting such a randomized version σ^*_{ch} of σ_{ch}. Therefore to estimate $\Pr[\mathsf{S}_5]$ we must upper-bound the probability $\Pr[\mathsf{R}]$. For that we prove that if R occurs, $\mathcal{A}$ wins the strong unforgeability experiment for Π_{SFPK} for the random key pair $(\mathsf{pk}, \mathsf{sk})$ from G_5, which yields

$$\Pr[\mathsf{R}] \leq Adv^{\mathsf{seuf-sfpk}}_{\Pi_{\mathsf{SFPK}},\mathcal{A}}(\lambda) \tag{5.5}$$

So with the strong existential unforgeability of Π_{SFPK} being a prerequisite for Thm. 5.3, we get that there exists a negligible function κ with

$$\Pr[\mathsf{R}] \leq \kappa(\lambda)$$

Therefore what is left to prove is that if R occurs throughout the course of G_5, $\mathcal{A}$ wins the strong unforgeability experiment for Π_{SFPK} for the random key pair $(\mathsf{pk}, \mathsf{sk})$ from G_5. For proving this, we will use the terminology from G_5. Let $\sigma^*_{ch} := (\mathsf{pk}^*, \sigma^*_0, \sigma^*_{\mathsf{SPS}})$ be a randomized version of the challenge signature $\sigma_{ch} := (\mathsf{pk}', \sigma_0, \sigma'_{\mathsf{SPS}})$. So by definition (see Def. 4.4) we get

- $\mathsf{pk}'\ R'\ \mathsf{pk}^*$
- $\sigma_{ch} \neq \sigma^*_{ch}$

Let $M := m||\sigma'_{\mathsf{SPS}}||\mathsf{pk}'$, $M^* := m||\sigma^*_{\mathsf{SPS}}||\mathsf{pk}^*$. We claim that when outputting $(\mathsf{pk}^*, M^*, \sigma^*_0)$, $\mathcal{A}$ wins the strong existential unforgeability game for Π_{SFPK} for $(\mathsf{pk}, \mathsf{sk})$. We see that by construction of GSign in $gFPK\text{-}GS_{\Pi_{\mathsf{SFPK}},\Pi_{\mathsf{SPS}}}$, (M^*, σ^*_0) is a valid SFPK message-signature pair under pk and we get

$$\mathsf{pk}' \neq \mathsf{pk}^* \vee \sigma'_{\mathsf{SPS}} \neq \sigma^*_{\mathsf{SPS}} \Rightarrow M \neq M^*$$

So if $\mathsf{pk}' \neq \mathsf{pk}^* \vee \sigma'_{\mathsf{SPS}} \neq \sigma^*_{\mathsf{SPS}}$ holds it is easy to see that $\mathcal{A}$ wins the strong existential unforgeability game since it forged a signature for a message that has never been signed before with sk (since M is the only message that has been signed with sk). If on the other hand $\mathsf{pk}' = \mathsf{pk}^* \wedge \sigma'_{\mathsf{SPS}} = \sigma^*_{\mathsf{SPS}}$ holds, we get that $\sigma_0 \neq \sigma^*_0$ holds since a randomized signature must be distinct from the original one by definition. This yields that in this case, $\mathcal{A}$ managed to create a distinct and valid signature for a message that has been signed before using sk (an SFPK secret key not known to $\mathcal{A}$) which by definition of the strong existential unforgeability experiment $Exp^{\mathsf{seuf-sfpk}}_{\Pi_{\mathsf{SFPK}},\mathcal{A}}(\lambda)$ also means that $\mathcal{A}$ wins the game. With that we proved Eq. (5.5) which means there exists a negligible function κ such that

$$\Pr[\mathsf{R}] \leq \kappa(\lambda)$$

An interesting side remark is that in the case that

$$\mathsf{pk}' \neq \mathsf{pk}^* \vee \sigma'_{\mathsf{SPS}} \neq \sigma^*_{\mathsf{SPS}}$$

holds, $\mathcal{A}$ would even win the weak EUF-CMA game from Def. 3.20 with the forgery (M^*, σ^*_0), since $M^* \neq M$ and M was the only message that was signed before. But in the case that

$$\mathsf{pk}' = \mathsf{pk}^* \wedge \sigma'_{\mathsf{SPS}} = \sigma^*_{\mathsf{SPS}}$$

holds, (M^*, σ^*_0) is not a suitable forgery to win the weak existential unforgeability game since $M^* = M$ and M has been signed before. This gives an intuition of why strong existential unforgeability of Π_{SFPK} is required for the full-anonymity of $gFPK\text{-}GS_{\Pi_{\mathsf{SFPK}},\Pi_{\mathsf{SPS}}}$. With Eq. (5.5) and the fact that $\mathcal{A}$'s output $\tilde{b}$ is independent of the hidden bit b, we get that

$$\begin{aligned}
\Pr[\mathsf{S}_5] =& \Pr[\mathsf{S}_5 \mid \mathsf{R}] \cdot \Pr[\mathsf{R}] + \Pr[\mathsf{S}_5 \mid \overline{\mathsf{R}}] \cdot \Pr[\overline{\mathsf{R}}] \\
\leq& \Pr[\mathsf{R}] + \Pr[\mathsf{S}_5 \mid \overline{\mathsf{R}}] \\
=& Adv^{\mathsf{seuf-sfpk}}_{\Pi_{\mathsf{SFPK}},\mathcal{A}}(\lambda) + \frac{1}{2n} \\
\leq& \kappa(\lambda) + \frac{1}{2n}
\end{aligned}$$

The last equation deserves a little more attention. Let S'_5 denote the event that S_5 occurs

under the prerequisite that R does not occur (so S_5' is the event that $\mathcal{A}$ guesses the bit correctly in G_5, $\mathcal{A}$ does not submit a randomized version of the challenge signature to the opening oracle and the identity i matches the challenge identity i_b (since otherwise, G_5 would be aborted after the choose-phase)). We see that under that prerequisite, the hidden bit b and $\mathcal{A}$'s output $\tilde{b}$ are independent random variables, furthermore the events $i = i_b$ and $\tilde{b} = b$ are also independent. With this we get

$$\begin{aligned}
\Pr[\mathsf{S}_5'] &= \Pr[i = i_b \wedge \tilde{b} = b] \\
&= \Pr[i = i_b] \cdot \Pr[\tilde{b} = b] \\
&= \frac{1}{n} \cdot \Pr[\tilde{b} = b] \\
&= \frac{1}{n}(\Pr[\tilde{b} = 0 \mid b = 0] \cdot \Pr[b = 0] + \Pr[\tilde{b} = 1 \mid b = 1] \cdot \Pr[b = 1]) \\
&= \frac{1}{n}(\Pr[\tilde{b} = 0 \mid b = 0] \cdot \frac{1}{2} + \Pr[\tilde{b} = 1 \mid b = 1] \cdot \frac{1}{2}) \\
&= \frac{1}{n}(\Pr[\tilde{b} = 0 \mid b = 0] \cdot \frac{1}{2} + (1 - \Pr[\tilde{b} = 0 \mid b = 1]) \cdot \frac{1}{2}) \\
&= \frac{1}{n}(\frac{\Pr[\tilde{b} = 0 \wedge b = 0]}{\Pr[b = 0]} \cdot \frac{1}{2} + (1 - \frac{\Pr[\tilde{b} = 0 \wedge b = 1]}{\Pr[b = 1]}) \cdot \frac{1}{2}) \\
&= \frac{1}{n}(\frac{\Pr[\tilde{b} = 0] \cdot \Pr[b = 0]}{\Pr[b = 0]} \cdot \frac{1}{2} + (1 - \frac{\Pr[\tilde{b} = 0] \cdot \Pr[b = 1]}{\Pr[b = 1]}) \cdot \frac{1}{2}) \\
&= \frac{1}{n}(\Pr[\tilde{b} = 0] \cdot \frac{1}{2} + (1 - \Pr[\tilde{b} = 0]) \cdot \frac{1}{2}) \\
&= \frac{1}{n}((\Pr[\tilde{b} = 0] + 1 - \Pr[\tilde{b} = 0]) \cdot \frac{1}{2}) \\
&= \frac{1}{n} \cdot \frac{1}{2} \\
&= \frac{1}{2n}
\end{aligned}$$

which proves the last equation from the previous computation. So all in all we get that there exists a negligible function κ such that

$$\Pr[\mathsf{S}_5] \leq \kappa(\lambda) + \frac{1}{2n} \tag{5.6}$$

as claimed, which finally allows us to upper-bound the advantage of an arbitrary PPT adversary $\mathcal{A}$ against full-anonymity of $gFPK\text{-}GS_{\Pi_{\mathsf{SFPK}},\Pi_{\mathsf{SPS}}}$. For doing so, we use Eq. (5.4) and Eq. (5.6) and the fact that we can assume $\Pr[\mathsf{S}_0] \geq \frac{1}{2}$. This follows from the fact that from every adversary $\mathcal{A}$ with a winning probability $p \leq \frac{1}{2}$ we can construct an adversary $\mathcal{A}'$ with winning probability $1 - p \geq \frac{1}{2}$ by simply flipping the bit output by $\mathcal{A}$. With that we get that there exist negligible functions $\kappa_1, \kappa_2, \kappa_3, \kappa_4$ such that

$$Adv^{\mathsf{anon}}_{gFPK\text{-}GS_{\Pi_{\mathsf{SFPK}},\Pi_{\mathsf{SPS}}},\mathcal{A}}(\lambda, n) = |\Pr[\mathsf{S}_0] - \frac{1}{2}|$$

$$
\begin{aligned}
&= \Pr[\mathsf{S}_0] - \frac{1}{2} \\
&\leq n \cdot (\Pr[\mathsf{S}_5] + \kappa_1(\lambda)) - \frac{1}{2} \\
&\leq n \cdot (\frac{1}{2n} + \kappa_2(\lambda) + \kappa_1(\lambda)) - \frac{1}{2} \\
&\leq n \cdot (\frac{1}{2n} + \kappa_3(\lambda)) - \frac{1}{2} \\
&\leq \frac{1}{2} + \kappa_4(\lambda) - \frac{1}{2} \\
&= \kappa_4(\lambda)
\end{aligned}
$$

So for any PPT adversary $\mathcal{A}$ we have

$$Adv^{\mathsf{anon}}_{gFPK\text{-}GS_{\Pi_{\mathsf{SFPK}},\Pi_{\mathsf{SPS}}},\mathcal{A}}(\lambda, n) \text{ is negligible.}$$

which means that $gFPK\text{-}GS_{\Pi_{\mathsf{SFPK}},\Pi_{\mathsf{SPS}}}$ is fully-anonymous according to Def. 3.10. □

We will give proof of the Difference Lemma here for completeness.

Lemma 5.4 (**Difference Lemma**) *Let W_0, W_1, Z be events defined over the same probability space with*

$$W_0 \wedge \overline{Z} \Leftrightarrow W_1 \wedge \overline{Z}$$

Then it holds that

$$|\Pr[W_0] - \Pr[W_1]| \leq \Pr[Z]$$

Proof. We see that

$$
\begin{aligned}
|\Pr[W_0] - \Pr[W_1]| &= |\Pr[W_0 \wedge Z] + \Pr[W_0 \wedge \overline{Z}] - (\Pr[W_1 \wedge Z] + \Pr[W_1 \wedge \overline{Z}])| \\
&= |\Pr[W_0 \wedge Z] + \Pr[W_0 \wedge \overline{Z}] - \Pr[W_1 \wedge Z] - \Pr[W_1 \wedge \overline{Z}]| \\
&= |\Pr[W_0 \wedge Z] - \Pr[W_1 \wedge Z]| \\
&\leq \Pr[Z]
\end{aligned}
$$

The third equation sign holds since we have $W_0 \wedge \overline{Z} \Leftrightarrow W_1 \wedge \overline{Z}$ by assumption so the probabilities of these two events are obviously equal. The last inequality holds since $\Pr[W_0 \wedge Z]$ ad $\Pr[W_1 \wedge Z]$ are obviously less or equal than $\Pr[Z]$ and greater or equal than 0. □

5.3 Original approaches

In this section, we are going to discuss the approaches for the security proofs for *gFPK-GS* that were sketched by Backes et al. in the appendix of [3].

Full-traceability An equivalent formulation of Thm. 5.2 can be found as Theorem 3 in the original paper [3]. The authors tackle the proof with a sequence of games approach, using the following sequence of games:

G_0**:** regular full-traceability game from Def. 3.13

G_1**:** G_0 but the game is aborted if the adversary $\mathcal{A}$ forges a valid and unopenable (see Def. 3.7) signature

G_2**:** G_1 but before the beginning of the game an identity $i \leftarrow\!\!\$\, [n]$ is drawn and the game is aborted if $\mathcal{A}$ forges a signature that does not open to i

With S_j denoting the event that $\mathcal{A}$ wins in G_j (meaning it outputs a forgery that is either unopenable or opens to an identity not included in $\mathcal{A}$'s collusion set), Backes et al. conclude

$$|\Pr[\mathsf{S}_0] - \Pr[\mathsf{S}_1]| \leq Adv^{\mathsf{euf-sps}}_{\Pi_{\mathsf{SPS}},\mathcal{A}}(\lambda) \leq \kappa(\lambda) \tag{5.7}$$

for a suitable negligible function κ and

$$\Pr[\mathsf{S}_1] = n \cdot \Pr[\mathsf{S}_2] \tag{5.8}$$

and provide a sketch of a proof for Eq. (5.7). It holds since an unopenable yet valid forgery against full-traceability of *gFPK-GS* can be used to win the existential unforgeability game for Π_{SPS} according to Def. 3.23. This follows from the fact that a valid forgery must contain an SFPK public key and a valid SPS-EQ certificate for it and that the public key in an unopenable $gFPK\text{-}GS_{\Pi_{\mathsf{SFPK}},\Pi_{\mathsf{SPS}}}$-signature must be in a different equivalence class than all SFPK public keys pk_j in the personal secret keys $\mathsf{gsk}[j]$. So an adversary who is able to forge a valid an unopenable signature is capable of creating a valid SPS-EQ signature for a new message (here: the public key from the forgery which is in the message space of Π_{SPS} by construction of $gFPK\text{-}GS_{\Pi_{\mathsf{SFPK}},\Pi_{\mathsf{SPS}}}$) and therefore clearly breaks the existential unforgeability of Π_{SPS}. We see that the adversary $\mathcal{D}$ against existential unforgeability of Π_{SPS} from the proof of full-traceability of $gFPK\text{-}GS_{\Pi_{\mathsf{SFPK}},\Pi_{\mathsf{SPS}}}$ that is given in this thesis is based on the same idea of how to break existential unforgeability of Π_{SPS}. However, in this thesis, a formal description of $\mathcal{D}$ was added as well as a proof that it has non-negligible advantage in breaking the existential unforgeability of Π_{SPS}. With that done, Backes et al. sketch how an adversary $\mathcal{A}$ winning in G_2 can be used to win the existential unforgeability game for the SFPK scheme Π_{SFPK}. Their reduction is analogous to the construction of the adversary $\mathcal{B}$ in the proof of Thm. 5.2 in this thesis, making use of the fact that the SFPK signature σ_0^* from the forgery $(m^*, (\mathsf{pk}^*, \sigma_0^*, \sigma_{\mathsf{SPS}}^*))$ $\mathcal{A}$ outputs is a valid signature for $M^* := m^* || \sigma_{\mathsf{SPS}}^* || \mathsf{pk}^*$ under $\mathsf{pk}^* \in [\mathsf{pk}_i]_{R'}$. Since M^* was never signed before, (M^*, σ_0^*) therefore is a forgery that allows to win the existential unforgeability experiment from Def. 3.20 for the challenge key pair $(\mathsf{pk}_i, \mathsf{sk}_i)$ from the personal secret key $\mathsf{gsk}[i]$ of user i. So with that Backes et al. conclude

$$\Pr[\mathsf{S}_2] = Adv^{\mathsf{seuf-sfpk}}_{\Pi_{\mathsf{SFPK}},\mathcal{A}}(\lambda)$$

which allows them to upper-bound the winning probability $\Pr[\mathsf{S}_0]$ of an arbitrary PPT adversary $\mathcal{A}$ to

$$n \cdot Adv^{\mathsf{euf-sfpk}}_{\Pi_{\mathsf{SFPK}},\mathcal{A}}(\lambda) + Adv^{\mathsf{euf-sps}}_{\Pi_{\mathsf{SPS}},\mathcal{A}}(\lambda)$$

Since this is a negligible term by prerequisite, the proof of full-traceability of $gFPK\text{-}GS_{\Pi_{\mathsf{SFPK}},\Pi_{\mathsf{SPS}}}$ is finished with that inequality. Again, the formal description of above reduction algorithm was not included in [3] and added in this thesis (Sect. 5.1). As mentioned above, the proof for full-traceability under Thm. 5.2 in this thesis uses similiar intuitions to reduce full-traceability of $gFPK\text{-}GS_{\Pi_{\mathsf{SFPK}},\Pi_{\mathsf{SPS}}}$ to existential unforgeability of Π_{SFPK} and Π_{SPS}. However, the framework of the proof in this thesis is different since it does not use a sequence of games but constructs two independent adversaries $\mathcal{B}$ and $\mathcal{D}$, one against each of the prerequisited security requirements, and proves that for each adversary $\mathcal{A}$ against the full-traceability of $gFPK\text{-}GS_{\Pi_{\mathsf{SFPK}},\Pi_{\mathsf{SPS}}}$ exactly one of the constructed adversaries has non-negligible advantage in winning the respective security game. Thus the approach chosen in this thesis directly maps the ways to win $Exp^{\mathsf{trace}}_{gFPK\text{-}GS_{\Pi_{\mathsf{SFPK}},\Pi_{\mathsf{SPS}}},\mathcal{A}}(\lambda, n)$ to the security requirements and can therefore be considered more intuitive than the sequence of games approach in [3].

Full-anonymity An equivalent formulation of Thm. 5.3 can be found as Theorem 4 in the original paper [3]. As in their proof sketch for full-traceability of $gFPK\text{-}GS_{\Pi_{\mathsf{SFPK}},\Pi_{\mathsf{SPS}}}$, the authors choose an approach based on a sequence of games. The proof of full-anonymity of $gFPK\text{-}GS_{\Pi_{\mathsf{SFPK}},\Pi_{\mathsf{SPS}}}$ in this thesis uses the same sequence of games and follows the extended proof sketch from [3], especially on the last transition in the game sequence and the way the fact that G_5 is independent of b is used. However, we added the complete pseudocode for all games in the sequence (see Appendix A), a detailed analysis of the game hops and the relations between the winning probabilities $\Pr[\mathsf{S}_j]$ of the adversary in G_j for all j. In the original work by Backes et al. the reduction of distingushing games 4 and 5 to class-hiding was only sketched, without formally writing out the reduction algorithm or analyzing its advantage. We added this in our proof (see Thm. 5.3). In addition, the proof that

- G_5 is independent of the hidden bit
- therefore the only way to win G_5 is guessing the bit at random or randomizing the challenge signature

was also skipped in the original work [3] and done in detail in this thesis. From these two facts, the authors concluded that the winning probability $\Pr[\mathsf{S}_5]$ was equal to the strong existential unforgeability advantage $Adv^{\mathsf{seuf-sfpk}}_{gFPK\text{-}GS_{\Pi_{\mathsf{SFPK}},\Pi_{\mathsf{SPS}}},\mathcal{A}}(\lambda)$ of the adversary $\mathcal{A}$ which was proven wrong in this thesis, in fact we get

$$\Pr[\mathsf{S}_5] \leq \frac{1}{2n} + Adv^{\mathsf{seuf-sfpk}}_{gFPK\text{-}GS_{\Pi_{\mathsf{SFPK}},\Pi_{\mathsf{SPS}}},\mathcal{A}}(\lambda)$$

as it can be seen in Sect. 5.2. Finally note that the definition of full-anonymity used for the extended proof sketch in [3] contains a major flaw which makes it useless for

any meaningful statements on a group signature scheme's security (see Sect. 3.7.2 for details). This makes the proof outline of the full-anonymity proof given in the appendix of [3] a lot harder to follow since it is not clear whether the authors intended to define full-anonymity via a distinguishing or bit-guessing game.

6 Future work

In Chapter 5, we gave full proof of the full-traceability (Thm. 5.2) and full-anonymity (Thm. 5.3) of the group signature scheme *gFPK-GS* by Backes et al. [3]. A detailed description of our additions to the original proof sketches given in [3] can be found in Sect. 5.3. To finish the thesis, we outline possible consecutive work to the findings in this thesis. More precisely, we look at a fully-dynamic group signature scheme from standard assumptions that Backes et al. presented in [4].

In [4], Backes et al. addressed the issue of application scenarios where even membership in a certain group is confidential information by extending the security model by Bootle et al. [8] by a new security notion for fully-dynamic group signature schemes which they call *membership privacy*. Intuitively, this notion means that any information about the list of members of a group shall be hidden from the public. To outline how membership privacy is formalized in terms of security experiments, we need to recall the notion of an *epoch* for group signature schemes (see [8]). Any period of time that the list of members of a given group does not change is called an epoch of that group. Information about the state of a group is published at the beginning of every epoch. The critical point when it comes to membership privacy therefore is to give this information about the group to the public without revealing any information that can help to identify the group's members. Backes et al. [4] split up membership privacy into the two requirements *join privacy* and *leave privacy*. Join privacy means that given two users who are non-members of a given group in a given epoch, it is infeasible to determine which of the two has become a member in the next epoch. Leave privacy is then defined analogously, given two group members, it shall be infeasible to tell which of them left the group in the next epoch. With membership privacy being defined formally, Backes et al. [4] continue with describing a generic construction of a fully-dynamic group signature fulfilling the requirements of membership privacy which combines SFPK and SPS-EQ in a similar way than *gFPK-GS*. We will refer to this generic construction by *gFPK-FDGS*.

We will continue to describe the intuition behind the membership privacy of *gFPK-FDGS* and the process of signing a message using this scheme. In contrast to *gFPK-GS*, the SFPK public key pk_i of user i is no longer certified itself but first randomized by the issuing authority before the randomized SFPK key pk'_i is signed with an SPS-EQ scheme. The outcome of the certfication process is the certificate $\sigma^{(i)}_{\mathsf{SPS}}$ for pk'_i alongside with a public-key encryption c of the randomness used to obtain it. Note that the secret key corresponding to the public key used to encrypt the randomness is only known to user i. Therefore only user i can decrypt c and obtain the decrypted randomness, which means that no one else can know the public key pk'_i that was signed. Note that in each epoch, a fresh SPS-EQ key pair is used to create new certficates for

each member of the group. Together with the facts that

- the SFPK public keys are not certified themselves but made unreadable to the public before the certificate is computed (so no certificate can be linked to a specific group member trivially)
- the SFPK public keys published which are published by the group authorities are randomized

this yields membership privacy of *gFPK-FDGS*. To sign a message using *gFPK-FDGS*, user i randomizes its SFPK key pair $(\mathsf{pk}_i, \mathsf{sk}_i)$ as in *gFPK-GS* and adapts the SPS-EQ certificate to fit the obtained randomized SFPK public key pk'_i. After that, it creates a proof of knowledge Π_{SFPK} of the globally known unique representative pk_i of the equivalence class of its SFPK public key $[\mathsf{pk}_i]_R = [\mathsf{pk}'_i]_R$ and the randomness r used above to randomize the SFPK key pair. The resulting group signature contains an SFPK signature under sk'_i over the the current epoch information and all data computed in the process of signing. The other components of the produced group signature are the public key pk'_i corresponding to sk'_i, the adapted SPS-EQ certificate $\sigma^{(i)\prime}_{\mathsf{SPS}}$ for it, the proof Π_{SFPK}, and a public-key encryption c_{SFPK} of the unique representative pk_i under a public key of a PKE key pair belonging to the opening authority. With c_{SFPK}, the unique representative pk_i of the SFPK public key user i is included in the signature in a way that only the opening authority can read it. This is needed to allow for subsequent openings of signatures while at the same time hiding the identity of the signer from the public.

As they did for *gFPK-GS* in [3], Backes et al. provided extended proof sketches for the relevant security properties of *gFPK-FDGS* in the full version of [4]. These include updated versions of full-anonymity and full-traceability as well as non-frameability and tracing soundness, which were all introduced by Bootle et al. in [8]. Furthermore Backes et al. prove the membership privacy of *gFPK-FDGS* by proving join and leave privacy according to their new definitions in [4]. The above mentioned extended proof sketches can be found as Theorems 3 to 8 in the full version of [4]. However, writing full formal proofs for the security properties of *gFPK-FDGS* is still an open task that might give deeper insight on where the security of *gFPK-FDGS* comes from.

Bibliography

[1] Giuseppe Ateniese, Jan Camenisch, Marc Joye, and Gene Tsudik. A practical and provably secure coalition-resistant group signature scheme. In *Annual International Cryptology Conference*, pages 255–270. Springer, 2000.

[2] Michael Backes, Lucjan Hanzlik, Kamil Kluczniak, and Jonas Schneider. Signatures with flexible public key: A unified approach to privacy-preserving signatures (full version). *IACR Cryptology ePrint Archive*, 2018:191, 2018.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] Dan Boneh and Victor Shoup. A graduate course in applied cryptography. *Draft 0.2*, 2015.

[8] 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.

[9] David Chaum and Eugène Van Heyst. Group signatures. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 257–265. Springer, 1991.

[10] Georg Fuchsbauer. Breaking existential unforgeability of a signature scheme from asiacrypt 2014. *IACR Cryptology ePrint Archive*, 2014:892, 2014.

[11] Georg Fuchsbauer, Christian Hanser, and Daniel Slamanig. Euf-cma-secure structure-preserving signatures on equivalence classes. *IACR Cryptology ePrint Archive*, 2014:944, 2014.

[12] 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.

[13] Benoît Libert, Thomas Peters, and Moti Yung. Short group signatures via structure-preserving signatures: Standard model security from simple assumptions. In *Annual Cryptology Conference*, pages 296–316. Springer, 2015.

[14] David Pointcheval and Olivier Sanders. Short randomizable signatures. In *Cryptographers Track at the RSA Conference*, pages 111–126. Springer, 2016.

[15] 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.

Appendix A

Formal definition of game sequence for full-anonymity proof

We will give a formal algorithmic description of the games in the game sequence used in the full-anonymity proof (Thm. 5.3). We will write down the original full-anonymity game from Def. 3.11 as G_0 in a more detailed way to make it easier to see the changes made in the following games.

$\mathsf{G}_0(\lambda)$

1 : $b \leftarrow_{\$} \{0,1\}$
2 : $BG := \mathsf{BGGen}_{\mathsf{SPS}}(\lambda)$
3 : $(\mathsf{pk}_{\mathsf{SPS}}, \mathsf{sk}_{\mathsf{SPS}}) \leftarrow_{\$} \mathsf{KGen}_{\mathsf{SPS}}(BG, l)$
4 : **for** $j \in [n]$
5 : $\quad \omega_j \leftarrow_{\$} \mathsf{coin}$
6 : $\quad (\mathsf{pk}_j, \mathsf{sk}_j, \tau_j) \leftarrow_{\$} \mathsf{TKGen}_{\mathsf{SFPK}}(\lambda, \omega_j)$
7 : $\quad \sigma_{\mathsf{SPS}}^{(j)} \leftarrow_{\$} \mathsf{Sign}_{\mathsf{SPS}}(\mathsf{sk}_{\mathsf{SPS}}, \mathsf{pk}_j)$
8 : $\mathsf{gpk} := (BG, \mathsf{pk}_{\mathsf{SPS}})$
9 : $\mathsf{gmsk} := ((\mathsf{pk}_j, \tau_j))_{j=1}^{n}$
10 : **for** $j \in [n]$
11 : $\quad \mathsf{gsk[j]} := (\mathsf{pk}_j, \mathsf{sk}_j, \sigma_{\mathsf{SPS}}^{(j)})$
12 : $\mathsf{gsk} := (\mathsf{gsk[j]})_{j=1}^{n}$
13 : $(\mathsf{St}, i_0, i_1, m) \leftarrow_{\$} \mathcal{A}^{\mathsf{Open}(\mathsf{gmsk},\cdot,\cdot)}(\mathsf{choose}, \mathsf{gpk}, \mathsf{gsk})$
14 : parse $\mathsf{gsk[i_b]} := (\mathsf{pk}_{i_b}, \mathsf{sk}_{i_b}, \sigma_{\mathsf{SPS}}^{(i_b)})$
15 : $r \leftarrow_{\$} \mathsf{coin}$
16 : $\mathsf{pk}'_{i_b} \leftarrow_{\$} \mathsf{ChgPK}_{\mathsf{SFPK}}(\mathsf{pk}_{i_b}, r)$
17 : $\mathsf{sk}'_{i_b} \leftarrow_{\$} \mathsf{ChgPK}_{\mathsf{SFPK}}(\mathsf{sk}_{i_b}, r)$
18 : $(\mathsf{pk}'_{i_b}, \sigma_{\mathsf{SPS}}^{(i_b)'}) \leftarrow_{\$} \mathsf{ChgRep}_{\mathsf{SPS}}(\mathsf{pk}_{i_b}, \sigma_{\mathsf{SPS}}^{(i_b)}, r, \mathsf{pk}_{\mathsf{SPS}})$
19 : $M := m || \sigma_{\mathsf{SPS}}^{(i_b)'} || \mathsf{pk}'_{i_b}$
20 : $\sigma_0 \leftarrow_{\$} \mathsf{Sign}_{\mathsf{SFPK}}(\mathsf{sk}'_{i_b}, M)$
21 : $\sigma_{ch} := (\mathsf{pk}'_{i_b}, \sigma_0, \sigma_{\mathsf{SPS}}^{(i_b)'})$
22 : $\tilde{b} \leftarrow_{\$} \mathcal{A}^{\mathsf{Open}(\mathsf{gmsk},\cdot,\cdot)}(\mathsf{guess}, \mathsf{St}, \sigma_{ch})$
23 : **if** $\mathcal{A}$ did not query Open-oracle with (m, σ_{ch}) in guess phase
24 : $\quad$ **then** return $\tilde{b} = b$
25 : **else** return 0

$\mathsf{G}_1(\lambda)$

1 : $b \leftarrow_\$ \{0,1\}$
2 : $BG := \mathsf{BGGen}_{\mathsf{SPS}}(\lambda)$
3 : $(\mathsf{pk}_{\mathsf{SPS}}, \mathsf{sk}_{\mathsf{SPS}}) \leftarrow_\$ \mathsf{KGen}_{\mathsf{SPS}}(BG, l)$
4 : **for** $j \in [n]$
5 : $\quad \omega_j \leftarrow_\$ \mathsf{coin}$
6 : $\quad (\mathsf{pk}_j, \mathsf{sk}_j, \tau_j) \leftarrow_\$ \mathsf{TKGen}_{\mathsf{SFPK}}(\lambda, \omega_j)$
7 : $\quad \sigma_{\mathsf{SPS}}^{(j)} \leftarrow_\$ \mathsf{Sign}_{\mathsf{SPS}}(\mathsf{sk}_{\mathsf{SPS}}, \mathsf{pk}_j)$
8 : $\mathsf{gpk} := (BG, \mathsf{pk}_{\mathsf{SPS}})$
9 : $\mathsf{gmsk} := ((\mathsf{pk}_j, \tau_j))_{j=1}^n$
10 : **for** $j \in [n]$
11 : $\quad \mathsf{gsk[j]} := (\mathsf{pk}_j, \mathsf{sk}_j, \sigma_{\mathsf{SPS}}^{(j)})$
12 : $\mathsf{gsk} := (\mathsf{gsk[j]})_{j=1}^n$
13 : $(\mathsf{St}, i_0, i_1, m) \leftarrow_\$ \mathcal{A}^{\mathsf{Open}(\mathsf{gmsk},\cdot,\cdot)}(\mathsf{choose}, \mathsf{gpk}, \mathsf{gsk})$
14 : parse $\mathsf{gsk[i_b]} := (\mathsf{pk}_{i_b}, \mathsf{sk}_{i_b}, \sigma_{\mathsf{SPS}}^{(i_b)})$
15 : $r \leftarrow_\$ \mathsf{coin}$
16 : $\mathsf{pk}'_{i_b} \leftarrow_\$ \mathsf{ChgPK}_{\mathsf{SFPK}}(\mathsf{pk}_{i_b}, r)$
17 : $\mathsf{sk}'_{i_b} \leftarrow_\$ \mathsf{ChgPK}_{\mathsf{SFPK}}(\mathsf{sk}_{i_b}, r)$
18 : $\sigma_{\mathsf{SPS}}^{(i_b)\prime} \leftarrow_\$ \mathsf{Sign}_{\mathsf{SPS}}(\mathsf{pk}'_{i_b}, \mathsf{sk}_{\mathsf{SPS}})$
19 : $M := m || \sigma_{\mathsf{SPS}}^{(i_b)\prime} || \mathsf{pk}'_{i_b}$
20 : $\sigma_0 \leftarrow_\$ \mathsf{Sign}_{\mathsf{SFPK}}(\mathsf{sk}'_{i_b}, M)$
21 : $\sigma_{ch} := (\mathsf{pk}'_{i_b}, \sigma_0, \sigma_{\mathsf{SPS}}^{(i_b)\prime})$
22 : $\tilde{b} \leftarrow_\$ \mathcal{A}^{\mathsf{Open}(\mathsf{gmsk},\cdot,\cdot)}(\mathsf{guess}, \mathsf{St}, \sigma_{ch})$
23 : **if** $\mathcal{A}$ did not query Open-oracle with (m, σ_{ch}) in guess phase
24 : $\quad$ **then** return $\tilde{b} = b$
25 : **else** return 0

$\mathsf{G}_2(\lambda)$

1 : $b \leftarrow\!\!\$\ \{0,1\}$
2 : $i \leftarrow\!\!\$\ [n]$
3 : $BG := \mathsf{BGGen}_{\mathsf{SPS}}(\lambda)$
4 : $(\mathsf{pk}_{\mathsf{SPS}}, \mathsf{sk}_{\mathsf{SPS}}) \leftarrow\!\!\$\ \mathsf{KGen}_{\mathsf{SPS}}(BG, l)$
5 : **for** $j \in [n]$
6 : $\quad \omega_j \leftarrow\!\!\$\ \mathsf{coin}$
7 : $\quad (\mathsf{pk}_j, \mathsf{sk}_j, \tau_j) \leftarrow\!\!\$\ \mathsf{TKGen}_{\mathsf{SFPK}}(\lambda, \omega_j)$
8 : $\quad \sigma_{\mathsf{SPS}}^{(j)} \leftarrow\!\!\$\ \mathsf{Sign}_{\mathsf{SPS}}(\mathsf{sk}_{\mathsf{SPS}}, \mathsf{pk}_j)$
9 : $\mathsf{gpk} := (BG, \mathsf{pk}_{\mathsf{SPS}})$
10 : $\mathsf{gmsk} := ((\mathsf{pk}_j, \tau_j))_{j=1}^n$
11 : **for** $j \in [n]$
12 : $\quad \mathsf{gsk[j]} := (\mathsf{pk}_j, \mathsf{sk}_j, \sigma_{\mathsf{SPS}}^{(j)})$
13 : $\mathsf{gsk} := (\mathsf{gsk[j]})_{j=1}^n$
14 : $(\mathsf{St}, i_0, i_1, m) \leftarrow\!\!\$\ \mathcal{A}^{\mathsf{Open}(\mathsf{gmsk},\cdot,\cdot)}(\mathsf{choose}, \mathsf{gpk}, \mathsf{gsk})$
15 : **if** $i \neq i_b$
16 : $\quad$ abort
17 : parse $\mathsf{gsk}[\mathsf{i_b}] := (\mathsf{pk}_{i_b}, \mathsf{sk}_{i_b}, \sigma_{\mathsf{SPS}}^{(i_b)})$
18 : $r \leftarrow\!\!\$\ \mathsf{coin}$
19 : $\mathsf{pk}'_{i_b} \leftarrow\!\!\$\ \mathsf{ChgPK}_{\mathsf{SFPK}}(\mathsf{pk}_{i_b}, r)$
20 : $\mathsf{sk}'_{i_b} \leftarrow\!\!\$\ \mathsf{ChgPK}_{\mathsf{SFPK}}(\mathsf{sk}_{i_b}, r)$
21 : $\sigma_{\mathsf{SPS}}^{(i_b)'} \leftarrow\!\!\$\ \mathsf{Sign}_{\mathsf{SPS}}(\mathsf{pk}'_{i_b}, \mathsf{sk}_{\mathsf{SPS}})$
22 : $M := m||\sigma_{\mathsf{SPS}}^{(i_b)'}||\mathsf{pk}'_{i_b}$
23 : $\sigma_0 \leftarrow\!\!\$\ \mathsf{Sign}_{\mathsf{SFPK}}(\mathsf{sk}'_{i_b}, M)$
24 : $\sigma_{ch} := (\mathsf{pk}'_{i_b}, \sigma_0, \sigma_{\mathsf{SPS}}^{(i_b)'})$
25 : $\tilde{b} \leftarrow\!\!\$\ \mathcal{A}^{\mathsf{Open}(\mathsf{gmsk},\cdot,\cdot)}(\mathsf{guess}, \mathsf{St}, \sigma_{ch})$
26 : **if** $\mathcal{A}$ did not query Open-oracle with (m, σ_{ch}) in guess phase
27 : $\quad$ **then** return $\tilde{b} = b$
28 : **else** return 0

$\mathsf{G}_3(\lambda)$

1 : $b \leftarrow\!\!\$\ \{0,1\}$
2 : $i \leftarrow\!\!\$\ [n]$
3 : $BG := \mathsf{BGGen}_{\mathsf{SPS}}(\lambda)$
4 : $(\mathsf{pk}_{\mathsf{SPS}}, \mathsf{sk}_{\mathsf{SPS}}) \leftarrow\!\!\$\ \mathsf{KGen}_{\mathsf{SPS}}(BG, l)$
5 : **for** $j \in [n]$
6 : $\quad \omega_j \leftarrow\!\!\$\ \mathsf{coin}$
7 : $\quad (\mathsf{pk}_j, \mathsf{sk}_j, \tau_j) \leftarrow\!\!\$\ \mathsf{TKGen}_{\mathsf{SFPK}}(\lambda, \omega_j)$
8 : $\quad \sigma^{(j)}_{\mathsf{SPS}} \leftarrow\!\!\$\ \mathsf{Sign}_{\mathsf{SPS}}(\mathsf{sk}_{\mathsf{SPS}}, \mathsf{pk}_j)$
9 : $\mathsf{gpk} := (BG, \mathsf{pk}_{\mathsf{SPS}})$
10 : $\mathsf{gmsk} := ((\mathsf{pk}_j, \tau_j))_{j=1}^{n}$
11 : **for** $j \in [n]$
12 : $\quad \mathsf{gsk[j]} := (\mathsf{pk}_j, \mathsf{sk}_j, \sigma^{(j)}_{\mathsf{SPS}})$
13 : $\mathsf{gsk} := (\mathsf{gsk[j]})_{j=1}^{n}$
14 : $(\mathsf{St}, i_0, i_1, m) \leftarrow\!\!\$\ \mathcal{A}^{\mathsf{Open}(\mathsf{gmsk},\cdot,\cdot)}(\mathsf{choose}, \mathsf{gpk}, \mathsf{gsk})$
15 : **if** $i \neq i_b$
16 : $\quad$ abort
17 : parse $\mathsf{gsk[i_b]} := (\mathsf{pk}_{i_b}, \mathsf{sk}_{i_b}, \sigma^{(i_b)}_{\mathsf{SPS}})$
18 : $r \leftarrow\!\!\$\ \mathsf{coin}$
19 : $\mathsf{pk}'_{i_b} \leftarrow\!\!\$\ \mathsf{ChgPK}_{\mathsf{SFPK}}(\mathsf{pk}_{i_b}, r)$
20 : $\mathsf{sk}'_{i_b} \leftarrow\!\!\$\ \mathsf{ChgPK}_{\mathsf{SFPK}}(\mathsf{sk}_{i_b}, r)$
21 : $\sigma^{(i_b)\prime}_{\mathsf{SPS}} \leftarrow\!\!\$\ \mathsf{Sign}_{\mathsf{SPS}}(\mathsf{pk}'_{i_b}, \mathsf{sk}_{\mathsf{SPS}})$
22 : $M := m||\sigma^{(i_b)\prime}_{\mathsf{SPS}}||\mathsf{pk}'_{i_b}$
23 : $\sigma_0 \leftarrow\!\!\$\ \mathsf{Sign}_{\mathsf{SFPK}}(\mathsf{sk}'_{i_b}, M)$
24 : $\sigma_{ch} := (\mathsf{pk}'_{i_b}, \sigma_0, \sigma^{(i_b)\prime}_{\mathsf{SPS}})$
25 : $\tilde{b} \leftarrow\!\!\$\ \mathcal{A}^{\mathsf{Open}(\mathsf{gmsk},\cdot,\cdot)}(\mathsf{guess}, \mathsf{St}, \sigma_{ch})$
26 : **if** $\mathcal{A}$ queried (m^*, σ^*) with $\mathsf{Open}(\mathsf{gmsk}, m^*, \sigma^*) = \perp$
27 : $\quad$ abort
28 : **if** $\mathcal{A}$ did not query Open-oracle with (m, σ_{ch}) in guess phase
29 : $\quad$ **then** return $\tilde{b} = b$
30 : **else** return 0

$\mathsf{G}_4(\lambda)$

$b \leftarrow\!\!\$\ \{0,1\}$
$i \leftarrow\!\!\$\ [n]$
$BG := \mathsf{BGGen}_{\mathsf{SPS}}(\lambda)$
$(\mathsf{pk}_{\mathsf{SPS}}, \mathsf{sk}_{\mathsf{SPS}}) \leftarrow\!\!\$\ \mathsf{KGen}_{\mathsf{SPS}}(BG, l)$
for $j \in [n]$
  $\omega_j \leftarrow\!\!\$\ \mathsf{coin}$
  if $j \neq i$
    $(\mathsf{pk}_j, \mathsf{sk}_j, \tau_j) \leftarrow\!\!\$\ \mathsf{TKGen}_{\mathsf{SFPK}}(\lambda, \omega_j)$
  else
    $(\mathsf{pk}_j, \mathsf{sk}_j) \leftarrow\!\!\$\ \mathsf{TKGen}_{\mathsf{SFPK}}(\lambda, \omega_j)$
  $\sigma_{\mathsf{SPS}}^{(j)} \leftarrow\!\!\$\ \mathsf{Sign}_{\mathsf{SPS}}(\mathsf{sk}_{\mathsf{SPS}}, \mathsf{pk}_j)$
$\mathsf{gpk} := (BG, \mathsf{pk}_{\mathsf{SPS}})$
$\mathsf{gmsk} := ((\mathsf{pk}_j, \tau_j))_{j=1}^n$
for $j \in [n]$
  $\mathsf{gsk[j]} := (\mathsf{pk}_j, \mathsf{sk}_j, \sigma_{\mathsf{SPS}}^{(j)})$
$\mathsf{gsk} := (\mathsf{gsk[j]})_{j=1}^n$
$(\mathsf{St}, i_0, i_1, m) \leftarrow\!\!\$\ \mathcal{A}^{\mathsf{Open}(\mathsf{gmsk},\cdot,\cdot)}(\mathsf{choose}, \mathsf{gpk}, \mathsf{gsk})$
if $i \neq i_b$
  abort
parse $\mathsf{gsk}[\mathsf{i_b}] := (\mathsf{pk}_{i_b}, \mathsf{sk}_{i_b}, \sigma_{\mathsf{SPS}}^{(i_b)})$
$r \leftarrow\!\!\$\ \mathsf{coin}$
$\mathsf{pk}'_{i_b} \leftarrow\!\!\$\ \mathsf{ChgPK}_{\mathsf{SFPK}}(\mathsf{pk}_{i_b}, r)$
$\mathsf{sk}'_{i_b} \leftarrow\!\!\$\ \mathsf{ChgPK}_{\mathsf{SFPK}}(\mathsf{sk}_{i_b}, r)$
$\sigma_{\mathsf{SPS}}^{(i_b)'} \leftarrow\!\!\$\ \mathsf{Sign}_{\mathsf{SPS}}(\mathsf{pk}'_{i_b}, \mathsf{sk}_{\mathsf{SPS}})$
$M := m||\sigma_{\mathsf{SPS}}^{(i_b)'}||\mathsf{pk}'_{i_b}$
$\sigma_0 \leftarrow\!\!\$\ \mathsf{Sign}_{\mathsf{SFPK}}(\mathsf{sk}'_{i_b}, M)$
$\sigma_{ch} := (\mathsf{pk}'_{i_b}, \sigma_0, \sigma_{\mathsf{SPS}}^{(i_b)'})$
$\tilde{b} \leftarrow\!\!\$\ \mathcal{A}^{\mathsf{Open}(\mathsf{gmsk},\cdot,\cdot)}(\mathsf{guess}, \mathsf{St}, \sigma_{ch})$
if $\mathcal{A}$ queried (m^*, σ^*) with $\mathsf{Open}(\mathsf{gmsk}, m^*, \sigma^*) = \perp$
  abort
if $\mathcal{A}$ did not query Open-oracle with (m, σ_{ch}) in guess phase
then return $\tilde{b} = b$
else return 0

$\mathsf{G}_5(\lambda)$

1 : $b \leftarrow_\$ \{0,1\}$
2 : $i \leftarrow_\$ [n]$
3 : $BG := \mathsf{BGGen}_{\mathsf{SPS}}(\lambda)$
4 : $(\mathsf{pk}_{\mathsf{SPS}}, \mathsf{sk}_{\mathsf{SPS}}) \leftarrow_\$ \mathsf{KGen}_{\mathsf{SPS}}(BG, l)$
5 : **for** $j \in [n]$
6 : $\quad \omega_j \leftarrow_\$ \mathsf{coin}$
7 : $\quad$ **if** $j \neq i$
8 : $\quad\quad (\mathsf{pk}_j, \mathsf{sk}_j, \tau_j) \leftarrow_\$ \mathsf{TKGen}_{\mathsf{SFPK}}(\lambda, \omega_j)$
9 : $\quad$ **else**
10 : $\quad\quad (\mathsf{pk}_j, \mathsf{sk}_j) \leftarrow_\$ \mathsf{TKGen}_{\mathsf{SFPK}}(\lambda, \omega_j)$
11 : $\quad \sigma^{(j)}_{\mathsf{SPS}} \leftarrow_\$ \mathsf{Sign}_{\mathsf{SPS}}(\mathsf{sk}_{\mathsf{SPS}}, \mathsf{pk}_j)$
12 : $\mathsf{gpk} := (BG, \mathsf{pk}_{\mathsf{SPS}})$
13 : $\mathsf{gmsk} := ((\mathsf{pk}_j, \tau_j))_{j=1}^n$
14 : **for** $j \in [n]$
15 : $\quad \mathsf{gsk[j]} := (\mathsf{pk}_j, \mathsf{sk}_j, \sigma^{(j)}_{\mathsf{SPS}})$
16 : $\mathsf{gsk} := (\mathsf{gsk[j]})_{j=1}^n$
17 : $(\mathsf{St}, i_0, i_1, m) \leftarrow_\$ \mathcal{A}^{\mathsf{Open}(\mathsf{gmsk},\cdot,\cdot)}(\mathsf{choose}, \mathsf{gpk}, \mathsf{gsk})$
18 : **if** $i \neq i_b$
19 : $\quad$ abort
20 : $\omega \leftarrow_\$ \mathsf{coin}$
21 : $(\mathsf{pk}, \mathsf{sk}) \leftarrow_\$ \mathsf{KGen}_{\mathsf{SFPK}}(\lambda, \omega)$
22 : $r \leftarrow_\$ \mathsf{coin}$
23 : $\mathsf{pk}' \leftarrow_\$ \mathsf{ChgPK}_{\mathsf{SFPK}}(\mathsf{pk}, r)$
24 : $\mathsf{sk}' \leftarrow_\$ \mathsf{ChgPK}_{\mathsf{SFPK}}(\mathsf{sk}, r)$
25 : $\sigma'_{\mathsf{SPS}} \leftarrow_\$ \mathsf{Sign}_{\mathsf{SPS}}(\mathsf{pk}', \mathsf{sk}_{\mathsf{SPS}})$
26 : $M := m || \sigma'_{\mathsf{SPS}} || \mathsf{pk}'$
27 : $\sigma_0 \leftarrow_\$ \mathsf{Sign}_{\mathsf{SFPK}}(\mathsf{sk}', M)$
28 : $\sigma_{ch} := (\mathsf{pk}', \sigma_0, \sigma'_{\mathsf{SPS}})$
29 : $\tilde{b} \leftarrow_\$ \mathcal{A}^{\mathsf{Open}(\mathsf{gmsk},\cdot,\cdot)}(\mathsf{guess}, \mathsf{St}, \sigma_{ch})$
30 : **if** $\mathcal{A}$ queried (m^*, σ^*) with $\mathsf{Open}(\mathsf{gmsk}, m^*, \sigma^*) = \perp$
31 : $\quad$ abort
32 : **if** $\mathcal{A}$ did not query Open-oracle with (m, σ_{ch}) in guess phase
33 : $\quad$ **then** return $\tilde{b} = b$
34 : **else** return 0