A preliminary version of this paper appears in *Topics in Cryptology – CT-RSA '05*, Lecture Notes in Computer Science Vol. ??, A. Menezes ed., Springer-Verlag, 2005. This is the full version.

Foundations of Group Signatures: The Case of Dynamic Groups

Mihir Bellare* Haixia Shi† Chong Zhang‡

June 2004

Abstract

Recently, a first step toward establishing foundations for group signatures was taken [5], with a treatment of the case where the group is static. However the bulk of existing practical schemes and applications are for dynamic groups, and these involve important new elements and security issues. This paper treats this case, providing foundations for dynamic group signatures, in the form of a model, strong formal definitions of security, and a construction proven secure under general assumptions. We believe this is an important and useful step because it helps bridge the gap between [5] and the previous practical work, and delivers a basis on which existing practical schemes may in future be evaluated or proven secure.

*Dept. of Computer Science & Engineering, University of California at San Diego, 9500 Gilman Drive, La Jolla, California 92093, USA. E-Mail: `mihir@cs.ucsd.edu`. URL: `http://www-cse.ucsd.edu/users/mihir`. Supported in part by NSF grants CCR-0098123, ANR-0129617 and CCR-0208842, ANR-0129617 and an IBM Faculty Partnership Development Award.

†Dept. of Computer Science & Engineering, University of California at San Diego, 9500 Gilman Drive, La Jolla, California 92093, USA. E-Mail: `hashi@cs.ucsd.edu`. URL: `http://www-cse.ucsd.edu/users/hashi`. Supported in part by above-mentioned grants of first author.

‡Dept. of Computer Science & Engineering, University of California at San Diego, 9500 Gilman Drive, La Jolla, California 92093, USA. E-Mail: `c2zhang@cs.ucsd.edu`. URL: `http://www-cse.ucsd.edu/users/c2zhang`. Supported in part by above-mentioned grants of first author.

Contents

1 Introduction

The purpose of foundational work is to provide strong, formal definitions of security for cryptographic primitives, thereby enabling one to unambiguously assess and prove the security of constructs and their use in applications, and then prove the existence of schemes meeting the given definitions. As evidenced by the development of the foundations of encryption [21, 25, 20, 26, 28, 17], however, this program can require several steps and considerable effort.

This paper takes the next step in the foundational effort in group signatures begun by [5]. Below we provide some background and then discuss our contributions.

1.1 Background and motivation

Group signatures. The setting, introduced by Chaum and Van Heyst [15], is of a group of entities, each having its own private signing key, using which it can produce signatures on behalf of the group, meaning verifiable under a single public verification key associated to the group as a whole. The basic security requirements are that the identity of the group member producing a particular signature not be discernible from this signature (anonymity), except to an authority possessing a special "opening" key (traceability).

With time, more security requirements were added, including unlinkability, unforgeability, collusion resistance [4], exculpability [4], and framing resistance [16]. Many practical schemes were presented, some with claims of proven security in the random oracle model [1]. However, it is often unclear what the schemes or claimed proofs in these works actually deliver in terms of security guarantees, due largely to the fact that the requirements are informal and sometimes ambiguous, not precisely specifying adversary capabilities and goals. It would be beneficial in this context to have proper foundations, meaning strong formal definitions and rigorously proven-secure schemes.

Foundations for static groups. The first step toward this end was taken by [5], who consider the case where the group is *static*. In their setting, the number of group members and their identities are fixed and frozen in the setup phase, where a trusted entity chooses not only the group public key and an opening key for the opening authority, but also, for each group member, chooses a signing key and hands it to the member in question. Within this framework, they formalize two (strong) security requirements that they call full-anonymity and full-traceability, and show that these imply all the informal existing requirements in the previous literature. They then present a static group signature scheme shown to meet these requirements, assuming the existence of trapdoor permutations.

Dynamic groups. However, static groups limit applications of group signatures, since they do not allow one to add members to the group with time. They also require an uncomfortably high degree of trust in the party performing setup, since the latter knows the signing keys of all members and can thus frame any group member. These limitations were in fact recognized early in the development of the area, and the practical literature has from the start focused on the case where the group is *dynamic*. In this setting, neither the number nor the identities of group members are fixed or known in the setup phase, which now consists of the trusted entity choosing only a group public key and a key for the authority. An entity can join the group, and obtain a private signing key at any time, by engaging in an appropriate join protocol with the authority.

Closing the gap. We thus have the following gap: foundations have been provided for the static case [5], but the bulk of applications and existing practical schemes are for the dynamic case [15, 16, 11, 14, 27, 13, 4, 3, 1]. Since the ultimate goal is clearly to have proven secure schemes in settings suitable for applications, it is important to bridge the above-mentioned gap by providing foundations for dynamic group signatures.

However, an extension of the existing treatment of static groups [5] to the dynamic case does not seem to be immediate. Dynamic groups are more complex, bringing in new elements, security requirements and issues. A dedicated and detailed treatment is required to resolve the numerous existing issues and ambiguities. This paper provides such a treatment.

1.2 Model and definitions for the dynamic group setting

The first contribution of this paper is to provide a model and strong, formal definitions of a small number of key security requirements for dynamic group signatures that, in keeping with [5], are then shown to imply the large number of existing informal requirements.

Selected features. We highlight a few important features of the model and definitions:

- Two authorities. As suggested in some previous works, we separate the authority into two, an opener (who can open signatures) and an issuer (who interacts with a user to issue the latter a signing key). Each has its own secret key. This provides more security (compared to having a single authority) in the face of the possibility that authorities can be dishonest.
- Trust levels. We consider different levels of trust in each authority, namely that it may be uncorrupt (trusted), partially corrupt (its secret key is available to the adversary but it does not deviate from its prescribed program) or fully corrupt (the adversary controls it entirely, so that it may not follow its program). In order to protect group members against dishonest authorities to the maximum extent possible, we formulate security requirements to require the lowest possible level of trust in each authority.
- Three key requirements. We formulate three key requirements, namely anonymity, traceability and non-frameability. The levels of trust for each authority for each requirement are summarized in Figure 1, and, as we explain in Section 4, are the minimum possible in each case. (In the static setting, the single full-traceability requirement covered both traceability and non-frameability [5]. We separate them here because we can ask for and achieve non-frameability with lower levels of trust in the authorities than traceability.)
- PKI. We assume that each group member or potential group member has a personal public key, established and certified, for example by a PKI, independently of any group authority, so that it has a means to sign information, using a matching personal private key that it retains. This is *necessary* in order for group members to protect themselves from being framed by a partially or fully corrupt issuer, and makes explicit what were called "long-term credentials" in [1].
- Publicly verifiable proofs of opening. In order to be protected against a fully corrupt opener, the opener is required to accompany any claim that a particular identity produced a particular signature with a publicly verifiable proof to this effect (cf. [13]).
- Concurrent join protocols. In an Internet-based system, we would expect that many entities may concurrently engage in the join protocol with the issuer. Our model captures this by allowing the adversary to schedule all message delivery in any number of concurrent join sessions.

Definitional approach. In order to provide clear, succinct yet formal definitions, and also allow for easy additions of more definitions, we take a modular approach that follows the paradigm of [7]. We first specify a model that consists of defining various oracles that provide the adversary with various attack capabilities. Each of the formal definitions then provides the adversary with some appropriate subset of these oracles, depending on the type of attack capabilities the definition wishes to give the adversary.

As research in this area has shown, requirements for group signatures tend to grow and evolve with time (cf. [4, 16, 24]). The benefit of the modular definitional approach we employ here is that it

Requirement	Opener	Issuer
Anonymity	uncorrupt	fully corrupt
Traceability	partially corrupt	uncorrupt
Non-frameability	fully corrupt	fully corrupt

Figure 1: Levels of trust in authorities for each of our three security requirements. In each case, these are the lowest levels of trust achievable.

is easy to add new requirements, first by introducing new oracles to capture new attack capabilities if necessary, and then by formulating new definitions in terms of adversaries that call on the old and new oracles.

1.3 A construction of a secure dynamic group signature scheme

Given the stringency of our security requirements, the first and most basic question that should be considered is whether a secure dynamic group signature scheme even exists, and, if so, under what assumptions its existence can be proved. Although the setting and requirements for dynamic groups are more complex and demanding than for static groups, we can prove the existence of a secure dynamic group signature scheme under the same assumptions as used to prove the existence of a secure static group signature scheme [5], namely the existence of trapdoor permutations. As is not uncommon with foundational schemes, ours is polynomial-time but not efficient, and should be taken as a proof of concept only.

The construction uses as building blocks the following: trapdoor permutation based public-key encryption schemes secure against chosen-ciphertext attack [17], trapdoor permutation based (ordinary) digital signature schemes secure against chosen-message attack [6], and trapdoor permutation based simulation-sound adaptive non-interactive zero-knowledge (NIZK) proofs for NP [29]. We provide a way to define a group public key, keys for the two authorities, and a join protocol so that the private signing key of any group member, as well as the signature created, have essentially the same format as in the scheme of [5], thereby enabling us to build on the latter. We then augment the opening algorithm to also produce NIZK proofs of its claims, and define a judge algorithm to check such proofs. To provide traceability and non-frameability, the join protocol requires, on the one hand, that the group member provide the issuer with a signature (relative to the personal public key that the group member has via the PKI) of some information related to the private signing key it is issued. (This signature is stored by the issuer in the registration table and can later be accessed by the opener.) However, it also ensures that the issuer does not know the private signing key of the group member. We note that in our scheme, the length of signatures and the size of keys do not depend on the number of members in the group. (The registration table has size proportional to the number of users but is not considered part of the keys.)

We remark that the join protocol is simple and uses no zero-knowledge (ZK) proofs. This is important because it facilitates showing security under arbitrary concurrent executions. But it may be surprising because the join protocols in practical schemes such as that of [1] use ZK proofs even though the security requirements there are milder than in our case.

1.4 Discussion and related work

We do not consider revocation of group members.[1] Different solutions tend to require or depend on different model elements [10, 2, 30] and we believe it is restrictive to pin down features geared toward some solution as part of what is supposed to be a general model. However, as noted above, our model has an extensible format, and can be extended in different ways to accommodate different revocation approaches and requirements.

In specifying our model and definitions we have built on numerous elements of previous works, including informal discussions in [5] about extensions to the dynamic setting. We remark however that we were not always able to follow the suggestions of the latter. For example they suggested that a proof of opening could consist of the coins underlying a certain ciphertext in the signature. But the decryption algorithms of existing trapdoor permutation based, chosen-ciphertext secure encryption schemes [17, 29] do not recover the coins, and, even if one had a scheme that did, one would need to know whether it was secure against a stronger type of chosen-ciphertext attacks in which the decryption oracle returns not just the message but also the coins underlying a given ciphertext. Instead, we use NIZK proofs.

Our model assumes that the issuer and opener are provided their keys by a trusted initialization process that chooses these keys along with the group public key. Naturally, if so desired, such a process may be implemented by a secure distributed computation protocol in which the authorities jointly compute their keys and the group public key. This would enable one to dispense with the trusted initialization.

There may be schemes or setting in which there is a single authority that plays the roles of both issuer and opener, rather than there being two separate authorities as in our model. This case is simpler than the one we consider, and our definitions and scheme can easily be "dropped down" to handle it. Of course, the security achieved will be weaker.

Our model captures the functionality of current efficient proposals for group signature schemes, in particular that of [1]. Although we do not know whether their scheme can be proven secure in our model, providing the model at least enables one to address this question rigorously in the future.

Camenisch and Lysyanskaya [12] present simulation-based definitions for identity-escrow schemes with appointed verifiers, which are related to group signature schemes. We believe however that models like that of [5] and ours are easier to use.

In concurrent and independent work, Kiayis, Tsiounis and Yung [24] introduce an extension of group signatures called traceable signatures. However, in the dynamic group signature setting, their model is different from ours. In particular, they consider a single authority rather than separate issuing and opening authorities. This means that they cannot consider authority behavior that is as adversarial as the ones we consider, namely fully corrupt, and, in some cases, even partially corrupt authorities. Not only does this mean their requirements are weaker than ours, but also this is where most of the novel issues, as compared with [5], arise. We also note that their model does not include a PKI, yet some such structure would appear to be required to realize certain assumptions they make. (Namely that the authority is not given the power to modify transcripts of the join protocol in non-frameability).

Finally we note that the traceable signature scheme of [24] is in the random-oracle model, being derived as the Fiat-Shamir transform [18] of a traceable identification scheme. Note that it may be *impossible* to "implement" the random oracle of the Fiat-Shamir transform with a "real" function in a way that results in a secure real-world scheme (Goldwasser and Tauman [23]). In contrast our scheme is in the standard model.

[1] Our terminology may thus be misleading. In some previous works, what we are considering are called partially dynamic groups rather than dynamic groups. The term monotonically growing groups has also been suggested.

2 Notation

We let $\mathbb{N} = \{1, 2, 3, \ldots\}$ be the set of *positive* integers. If x is a string, then $|x|$ denotes its length, while if S is a set then $|S|$ denotes its size. The empty string is denoted by ε. If $k \in \mathbb{N}$ then 1^k denotes the string of k ones. If n is an integer then $[n] = \{1, \ldots, n\}$. If S is a set then $s \stackrel{\$}{\leftarrow} S$ denotes the operation of picking an element s of S uniformly at random.

Unless otherwise indicated, algorithms are randomized. We write $A(x, y, \ldots)$ to indicate that A is an algorithm with inputs $x, y, \ldots,$, and by $z \stackrel{\$}{\leftarrow} A(x, y, \ldots)$ we denote the operation of running A with inputs $x, y, \ldots$ and letting z be the output. We write $A(x, y, \ldots : \mathcal{O}_1, \mathcal{O}_2, \ldots)$ to indicate that A is an algorithm with inputs $x, y, \ldots$ and access to oracles $\mathcal{O}_1, \mathcal{O}_2, \ldots$, and by $z \stackrel{\$}{\leftarrow} A(x, y, \ldots : \mathcal{O}_1, \mathcal{O}_2, \ldots)$ we denote the operation of running A with inputs $x, y, \ldots$ and access to oracles $\mathcal{O}_1, \mathcal{O}_2, \ldots$, and letting z be the output.

3 A model for dynamic group signature schemes

Here we provide a model in which definitions can later be formulated. We begin with a discussion of the syntax, namely the algorithms that constitute a dynamic group signature scheme.

ALGORITHMS AND THEIR USAGE. Involved in a group signature scheme are a trusted party for initial key generation, an authority called the issuer, an authority called the opener, and a body of users, each with a unique identity $i \in \mathbb{N}$, that may become group members. The scheme is specified as a tuple $\mathcal{GS} = (\mathsf{GKg}, \mathsf{UKg}, \mathsf{Join}, \mathsf{Iss}, \mathsf{GSig}, \mathsf{GVf}, \mathsf{Open}, \mathsf{Judge})$ of polynomial-time algorithms whose intended usage and functionality are as follows. Throughout, $k \in \mathbb{N}$ denotes the security parameter.

GKg– In a setup phase, the trusted party runs the *group-key generation* algorithm GKg on input 1^k to obtain a triple (gpk, ik, ok). The *issuer key* ik is provided to the issuer, and the *opening key* ok is provided to the opener. The *group public key* gpk, whose possession enables signature verification, is made public.

UKg– A user that wants to be a group member should begin by running the *user-key generation* algorithm UKg on input 1^k to obtain a *personal public and private key pair* $(\boldsymbol{upk}[i], \boldsymbol{usk}[i])$. We assume that the table $\boldsymbol{upk}$ is public. (Meaning, anyone can obtain an authentic copy of the personal public key of any user. This might be implemented via a PKI.)

$\mathsf{Join}, \mathsf{Iss}$– Once a user has its personal key pair, it can join the group by engaging in a *group-joining protocol* with the issuer. The *interactive algorithms* $\mathsf{Join}, \mathsf{Iss}$ implement, respectively, the user's and issuer's sides of this interaction. Each takes input an incoming message (this is ε if the party is initiating the interaction) and a current state, and returns an outgoing message, an updated state, and a decision which is one of $\mathsf{accept}, \mathsf{reject}, \mathsf{cont}$. The communication is assumed to take place over secure (i.e. private and authenticated) channels, and we assume the user sends the first message. If the issuer accepts, it makes an entry for i, denoted $\boldsymbol{reg}[i]$, in its *registration table* $\boldsymbol{reg}$, the contents of this entry being the final state output by Iss. If i accepts, the final state output by Join is its private signing key, denoted $\boldsymbol{gsk}[i]$.

GSig– A group member i, in possession of its signing key $\boldsymbol{gsk}[i]$, can apply the *group signing* algorithm GSig to $\boldsymbol{gsk}[i]$ and a message $m \in \{0, 1\}^*$ to obtain a quantity called a signature on m.

GVf– Anyone in possession of the group public key gpk can run the deterministic *group signature verification* algorithm GVf on inputs gpk, a message m, and a candidate signature σ for m, to obtain a bit. We say that σ is a *valid* signature of m with respect to gpk if this bit is one.

Open– The opener, who has read-access to the registration table $\boldsymbol{reg}$ being populated by the issuer, can apply the deterministic *opening* algorithm Open to its opening key ok, the registration table $\boldsymbol{reg}$, a message m, and a valid signature σ of m under gpk. The algorithm returns a pair (i, τ), where $i \geq 0$ is an integer. In case $i \geq 1$, the algorithm is claiming that the group member with identity i produced σ, and in case $i = 0$, it is claiming that no group member produced σ. In the former case, τ is a proof of this claim that can be verified via the Judge algorithm.

Judge– The deterministic *judge* algorithm Judge takes inputs the group public key gpk, an integer $j \geq 1$, the public key $\boldsymbol{upk}[j]$ of the entity with identity j (this is ε if this entity has no public key), a message m, a valid signature σ of m, and a proof-string τ. It aims to check that τ is a proof that j produced σ. We note that the judge will base its verification on the public key of j.

THE ORACLES. The correctness and security definitions will be formulated via experiments in which an adversary's attack capabilities are modeled by providing it access to certain oracles. We now introduce the oracles that we will need. (Different experiments will provide the adversary with different subsets of this set of oracles.)

The oracles are specified in Figure 2 and explained below. It is assumed that the overlying experiment has run GKg on input 1^k to obtain keys gpk, ik, ok that are used by the oracles. It is also assumed that this experiment maintains the following global variables which are manipulated by the oracles: a set HU of honest users; a set CU of corrupted users; a set GSet of message-signature pairs; a table $\boldsymbol{upk}$ such that $\boldsymbol{upk}[i]$ contains the public key of $i \in \mathbb{N}$; a table $\boldsymbol{reg}$ such that $\boldsymbol{reg}[i]$ contains the registration information of group member i. The sets $\mathrm{HU}, \mathrm{CU}, \mathrm{GSet}$ are assumed initially empty, and all entries of the tables $\boldsymbol{upk}, \boldsymbol{reg}$ are assumed initially to be ε. Randomized oracles or algorithms use fresh coins upon each invocation unless otherwise indicated.

$\mathsf{AddU}(\cdot)$– By calling this *add user* oracle with argument an identity $i \in \mathbb{N}$, the adversary can add i to the group as an honest user. The oracle adds i to the set HU of honest users, and picks a personal public and private key pair $(\boldsymbol{upk}[i], \boldsymbol{usk}[i])$ for i. It then executes the group-joining protocol by running Join (on behalf of i, initialized with $gpk, \boldsymbol{upk}[i], \boldsymbol{usk}[i]$) and Iss (on behalf of the issuer, initialized with $gpk, ik, i, \boldsymbol{upk}[i]$). When Iss accepts, its final state is recorded as entry $\boldsymbol{reg}[i]$ in the registration table. When Join accepts, its final state is recorded as the private signing key $\boldsymbol{gsk}[i]$ of i. The calling adversary is returned $upk[i]$.

$\mathsf{CrptU}(\cdot, \cdot)$– By calling this *corrupt user* oracle with arguments an identity $i \in \mathbb{N}$ and a string upk, the adversary can corrupt user i and set its personal public key $\boldsymbol{upk}[i]$ to the value upk chosen by the adversary. The oracle initializes the issuer's state in anticipation of a group-joining protocol with i.

$\mathsf{SndToI}(\cdot, \cdot)$– Having corrupted user i, the adversary can use this *send to issuer* oracle to engage in a group-joining protocol with the honest, Iss-executing issuer, itself playing the role of i and not necessarily executing the interactive algorithm Join prescribed for an honest user. The adversary provides the oracle with i and a message M_{in} to be sent to the issuer. The oracle, which maintains the issuer's state (the latter having been initialized by an earlier call to $\mathsf{CrptU}(i, \cdot)$), computes a response as per Iss, returns the outgoing message to the adversary, and sets entry $\boldsymbol{reg}[i]$ of the registration table to Iss's final state if the latter accepts.

$\mathsf{SndToU}(\cdot, \cdot)$– In some definitions we will want to consider an adversary that has corrupted the issuer. The *send to user* oracle $\mathsf{SndToU}(\cdot, \cdot)$ can be used by such an adversary to engage in a group-joining protocol with an honest, Join-executing user, itself playing the role of the issuer and not necessarily executing the interactive algorithm Iss prescribed for the honest issuer. The adversary provides the oracle with i and a message M_{in} to be sent to i. The oracle maintains the state of user i, initializing this the first time it is called by choosing a personal public and private key pair for i, computes a

$\mathsf{AddU}(i)$

If $i \in \mathrm{CU}$ or $i \in \mathrm{HU}$ then return ε
$\mathrm{HU} \leftarrow \mathrm{HU} \cup \{i\}$
$dec^i \leftarrow \mathsf{cont}$; $\boldsymbol{gsk}[i] \leftarrow \varepsilon$
$(\boldsymbol{upk}[i], \boldsymbol{usk}[i]) \xleftarrow{\$} \mathsf{UKg}(1^k)$
$\mathrm{St}^i_{jn} \leftarrow (gpk, \boldsymbol{upk}[i], \boldsymbol{usk}[i])$
$\mathrm{St}^i_{iss} \leftarrow (gpk, ik, i, \boldsymbol{upk}[i])$; $M_{jn} \leftarrow \varepsilon$
$(\mathrm{St}^i_{jn}, M_{iss}, dec^i) \leftarrow \mathsf{Join}(\mathrm{St}^i_{jn}, M_{jn})$
While $dec^i = \mathsf{cont}$ do
  $(\mathrm{St}^i_{iss}, M_{jn}, dec^i) \leftarrow \mathsf{Iss}(\mathrm{St}^i_{iss}, M_{iss}, dec^i)$
  If $dec^i = \mathsf{accept}$ then $\boldsymbol{reg}[i] \leftarrow \mathrm{St}^i_{iss}$
  $(\mathrm{St}^i_{jn}, M_{iss}, dec^i) \leftarrow \mathsf{Join}(\mathrm{St}^i_{jn}, M_{jn})$
Endwhile
$\boldsymbol{gsk}[i] \leftarrow \mathrm{St}^i_{jn}$
Return $\boldsymbol{upk}[i]$

$\mathsf{SndToI}(i, M_{in})$

If $i \notin \mathrm{CU}$ then return ε
$(\mathrm{St}^i_{iss}, M_{out}, dec^i) \leftarrow \mathsf{Iss}(\mathrm{St}^i_{iss}, M_{in}, dec^i)$
If $dec^i = \mathsf{accept}$ then $\boldsymbol{reg}[i] \leftarrow \mathrm{St}^i_{iss}$
Return M_{out}

$\mathsf{SndToU}(i, M_{in})$

If $i \notin \mathrm{HU}$ then
  $\mathrm{HU} \leftarrow \mathrm{HU} \cup \{i\}$
  $(\boldsymbol{upk}[i], \boldsymbol{usk}[i]) \xleftarrow{\$} \mathsf{UKg}(1^k)$
  $\boldsymbol{gsk}[i] \leftarrow \varepsilon$; $M_{in} \leftarrow \varepsilon$
  $\mathrm{St}^i_{jn} \leftarrow (gpk, \boldsymbol{upk}[i], \boldsymbol{usk}[i])$
$(\mathrm{St}^i_{jn}, M_{out}, dec) \leftarrow \mathsf{Join}(\mathrm{St}^i_{jn}, M_{in})$;
If $dec = \mathsf{accept}$ then $\boldsymbol{gsk}[i] \leftarrow \mathrm{St}^i_{jn}$
Return (M_{out}, dec)

$\mathsf{CrptU}(i, upk)$

If $i \in \mathrm{HU} \cup \mathrm{CU}$ then return ε
$\mathrm{CU} \leftarrow \mathrm{CU} \cup \{i\}$
$\boldsymbol{upk}[i] \leftarrow upk$
$dec^i \leftarrow \mathsf{cont}$
$\mathrm{St}^i_{iss} \leftarrow (gpk, ik, i, \boldsymbol{upk}[i])$
Return 1

$\mathsf{USK}(i)$

Return $(\boldsymbol{gsk}[i], \boldsymbol{usk}[i])$

$\mathsf{RReg}(i)$

Return $\boldsymbol{reg}[i]$

$\mathsf{WReg}(i, \rho)$

$\boldsymbol{reg}[i] \leftarrow \rho$

$\mathsf{Open}(m, \sigma)$

If $(m, \sigma) \in \mathrm{GSet}$ then return $\perp$
Return $\mathsf{Open}(gpk, ok, \boldsymbol{reg}, m, \sigma)$

$\mathsf{GSig}(i, m)$

If $i \notin \mathrm{HU}$ then return $\perp$
If $\boldsymbol{gsk}[i] = \varepsilon$ then return $\perp$
Else return $\mathsf{GSig}(gpk, \boldsymbol{gsk}[i], m)$

$\mathsf{Ch}_b(i_0, i_1, m)$

If $i_0 \notin \mathrm{HU}$ or $i_1 \notin \mathrm{HU}$ then
  return $\perp$
If $\boldsymbol{gsk}[i_0] = \varepsilon$ or $\boldsymbol{gsk}[i_1] = \varepsilon$ then
  return $\perp$
$\sigma \leftarrow \mathsf{GSig}(gpk, \boldsymbol{gsk}[i_b], m)$
$\mathrm{GSet} \leftarrow \mathrm{GSet} \cup \{(m, \sigma)\}$
Return σ

Figure 2: Oracles provided to adversaries in the experiments of Figure 3.

---

response as per Join, returns the outgoing message to the adversary, and sets the private signing of i to Join's final state if the latter accepts.

$\mathsf{USK}(\cdot)$– The adversary can call this *user secret keys* oracle with argument the identity $i \in \mathbb{N}$ of a user to expose both the private signing key $\boldsymbol{gsk}[i]$ and the personal private key $\boldsymbol{usk}[i]$ of this user.

$\mathsf{RReg}(\cdot)$– The adversary can read the contents of entry i of the registration table $\boldsymbol{reg}$ by calling this *read registration table* oracle with argument $i \in \mathbb{N}$.

$\mathsf{WReg}(\cdot, \cdot)$– In some definitions we will allow the adversary to write/modify the contents of entry i of the registration table $\boldsymbol{reg}$ by calling this *write registration table* oracle with argument $i \in \mathbb{N}$.

$\mathsf{GSig}(\cdot,\cdot)$– A *signing* oracle, enabling the adversary to specify the identity i of a user and a message m, and obtain the signature of m under the private signing key $\boldsymbol{gsk}[i]$ of i, as long as i is an honest user whose private signing key is defined.

$\mathsf{Ch}(b,\cdot,\cdot,\cdot)$– A *challenge* oracle provided to an adversary attacking anonymity, and depending on a challenge bit b set by the overlying experiment. The adversary provides a pair i_0, i_1 of identities and a message m, and obtains the signature of m under the private signing key of i_b, as long as both i_0, i_1 are honest users with defined private signing keys. The oracle records the message-signature pair in GSet to ensure that the adversary does not later call the opening oracle on it.

$\mathsf{Open}(\cdot,\cdot)$– The adversary can call this *opening* oracle with arguments a message m and signature σ to obtain the output of the opening algorithm on m, σ, computed under the opener's key ok, as long as σ was not previously returned in response to a query to $\mathsf{Ch}(b,\cdot,\cdot,\cdot)$.

Remarks. We are assuming the existence of a secure (private and authentic) channel between any prospective group member and the issuer, as in [1]. The privacy assumption is reflected in the fact that the adversary is not provided the transcript of an interaction generated by the $\mathsf{AddU}(\cdot)$ oracle. The authenticity assumption is reflected in the fact that a party is initialized with the correct identity and personal public key of its partner if relevant. (When the issuer is fully corrupted, reflected by the adversary having a $\mathsf{SndToU}(\cdot,\cdot)$ oracle, the adversary does get the transcript of the communication, via its oracle queries and answers.) We note however that the secure channels assumption is made more for simplicity than anything else, and protocols are easily modified to avoid it.

4 Notions of correctness and security

Here we provide the definitions of correctness and security of a dynamic group signature scheme, based on the model of an adversary with oracles introduced above. We begin with correctness and then define three security requirements: anonymity, traceability and non-frameability.

4.1 Correctness

The correctness condition pertains to signatures generated by honest group members, and asks the following: the signature should be valid; the opening algorithm, given the message and signature, should correctly identify the signer; the proof returned by the opening algorithm should be accepted by the judge. Formalizing these conditions in the dynamic group setting is more involved than formalizing them in a static setting in that these conditions must hold for all honest users under any "schedule" under which these users join the group. Accordingly, we formalize correctness via an experiment involving an adversary. To dynamic group signature scheme $\mathcal{GS}$, any adversary A and any $k \in \mathbb{N}$ we associate the experiment $\mathbf{Exp}^{\mathrm{corr}}_{\mathcal{GS},A}(k)$ depicted in Figure 3. We let

$$\mathbf{Adv}^{\mathrm{corr}}_{\mathcal{GS},A}(k) \;=\; \Pr\left[\,\mathbf{Exp}^{\mathrm{corr}}_{\mathcal{GS},A}(k) = 1\,\right] .$$

We say that dynamic group signature scheme $\mathcal{GS}$ is *correct* if $\mathbf{Adv}^{\mathrm{corr}}_{\mathcal{GS},A}(k) = 0$ for any adversary A and any $k \in \mathbb{N}$. Note that the adversary is not computationally restricted.

4.2 Anonymity

Formal definition. To dynamic group signature scheme $\mathcal{GS}$, any adversary A, a bit $b \in \{0,1\}$ and any $k \in \mathbb{N}$ we associate the experiment $\mathbf{Exp}^{\mathrm{anon}\text{-}b}_{\mathcal{GS},A}(k)$ depicted in Figure 3. We let

$$\mathbf{Adv}^{\mathrm{anon}}_{\mathcal{GS},A}(k) \;=\; \Pr\left[\,\mathbf{Exp}^{\mathrm{anon}\text{-}1}_{\mathcal{GS},A}(k) = 1\,\right] - \Pr\left[\,\mathbf{Exp}^{\mathrm{anon}\text{-}0}_{\mathcal{GS},A}(k) = 1\,\right] .$$

Experiment $\mathbf{Exp}^{\mathrm{corr}}_{\mathcal{GS},A}(k)$

$(gpk, ik, ok) \xleftarrow{\$} \mathsf{GKg}(1^k)$; CU $\leftarrow \emptyset$; HU $\leftarrow \emptyset$; $(i, m) \xleftarrow{\$} A(gpk \ :\ \mathsf{AddU}(\cdot), \mathsf{RReg}(\cdot))$

If $i \notin$ HU then return 0 ; If $\boldsymbol{gsk}[i] = \varepsilon$ then return 0

$\sigma \leftarrow \mathsf{GSig}(gpk, \boldsymbol{gsk}[i], m)$; If $\mathsf{GVf}(gpk, m, \sigma) = 0$ then return 1

$(j, \tau) \leftarrow \mathsf{Open}(gpk, ok, \boldsymbol{reg}, m, \sigma)$; If $i \neq j$ then return 1

If $\mathsf{Judge}(gpk, i, \boldsymbol{upk}[i], m, \sigma, \tau) = 0$ then return 1 else return 0

Experiment $\mathbf{Exp}^{\mathrm{anon}\text{-}b}_{\mathcal{GS},A}(k)$ // $b \in \{0, 1\}$

$(gpk, ik, ok) \xleftarrow{\$} \mathsf{GKg}(1^k)$; CU $\leftarrow \emptyset$; HU $\leftarrow \emptyset$; GSet $\leftarrow \emptyset$

$d \xleftarrow{\$} A(gpk, ik \ :\ \mathsf{Ch}(b, \cdot, \cdot, \cdot), \mathsf{Open}(\cdot, \cdot), \mathsf{SndToU}(\cdot, \cdot), \mathsf{WReg}(\cdot, \cdot), \mathsf{USK}(\cdot), \mathsf{CrptU}(\cdot, \cdot))$

Return d

Experiment $\mathbf{Exp}^{\mathrm{trace}}_{\mathcal{GS},A}(k)$

$(gpk, ik, ok) \xleftarrow{\$} \mathsf{GKg}(1^k)$; CU $\leftarrow \emptyset$; HU $\leftarrow \emptyset$

$(m, \sigma) \xleftarrow{\$} A(gpk, ok \ :\ \mathsf{SndToI}(\cdot, \cdot), \mathsf{AddU}(\cdot), \mathsf{RReg}(\cdot), \mathsf{USK}(\cdot), \mathsf{CrptU}(\cdot, \cdot))$

If $\mathsf{GVf}(gpk, m, \sigma) = 0$ then return 0 ; $(i, \tau) \leftarrow \mathsf{Open}(gpk, ok, \boldsymbol{reg}, m, \sigma)$

If $i = 0$ or $\mathsf{Judge}(gpk, i, \boldsymbol{upk}[i], m, \sigma, \tau) = 0$ then return 1 else return 0

Experiment $\mathbf{Exp}^{\mathrm{nf}}_{\mathcal{GS},A}(k)$

$(gpk, ik, ok) \xleftarrow{\$} \mathsf{GKg}(1^k)$; CU $\leftarrow \emptyset$; HU $\leftarrow \emptyset$

$(m, \sigma, i, \tau) \xleftarrow{\$} A(gpk, ok, ik \ :\ \mathsf{SndToU}(\cdot, \cdot), \mathsf{WReg}(\cdot, \cdot), \mathsf{GSig}(\cdot, \cdot), \mathsf{USK}(\cdot), \mathsf{CrptU}(\cdot, \cdot))$

If $\mathsf{GVf}(gpk, m, \sigma) = 0$ then return 0

If the following are all true then return 1 else return 0:

- $i \in$ HU and $\boldsymbol{gsk}[i] \neq \varepsilon$ and $\mathsf{Judge}(gpk, i, \boldsymbol{upk}[i], m, \sigma, \tau) = 1$
- A did not query $\mathsf{USK}(i)$ or $\mathsf{GSig}(i, m)$

Figure 3: Experiments used to define correctness, anonymity, traceability and non-frameability of a dynamic group signature scheme $\mathcal{GS} = (\mathsf{GKg}, \mathsf{UKg}, \mathsf{Join}, \mathsf{Iss}, \mathsf{GSig}, \mathsf{GVf}, \mathsf{Open}, \mathsf{Judge})$.

We say that dynamic group signature scheme $\mathcal{GS}$ is *anonymous* if the function $\mathbf{Adv}^{\mathrm{anon}}_{\mathcal{GS},A}(\cdot)$ is negligible for any polynomial-time adversary A.

Discussion. The definition is liberal with regard to what it means for the adversary to win. It need not recover the identity of a signer from a signature, but, following [5], need only distinguish which of two signers of its choice signed a target message of its choice. Formally, this means it wins if it guesses the value of the bit b in the $\mathsf{Ch}(b, \cdot, \cdot, \cdot)$ oracle. In the process, the adversary is provided with extremely strong attack capabilities, including the ability to fully corrupt the issuer. (The adversary is not only given the issuer key ik, but is provided access to the $\mathsf{SndToI}(\cdot, \cdot)$ oracle, which enables it to play the role of issuer in interacting with users in the join protocol.) The adversary is additionally allowed to obtain both the personal private key and the private signing key of any user (via the USK oracle); read, write or modify the content of the registration table (via the $\mathsf{RReg}, \mathsf{WReg}$ oracles); corrupt users and interact with the issuer on their behalf (via the $\mathsf{CrptU}, \mathsf{SndToU}$ oracles); and obtain the identity of the signer of any signature except the challenge one (via the Open oracle).

We do not provide the adversary access to the GSig and AddU oracles because they are redundant given the capabilities already provided to the adversary. Naturally, the adversary is also denied the opener's key ok, since the latter would enable it to run the Open algorithm. (Meaning the opener must be assumed uncorrupt.)

4.3 Traceability

Formal definition. To dynamic group signature scheme $\mathcal{GS}$, any adversary A and any $k \in \mathbb{N}$ we associate the experiment $\mathbf{Exp}^{\text{trace}}_{\mathcal{GS},A}(k)$ depicted in Figure 3. We let

$$\mathbf{Adv}^{\text{trace}}_{\mathcal{GS},A}(k) \;=\; \Pr\left[\,\mathbf{Exp}^{\text{trace}}_{\mathcal{GS},A}(k) = 1\,\right] .$$

We say that dynamic group signature scheme $\mathcal{GS}$ is *traceable* if the function $\mathbf{Adv}^{\text{trace}}_{\mathcal{GS},A}(\cdot)$ is negligible for any polynomial-time adversary A.

Discussion. Traceability asks that the adversary be unable to produce a signature such that either the honest opener declares itself unable to identify the origin of the signature (meaning the Open algorithm returns (i, τ) with $i = 0$), or, the honest opener believes it has identified the origin but is unable to produce a correct proof of its claim (meaning the Open algorithm returns (i, τ) with $i > 0$ but the proof τ is rejected by the judge). In the process, the adversary is allowed to create honest group members (via the AddU oracle); obtain both the personal private key and the private signing key of any user (via the USK oracle); read the content of the registration table (via the RReg oracles); and corrupt users and interact with the issuer on their behalf (via the CrptU, SndToU oracles).

Note that traceability cannot be achieved in the presence of even a partially corrupt issuer, for such an issuer can create dummy users with valid signing keys and thus create untraceable signatures. (That is, the assumption that the issuer is uncorrupt is minimal). Accordingly, in the definition, the adversary is not given ik as input and not given a SndToU oracle. Also it is not allowed to write to the registration table (meaning it is not given a WReg oracle) since it could otherwise remove the information enabling a group member to be traced. Also, the assumption that the opener is partially but not fully corrupt is minimal, for a fully corrupt opener could simply refuse to trace.

4.4 Non-frameability

Formal definition. To dynamic group signature scheme $\mathcal{GS}$, any adversary A and any $k \in \mathbb{N}$ we associate the experiment $\mathbf{Exp}^{\text{nf}}_{\mathcal{GS},A}(k)$ depicted in Figure 3. We let

$$\mathbf{Adv}^{\text{nf}}_{\mathcal{GS},A}(k) \;=\; \Pr\left[\,\mathbf{Exp}^{\text{nf}}_{\mathcal{GS},A}(k) = 1\,\right] .$$

We say that dynamic group signature scheme $\mathcal{GS}$ is *non-frameable* if the function $\mathbf{Adv}^{\text{nf}}_{\mathcal{GS},A}(\cdot)$ is negligible for any polynomial-time adversary A.

Discussion. Non-frameability asks that the adversary be unable to create a judge-accepted proof that an honest user produced a certain valid signature unless this user really did produce this signature. (This implies the more usual formulation, namely that it cannot produce a signature that an honest opener would attribute to a user unless the latter really did produce it, because, if it could produce such a signature, it could also produce the judge-accepted proof. The latter is true because we give it the secret key of the opener). The adversary outputs a message m, a signature σ, an identity i and a proof τ. It wins if σ is a valid signature of m, i is an honest user, and the judge accepts τ as a proof that i produced σ, yet the adversary did not query the signing oracle GSig with i, m and did not obtain i's signing key $\boldsymbol{gsk}[i]$ via the USK oracle. Barring these restrictions, the adversary is extremely powerful, and in particular much stronger than for traceability (which is why, unlike [5], we separate the two). In particular it may fully corrupt both the opener and the issuer. (Reflected in its getting input ok, ik and having access to the SndToU oracle.) Additionally, it may create a colluding subset of users by using its USK oracle to obtain signing keys of all users except the target one it outputs, and also corrupt users via CrptU.

4.5 Remarks

Recall that in [5] the issuing process was static and trusted, and their single authority played the role of opener. Their full-traceability requirement, which covered both traceability and non-frameability, allowed the opener to be partially but not fully corrupt. We are asking for traceabiltiy under the same conditions, which, as we have argued above, are minimal in the dynamic setting. But we ask for non-frameability under much more adverse conditions, namely when both authorities may be fully corrupt. (In achieving this, the PKI is crucial). This is the motivation for separating their single requirement into two.

We note that a reader might find that what is intuitively regarded as traceability is covered by the combination of traceability and non-frameability rather than by the formal traceability alone.

In Appendix A we point out that, as in the static case [5], the key requirements that we define (anonymity, traceability and non-frameability) are strong enough to capture and imply all existing informal security requirements in the literature.

5 Our Construction

We begin by describing the primitives we use. We then describe our construction, and state the security results. Their proofs are in Appendix C.

5.1 Primitives

Digital signature schemes. We use a digital signature scheme $\mathcal{DS} = (\mathsf{K_s}, \mathsf{Sig}, \mathsf{Vf})$ specified, as usual, by algorithms for key generation, signing and verifying. It should satisfy the standard notion of unforgeability under chosen message attack [22].

We now recall the definition. Consider the experiment $\mathbf{Exp}^{\text{unforg-cma}}_{\mathcal{DS},A}(k)$ in Figure 4, involving a *forger* A. A pair (pk, sk) of public/secret keys for the signature scheme is generated by running the key generation algorithm on the security parameter $(pk, sk) \stackrel{\$}{\leftarrow} \mathsf{K_s}(1^k)$. Next, A is given as input pk, and is also provided access to a signing oracle $\mathsf{Sig}(sk, \cdot)$. The forger can submit (any number of) messages to the oracle, and obtain in return signatures, under secret key sk, on these messages. Finally, A outputs an attempted forgery (m, σ). The experiment returns 1 if σ is a valid signature on m, and m was never queried to the signing oracle, and returns 0 otherwise. We define the advantage of forger A as:

$$\mathbf{Adv}^{\text{unforg-cma}}_{\mathcal{DS},A}(k) = \Pr\left[\mathbf{Exp}^{\text{unforg-cma}}_{\mathcal{DS},A}(k) = 1\right]$$

where the probability is taken on the coins of the key generation algorithm, the coins of the signature algorithm and the coins of the adversary. We say that a digital scheme $\mathcal{DS}$ is secure against forgeries under chosen message attack if the function $\mathbf{Adv}^{\text{unforg-cma}}_{\mathcal{DS},A}(\cdot)$ is negligible for any polynomial-time adversary A.

Encryption schemes. We use a public-key encryption scheme $\mathcal{AE} = (\mathsf{K_e}, \mathsf{Enc}, \mathsf{Dec})$ specified, as usual, by algorithms for key generation, encryption and decryption. It should satisfy the standard notion of indistinguishability under adaptive chosen-ciphertext attack (IND-CCA) [28]. We now recall the definition. Consider the experiment $\mathbf{Exp}^{\text{ind-cca-}b}_{\mathcal{AE},A}(k)$ in Figure 4, involving an adversary A. A pair (pk, sk) of public/secret keys for the encryption scheme is generated by running the randomized

$\mathbf{Exp}^{\text{unforg-cma}}_{\mathcal{DS},A}(k)$ $(pk, sk) \xleftarrow{\$} \mathsf{K_s}(1^k)$ $(m, \sigma) \leftarrow A(pk \,:\, \mathsf{Sig}(sk, \cdot))$ If the following are true then return 1 Else return 0: - $\mathsf{Vf}(pk, m, \sigma) = 1$ - A did not make oracle query m	$\mathbf{Exp}^{\text{ind-cca}\_b}_{\mathcal{AE},A}(k)$ $r_e \xleftarrow{\$} \{0,1\}^{r(k)}$ $(pk, sk) \leftarrow \mathsf{K_e}(1^k; r_e)$ $d \leftarrow A(pk \,:\, \mathrm{LR}(\cdot,\cdot,b)), \mathsf{Dec}(sk, \cdot))$ Return d

Figure 4: $\mathbf{Exp}^{\text{unforg-cma}}_{\mathcal{DS},A}(k)$ and $\mathbf{Exp}^{\text{ind-cca}\_b}_{\mathcal{AE},A}(k)$

key generation algorithm on the security parameter $(pk, sk) \leftarrow \mathsf{K_e}(1^k; r_e)$ where the length of the randomness string $|r_e|$ is bounded by some fixed polynomial $r(k)$. Assume A never queries $\mathsf{Dec}(sk, \cdot)$ on a ciphertext previously returned by $\mathsf{Enc}(pk, \mathrm{LR}(\cdot,\cdot,b))$, and all queries to $\mathrm{LR}(\cdot,\cdot,b)$ consists of a pair of equal-length messages. For a bit b and message M_0, M_1, define $\mathrm{LR}(M_0, M_1, b) = M_b$. The advantage function of A is defined as:

$$\mathbf{Adv}^{\text{ind-cca}}_{\mathcal{AE},A}(k) = \Pr\left[\,\mathbf{Exp}^{\text{ind-cca}\_1}_{\mathcal{AE},A}(k) = 1\,\right] - \Pr\left[\,\mathbf{Exp}^{\text{ind-cca}\_0}_{\mathcal{AE},A}(k) = 1\,\right]$$

An encryption scheme $\mathcal{AE}$ is said to be IND-CCA secure if the function $\mathbf{Adv}^{\text{ind-cca}}_{\mathcal{AE},A}(\cdot)$ is negligible for any polynomial-time adversary A.

Simulation-sound non-interactive zero knowledge proof systems. The last building block we need are simulation-sound NIZK proofs of membership in NP languages. We use the following terminology. An NP-*relation over domain* Dom $\subseteq \{0,1\}^*$ is a subset ρ of $\{0,1\}^* \times \{0,1\}^*$ such that membership of $(x, w) \in \rho$ is decidable in time polynomial in the length of the first argument for all x in domain Dom. The language associated to ρ is the set of all $x \in \{0,1\}^*$ such that there exists a w for which $(x, w) \in \rho$. Often we will just use the term NP-relation, the domain being implicit. If $(x, w) \in \rho$ we will say that x is a *theorem* and w is a *proof* of x.

Fix an NP relation ρ over domain Dom. Consider a pair of polynomial time algorithms (P, V), where P is randomized and V is deterministic. They have access to a *common reference string*, R. We say that (P, V) is a non-interactive proof system for ρ over Dom if there exist polynomials p and ℓ such that the following two conditions are satisfied:

1. **Completeness:** $\forall k \in \mathbb{N},\ \forall (x, w) \in \rho$ with $|x| \leq \ell(k)$ and $x \in$ Dom–

$$\Pr\left[\, R \xleftarrow{\$} \{0,1\}^{p(k)}\,;\, \pi \xleftarrow{\$} P(1^k, x, w, R) \;:\; V(1^k, x, \pi, R) = 1 \,\right] \;=\; 1\,.$$

2. **Soundness:** $\forall k \in \mathbb{N},\ \forall \widehat{P},\ \forall x \in$ Dom such that $x \notin L_\rho$–

$$\Pr\left[\, R \xleftarrow{\$} \{0,1\}^{p(k)}\,;\, \pi \leftarrow \widehat{P}(1^k, x, R) \;:\; V(1^k, x, \pi, R) = 1 \,\right] \;\leq\; 2^{-k}\,.$$

We now detail the zero-knowledge requirement. Given a non-interactive proof-system (P, V) for relation ρ, consider a simulator Sim, i.e. a polynomial-time algorithm running in two stages. In the randomized gen stage it produces a simulated common reference string R. We stress that it does so before seeing any theorem, based only on a bound on the theorem length. In the (w.l.o.g. deterministic) prove stage it takes as input a theorem x and state information passed on by the first stage, and then produces a simulated proof for the validity of x with respect to R.

This two phase behavior is not required explicitly in the definitions of [19, 9] but the construction of [19] does have this property, and it is noted and and used in other places too.

Zero-knowledge is defined by means of a *distinguisher* D which tries to distinguish between proofs

$\mathbf{Exp}^{\text{zk-0}}_{P,\text{SIM},D}(k)$	$\mathbf{Exp}^{\text{zk-1}}_{P,\text{SIM},D}(k)$
$\quad (R, \text{St}_S) \xleftarrow{\$} \text{SIM}(\mathsf{gen}, 1^k)$	$\quad R \xleftarrow{\$} \{0,1\}^{p(k)}$
$\quad d \leftarrow D(R : \mathsf{Prove}_0(\cdot,\cdot))$	$\quad d \leftarrow D(R : \mathsf{Prove}_1(\cdot,\cdot))$
$\quad$ Return d	$\quad$ Return d
$\mathsf{Prove}_0(x, w)$	$\mathsf{Prove}_1(x, w)$
$\quad \pi \leftarrow \text{SIM}(\mathsf{prove}, \text{St}_S, x)$	$\quad \pi \leftarrow P(1^k, x, w, R)$
$\quad$ Return π	$\quad$ Return π

$\mathbf{Exp}^{\text{ss}}_{\Pi,A}(k)$

$\quad (R, \text{St}_S) \leftarrow \text{SIM}(\mathsf{gen}, 1^k)$; $(x, \pi) \xleftarrow{\$} A(R : \text{SIM}(\mathsf{prove}, \text{St}_S, \cdot))$

$\quad$ If all of the following are true then return 1 else return 0:

$\quad\quad$ (1) $x \notin L_\rho$

$\quad\quad$ (2) π was not returned by A's oracle in response to a query x

$\quad\quad$ (3) $V(1^k, x, \pi, R) = 1$.

Figure 5: $\mathbf{Exp}^{\text{zk-}b}_{P,\text{SIM},D}(k)$ and $\mathbf{Exp}^{\text{ss}}_{\Pi,A}(k)$

produced by a prover (with respect to a real common random string), or a simulator (with respect to a simulated common random string). More precisely, we consider two experiments in Figure 5 involving distinguisher D, $\mathbf{Exp}^{\text{zk-0}}_{P,\text{SIM},D}(k)$ and $\mathbf{Exp}^{\text{zk-1}}_{P,\text{SIM},D}(k)$. In the first experiment, a reference string is produced via the simulator's gen stage, while in the second it is a random string. In either case, the distinguisher chooses a theorem x based on R. It is mandated that $x \in \text{Dom}$. D is required to supply a correct witness for x relative to ρ, else it loses, meaning the experiment returns 0. (Note this further weakens the distinguisher and thus makes the computational zk requirement less stringent.) By querying its Prove_b oracle with $(x, w) \in \rho$ where $x \in \text{Dom}$, the distinguisher is given as challenge a proof π, produced according to the simulator's prove stage in the first experiment, and according to the prover P in the second experiment. Here we assume that D makes exactly one query to Prove_b. The zk-advantage of D is

$$\mathbf{Adv}^{\text{zk}}_{P,\text{SIM},D}(k) = \Pr\left[\mathbf{Exp}^{\text{zk-1}}_{P,\text{SIM},D}(k) = 1\right] - \Pr\left[\mathbf{Exp}^{\text{zk-0}}_{P,\text{SIM},D}(k) = 1\right].$$

We say that a non-interactive proof system (P, V) is (computational) zero-knowledge if there exists a polynomial time simulator SIM s.t. for any polynomial time distinguisher D the function $\mathbf{Adv}^{\text{zk}}_{P,\text{SIM},D}(\cdot)$ is negligible. To show the dependency of SIM on (P, V) we will say that (P, V, SIM) is a zero-knowledge proof system.

We point out that we only require single-theorem NIZK as opposed to multiple-theorem NIZK, in that the distinguisher has a challenge real-or-simulated proof for only a single theorem. However, this weaker condition is made stronger by requiring that the simulator produce the reference string without seeing the theorem. Based on [19] and [8] (the latter corrects a bug in the former) there exists such zero-knowledge non-interactive proof system (P, V) for any NP-relation ρ assuming the existence of trapdoor permutations.

The last property we require is simulation-soundness [29]. Let $\Pi = (P, V, \text{SIM})$ be a zero knowledge interactive proof system for NP-relation ρ over domain Dom. Simulation-soundness is defined using the experiment $\mathbf{Exp}^{\text{ss}}_{\Pi,A}(k)$ in Figure 5 involving a simulation-soundness adversary A.

First, a "fake common" random string R, together with the associated trap-door information St_S is generated by running the simulator: $(R, \text{St}_S) \xleftarrow{\$} \text{SIM}(\mathsf{gen}, k)$. The string is passed to the simulation-soundness adversary, which has access to oracle $\text{SIM}(\mathsf{prove}, \text{St}_S, \cdot)$. Here we assume the adversary

User^i		Issuer^i
$(pk_i, sk_i) \stackrel{\$}{\leftarrow} \mathsf{K_s}(1^k)$; $sig_i \leftarrow \mathsf{Sig}(\boldsymbol{usk}[i], pk_i)$	$\xrightarrow{pk_i,\, sig_i}$	
		If $\mathsf{Vf}(\boldsymbol{upk}[i], pk_i, sig_i) = 1$ then
		$\quad \mathrm{cert}_i \leftarrow \mathsf{Sig}(sk_s, \langle i, pk_i \rangle)$
		$\quad \boldsymbol{reg}[i] \leftarrow (pk_i, sig_i)$
	$\xleftarrow{\mathrm{cert}_i}$	Else $\mathrm{cert}_i \leftarrow \varepsilon$
$\boldsymbol{gsk}[i] \leftarrow (i, pk_i, sk_i, \mathrm{cert}_i)$		

Figure 6: The group-joining protocol.

makes exactly one query to $\textsc{Sim}(\mathsf{prove}, \mathrm{St}_S, \cdot)$. At the end of the experiment the adversary is required to output a pair (x, π). The advantage of A is defined by

$$\mathbf{Adv}^{\mathrm{ss}}_{\Pi,A}(k) = \Pr\left[\,\mathbf{Exp}^{\mathrm{ss}}_{\Pi,A}(k) = 1\,\right]$$

and we say that $(P, V, \textsc{Sim})$ is a simulation-sound if for all polynomial time adversaries A, there exists a negligible function $\nu_A(\cdot)$, such that $\mathbf{Adv}^{\mathrm{ss}}_{\Pi,A}(k) \leq \nu_A(k)$ for all k.

Sahai [29], building on [19, 8], shows that if trapdoor permutations exist, then any NP-relation has a simulation-sound, non-interactive zero knowledge proof system.

5.2 Overview of our construction

We fix a digital signature scheme $\mathcal{DS} = (\mathsf{K_s}, \mathsf{Sig}, \mathsf{Vf})$ and a public-key encryption scheme $\mathcal{AE} = (\mathsf{K_e}, \mathsf{Enc}, \mathsf{Dec})$ as above. We now show that the building blocks above can be used to construct a group signature scheme $\mathcal{GS} = (\mathsf{GKg}, \mathsf{UKg}, \mathsf{GSig}, \mathsf{GVf}, \mathsf{Join}, \mathsf{Iss}, \mathsf{Open}, \mathsf{Judge})$ that is anonymous, traceable and non-frameable. We now present an overview of our construction.

The group public key gpk consists of the security parameter k, a public encryption key pk_e, a verification key pk_s for digital signatures which we call the *certificate verification* key, and two reference strings R_1 and R_2 for NIZK proofs. We denote by sk_s the signing key corresponding to pk_s, and call it the *certificate creation* key. The issuer secret key ik is the certificate creation key sk_s. The opener secret key ok is the decryption key sk_e corresponding to pk_e, together with the random coins r_e used to generate (sk_e, pk_e). The certificate creation key sk_s is however denied to the group opener. (This prevents the latter from issuing certificates for keys it generates itself, and is important to attain traceability.)

In the group-joining protocol, user i generates a verification key pk_i and the corresponding signing key sk_i. It uses its personal private key $\boldsymbol{usk}[i]$ to produce a signature sig_i on pk_i. The signature sig_i prevents the user from being framed by a corrupt issuer. (The personal public and private key pair $(\boldsymbol{upk}[i], \boldsymbol{usk}[i])$ were obtained by running the user-key generation algorithm prior to the group-joining protocol. This is handled by the oracles.) The users sends pk_i, sig_i to the issuer, who issues membership to i by signing pk_i using the certificate creation key sk_s. The issuer then stores (pk_i, sig_i) in the registration table. Later, sig_i can be used by the opener to produce proofs for its claims. See Figure 6.

A group member i can produce a signature for a message m under pk_i by using its secret signing key sk_i. To make this verifiable without losing anonymity, it encrypts the verification key pk_i under pk_e and then proves in zero-knowledge that verification succeeds with respect to pk_i. However, to prevent someone from simply creating their own key pair sk_i, pk_i and doing this, it also encrypts i and its certificate cert_i, and proves in zero-knowledge that this certificate is a signature of $\langle i, pk_i \rangle$ under

Algorithm $\mathsf{GKg}(1^k)$

$R_1 \xleftarrow{\$} \{0,1\}^{p_1(k)}$; $R_2 \xleftarrow{\$} \{0,1\}^{p_2(k)}$
$r_e \xleftarrow{\$} \{0,1\}^{r(k)}$; $(pk_e, sk_e) \leftarrow \mathsf{K_e}(1^k; r_e)$
$(pk_s, sk_s) \xleftarrow{\$} \mathsf{K_s}(1^k)$
$gpk \leftarrow (1^k, R_1, R_2, pk_e, pk_s)$
$ok \leftarrow (sk_e, r_e)$; $ik \leftarrow sk_s$
Return (gpk, ok, ik)

Algorithm $\mathsf{UKg}(1^k)$

$(upk, usk) \xleftarrow{\$} \mathsf{K_s}(1^k)$
Return (upk, usk)

Algorithm $\mathsf{GVf}(gpk, (m, \sigma))$

Parse gpk as $(1^k, R_1, R_2, pk_e, pk_s)$
Parse σ as (C, π_1)
Return $V_1(1^k, (pk_e, pk_s, m, C), \pi_1, R_1)$

Algorithm $\mathsf{GSig}(gpk, \boldsymbol{gsk}[i], m)$

Parse gpk as $(1^k, R_1, R_2, pk_e, pk_s)$
Parse $\boldsymbol{gsk}[i]$ as $(i, pk_i, sk_i, \mathrm{cert}_i)$
$s \leftarrow \mathsf{Sig}(sk_i, m)$; $r \xleftarrow{\$} \{0,1\}^k$
$C \leftarrow \mathsf{Enc}(pk_e, \langle i, pk_i, \mathrm{cert}_i, s\rangle; r)$
$\pi_1 \xleftarrow{\$} P_1(1^k, (pk_e, pk_s, m, C),$
$\quad (i, pk_i, \mathrm{cert}_i, s, r), R_1)$
$\sigma \leftarrow (C, \pi_1)$
Return σ

Algorithm $\mathsf{Open}(gpk, ok, \boldsymbol{reg}[\,], m, \sigma)$

Parse gpk as $(1^k, R_1, R_2, pk_e, pk_s)$
Parse ok as (sk_e, r_e); Parse σ as (C, π_1)
$M \leftarrow \mathsf{Dec}(sk_e, C)$; Parse M as $\langle i, pk, \mathrm{cert}, s\rangle$
If $\boldsymbol{reg}[i] \neq \varepsilon$ then
$\quad$ Parse $\boldsymbol{reg}[i]$ as (pk_i, sig_i)
Else $pk_i \leftarrow \varepsilon$; $sig_i \leftarrow \varepsilon$
$\pi_2 \leftarrow P_2(1^k, (pk_e, C, i, pk, \mathrm{cert}, s), (sk_e, r_e), R_2)$
If $V_1(1^k, (pk_e, pk_s, m, C), \pi_1, R_1) = 0$ then
$\quad$ Return $(0, \varepsilon)$
If $pk \neq pk_i$ or $\boldsymbol{reg}[i] = \varepsilon$ then return $(0, \varepsilon)$
$\tau \leftarrow (pk_i, sig_i, i, pk, \mathrm{cert}, s, \pi_2)$
Return (i, τ)

Algorithm $\mathsf{Judge}(gpk, i, \boldsymbol{upk}[i], m, \sigma, \tau)$

Parse gpk as $(1^k, R_1, R_2, pk_e, pk_s)$
Parse σ as (C, π_1)
If $(i, \tau) = (0, \varepsilon)$ then
$\quad$ Return $V_1(1^k, (pk_e, pk_s, m, C), \pi_1, R_1) = 0$
Parse τ as $(\overline{pk}, \overline{sig}, i', pk, \mathrm{cert}, s, \pi_2)$
If $V_2(1^k, (C, i', pk, \mathrm{cert}, s), \pi_2, R_2) = 0$ then
$\quad$ Return 0
If all of the following are true then return 1
Else return 0:
- $i = i'$;
- $\mathsf{Vf}(\boldsymbol{upk}[i], \overline{pk}, \overline{sig}) = 1$;
- $\overline{pk} = pk$

Figure 7: Algorithms $\mathsf{GKg}, \mathsf{UKg}, \mathsf{GVf}, \mathsf{GVf}, \mathsf{GSig}, \mathsf{Open}, \mathsf{Judge}$ of our dynamic group signature scheme.

the certificate verification key pk_s present in the group public key. Group signature verification comes down to verification of the NIZK proofs.

Opening is possible because the group opener has the decryption key sk_e. It obtains the user identity i by decrypting the ciphertext in the signature. When i is indeed an existing user, the opener proves its claim by supplying evidence that it decrypts the ciphertext correctly, and the user public key it obtained from decryption is authentic (i.e. signed by user i using $\boldsymbol{usk}[i]$). The former is accomplished by a zero-knowledge proof. The judge algorithm simply checks if these proofs are correct.

5.3 Specification of our construction

We now provide a detailed specification of the scheme. We begin with the witness relations ρ_1 and ρ_2 underlying the zero-knowledge proofs. Relation ρ_1 is defined as follows: $((pk_e, pk_s, m, C), (i, pk', \mathrm{cert}, s, r)) \in \rho_1$ iff

$$\mathsf{Vf}(pk_s, \langle i, pk'\rangle, \mathrm{cert}) = 1,\ \mathsf{Vf}(pk', m, s) = 1 \text{ and } \mathsf{Enc}(pk_e, \langle i, pk', \mathrm{cert}, s\rangle; r) = C\ .$$

Here m is a k-bit message, C a ciphertext and s a signature. We are writing $\mathsf{Enc}(pk_e, m; r)$ for the encryption of message m under key pk_e using coins r, and assume that $|r| = k$. The domain

Algorithm $\mathsf{Join}(\mathrm{St}_{join}, M_{in})$
  If $M_{in} = \varepsilon$ then
    Parse St_{join} as (gpk, i, upk_i, usk_i)
    $(pk_i, sk_i) \stackrel{\$}{\leftarrow} \mathsf{K_s}(1^k)$; $sig_i \leftarrow \mathsf{Sig}(usk_i, pk_i)$
    $\mathrm{St}'_{join} \leftarrow (i, pk_i, sk_i)$; $M_{out} \leftarrow (pk_i, sig_i)$
    Return $(\mathrm{St}'_{join}, M_{out}, \mathsf{cont})$
  Else
    Parse St_{join} as (i, pk_i, sk_i)
    Parse M_{in} as cert_i
    $\mathrm{St}'_{join} \leftarrow (i, pk_i, sk_i, \mathrm{cert}_i)$
    Return $(\mathrm{St}'_{join}, \varepsilon, \mathsf{accept})$
	Algorithm $\mathsf{Iss}(\mathrm{St}_{issue}, M_{in}, dec)$
	  $M_{out} \leftarrow \varepsilon$; $dec' \leftarrow \mathsf{reject}$
	  If $dec = \mathsf{cont}$ then
	    Parse St_{issue} as (gpk, ik, i, upk_i)
	    Parse M_{in} as (pk_i, sig_i)
	    Parse ik as sk_s
	    If $\mathsf{Vf}(upk_i, pk_i, sig_i) = 1$ then
	      $\mathrm{cert}_i \leftarrow \mathsf{Sig}(sk_s, \langle i, pk_i \rangle)$
	      $\mathrm{St}_{issue} \leftarrow (pk_i, sig_i)$
	      $M_{out} \leftarrow \mathrm{cert}_i$; $dec' \leftarrow \mathsf{accept}$
	  Return $(\mathrm{St}_{issue}, M_{out}, dec')$

Figure 8: The Join and Iss algorithms defining the group-joining protocol of our dynamic group signature scheme.

Dom_1 corresponding to ρ_1 is the set of all (pk_e, pk_s, m, C) such that pk_e (resp. pk_s) is a public key having non-zero probability of being produced by $\mathsf{K_e}$ (resp. $\mathsf{K_s}$) on input k, and m is a k-bit string. It is immediate that ρ_1 is an NP relation over Dom_1. Relation ρ_2 is defined as follows: $((pk_e, C, i, pk, \mathrm{cert}, s), (sk_e, r_e)) \in \rho_2$ iff

$$\mathsf{K_e}(1^k; r_e) = (pk_e, sk_e) \text{ and } \mathsf{Dec}(sk_e, C) = \langle i, pk, \mathrm{cert}, s \rangle$$

Here C is a ciphertext, i an identity and s a signature. The domain Dom_2 corresponding to ρ_2 is the set of all $(pk_e, C, i, pk, \mathrm{cert}, s)$ such that pk_e is a public key having non-zero probability of being produced by $\mathsf{K_e}$ on input k. It is immediate that ρ_2 is an NP relation over Dom_2.

We fix a proof system (P_1, V_1) for ρ_1 and (P_2, V_2) for ρ_2. Figure 7 shows the details of the algorithms $\mathsf{GKg}, \mathsf{UKg}, \mathsf{GVf}, \mathsf{GVf}, \mathsf{GSig}, \mathsf{Open}, \mathsf{Judge}$ of our dynamic group signature scheme $\mathcal{GS}$, based on the above. The details of the algorithms $\mathsf{Join}, \mathsf{Iss}$ that embody the join protocol of Figure 6 are shown in Figure 8.

Security Results. Fix digital signature scheme $\mathcal{DS} = (\mathsf{K_s}, \mathsf{Sig}, \mathsf{Vf})$, public-key encryption scheme $\mathcal{AE} = (\mathsf{K_e}, \mathsf{Enc}, \mathsf{Dec})$, NP-relations ρ_1 over domain Dom_1, ρ_2 over domain Dom_2, and their non-interactive proof systems (P_1, V_1) and (P_2, V_2) as above, and let $\mathcal{GS} = (\mathsf{GKg}, \mathsf{UKg}, \mathsf{GSig}, \mathsf{GVf}, \mathsf{Join}, \mathsf{Iss}, \mathsf{Open}, \mathsf{Judge})$ denote the dynamic group signature scheme associated to them as per our construction. We derive our main result (Theorem 5.4) via the following three lemmas, proved in Appendix C.

Lemma 5.1 *If $\mathcal{AE}$ is an IND-CCA secure encryption scheme, (P_1, V_1) is a simulation sound, computational zero-knowledge proof system for ρ_1 over Dom_1 and (P_2, V_2) is a computational zero-knowledge proof system for ρ_2 over Dom_2, then group signature scheme $\mathcal{GS}$ is anonymous.* ▮

Lemma 5.2 *If digital signature scheme $\mathcal{DS}$ is secure against forgery under chosen-message attack , (P_1, V_1) is a sound non-interactive proof system for ρ_1 over Dom_1 and (P_2, V_2) is a sound non-interactive proof system for ρ_2 over Dom_2, then group signature scheme $\mathcal{GS}$ is traceable.* ▮

Lemma 5.3 *If digital signature scheme $\mathcal{DS}$ is secure against forgery under chosen-message attack , (P_1, V_1) is a sound non-interactive proof system for ρ_1 over Dom_1 and (P_2, V_2) is a sound non-interactive proof system for ρ_2 over Dom_2, then group signature scheme $\mathcal{GS}$ is non-frameable.* ▮

We know that if trapdoor permutations exist then so do secure digital signature schemes [6], IND-CCA secure encryption schemes [17, 29] and simulation sound NIZK proofs for NP [29]. As a consequence we have:

Theorem 5.4 *If there exists a family of trapdoor permutations, then there exists a dynamic group signature scheme that is anonymous, traceable and non-frameable.* ▌

Acknowledgments

We thank Bogdan Warinschi for comments on a previous draft.

References

[1] G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A practical and provably secure coalition-resistant group signature scheme. *Advances in Cryptology – CRYPTO '00*, Lecture Notes in Computer Science Vol. 1880, M. Bellare ed., Springer-Verlag, 2000.

[2] G. Ateniese and G. Tsudik. Quasi-efficient revocation in group signature schemes. *Financial Cryptography '02*, Lecture Notes in Computer Science Vol. 2357, M. Blaze ed., Springer-Verlag, 2002.

[3] G. Ateniese and G. Tsudik. Group signatures à la carte. *Proceedings of the 10th Annual Symposium on Discrete Algorithms*, ACM-SIAM, 1999.

[4] G. Ateniese and G. Tsudik. Some open issues and directions in group signature. *Financial Cryptography '99*, Lecture Notes in Computer Science Vol. 1648, M. Franklin ed., Springer-Verlag, 1999.

[5] M. Bellare, D. Micciancio and B. Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. *Advances in Cryptology – EUROCRYPT '03*, Lecture Notes in Computer Science Vol. 2656, E. Biham ed., Springer-Verlag, 2003.

[6] M. Bellare and S. Micali. How to sign given any trapdoor permutation. *JACM*, 39(1):214–233, 1992.

[7] M. Bellare and P. Rogaway. Entity authentication and key distribution. *Advances in Cryptology – CRYPTO '93*, Lecture Notes in Computer Science Vol. 773, D. Stinson ed., Springer-Verlag, 1993.

[8] M. Bellare and M. Yung. Certifying permutations: Non-interactive zero-knowledge based on any trapdoor permutation. *J. of Cryptology* 9(1):149–166, 1996.

[9] M. Blum, A. DeSantis, S. Micali, and G. Persiano. Non-interactive zero-knowledge proof systems. *SIAM J. on Computing*, 20(6):1084–1118, 1991.

[10] E. Bresson and J. Stern. Efficient revocation in group signatures. *Public-Key Cryptography '01*, Lecture Notes in Computer Science Vol. 1992, K. Kim ed., Springer-Verlag, 2001.

[11] J. Camenisch. Efficient and generalized group signature. *Advances in Cryptology – EUROCRYPT '97*, Lecture Notes in Computer Science Vol. 1233, W. Fumy ed., Springer-Verlag, 1997.

[12] J. Camenisch and A. Lysyanskaya. An identity-escrow scheme with appointed verifiers. *Advances in Cryptology – CRYPTO '01*, Lecture Notes in Computer Science Vol. 2139, J. Kilian ed., Springer-Verlag, 2001.

[13] J. Camenisch and M. Michels. A group signature scheme with improved efficiency. *Advances in Cryptology – ASIACRYPT '98*, Lecture Notes in Computer Science Vol. 1514, D. Pei ed., Springer-Verlag, 1998.

[14] J. Camenisch and M. Stadler. Efficient group signatures schemes for large groups. *Advances in Cryptology – CRYPTO '97*, Lecture Notes in Computer Science Vol. 1294, B. Kaliski ed., Springer-Verlag, 1997.

[15] D. Chaum and E. van Heyst. Group signatures. *Advances in Cryptology – EUROCRYPT '91*, Lecture Notes in Computer Science Vol. 547, D. Davies ed., Springer-Verlag, 1991.

[16] L. Chen and T. P. Pedersen. New group signature schemes. *Advances in Cryptology – EUROCRYPT '94*, Lecture Notes in Computer Science Vol. 950, A. De Santis ed., Springer-Verlag, 1994.

[17] D. Dolev, C. Dwork, and M. Naor. Nonmalleable cryptography. *SIAM J. on Computing*, 30(2):391–437, 2000.

[18] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. *Advances in Cryptology – CRYPTO '86*, Lecture Notes in Computer Science Vol. 263, A. Odlyzko ed., Springer-Verlag, 1986.

[19] U. Feige, D. Lapidot, and A. Shamir. Multiple non-interactive zero-knowledge proofs under general assumptions. *SIAM J. on Computing*, 29(1):1–28, 1999.

[20] O. Goldreich. A uniform-complexity treatment of encryption and zero-knowledge. *J. of Cryptology*, 6(1):21–53, 1993.

[21] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Science*, 28:270–299, 1984.

[22] S. Goldwasser, S. Micali, and R. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. on Computing*, 17(2):281–308, 1988.

[23] S. Goldwasser and Y. Tauman. On the (In)security of the Fiat-Shamir paradigm. *Proceedings of the 44th Symposium on Foundations of Computer Science*, IEEE, 2003.

[24] A. Kiayias, Y. Tsiounis and M. Yung. Traceable signatures. *Advances in Cryptology – EUROCRYPT '04*, Lecture Notes in Computer Science Vol. 3027, C. Cachin and J. Camenisch ed., Springer-Verlag, 2004.

[25] S. Micali, C. Rackoff, and B. Sloan. The notion of security for probabilistic cryptosystems. *SIAM J. on Computing*, 17(2):412–426, 1988.

[26] M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. *Proceedings of the 22nd Annual Symposium on the Theory of Computing*, ACM, 1990.

[27] H. Petersen. How to convert any digital signature scheme into a group signature scheme. *Proceedings of Security Protocols Workshop '97.*

[28] C. Rackoff and D. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. *Advances in Cryptology – CRYPTO '91*, Lecture Notes in Computer Science Vol. 576, J. Feigenbaum ed., Springer-Verlag, 1991.

[29] A. Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. *Proceedings of the 40th Symposium on Foundations of Computer Science*, IEEE, 1999.

[30] D. Song. Practical forward-secure group signature schemes. *Proceedings of the 8th Annual Conference on Computer and Communications Security*, ACM, 2001.

A Relations to Existing Security Notions

As in the static case [5], the key requirements that we define (anonymity, traceability and non-frameability) are strong enough to capture and imply all existing informal security requirements in the literature. We briefly argue this here.

Unforgeability. Unforgeability means that it is computationally infeasible for an adversary A to produce message signature pairs (m, σ) that are accepted by the verification algorithm, without knowledge of the secret key(s). This follows immediately from traceability plus non-frameability. Let $(i, \tau) \stackrel{\$}{\leftarrow} \mathsf{Open}(gpk, ok, \boldsymbol{reg}, m, \sigma)$. If $i = 0$ then traceability is violated. If $i > 0$ then we can construct an adversary that violates non-frameability by itself running Open. We omit the details.

Exculpability. Exculpability means that no member of the group and not even the opener or issuer can produce signatures on behalf of other users. This is implied by our formulation of non-frameability.

Traceability. The informal notion of traceability means that it is not possible to produce signatures which can not be traced to one of the group that has produced the signature. Our formulation of the traceability requirement is stronger since the adversary has access to all user's secret key as well as the group manager's keys, and thus it captures the informal traceability requirement.

Coalition resistance. Coalition resistance means a group of signers colluding together should not be able to generate signatures that cannot be traced to any of them. As with unforgeability, this is implied by traceability plus non-frameability.

Experiment $\mathbf{Exp}^{i}_{\mathcal{GS}}(B)$
  $(gpk, ik, ok) \xleftarrow{\$} \mathsf{GKg}(1^k)$
  $\mathrm{CU} \leftarrow \emptyset$; $\mathrm{HU} \leftarrow \emptyset$; $\mathrm{GSet} \leftarrow \emptyset$; $cnt \leftarrow 0$
  $d \xleftarrow{\$} B(gpk : \mathsf{HGuess}^i(\cdot,\cdot,\cdot), \mathsf{Open}(\cdot,\cdot), \mathsf{SndToU}(\cdot,\cdot),$
    $\mathsf{WReg}(\cdot,\cdot), \mathsf{USK}(\cdot), \mathsf{CrptU}(\cdot,\cdot))$
  Return d

Oracle $\mathsf{HGuess}^i(i_0, i_1, m)$
  $cnt \leftarrow cnt + 1$
  If $cnt \le i$ then
    $\sigma \leftarrow \mathsf{GSig}(gpk, gsk_0, m)$
  Else $\sigma \leftarrow \mathsf{GSig}(gpk, gsk_1, m)$
  Return σ

Adversary $A(gpk : \mathsf{Ch}_b(\cdot,\cdot,\cdot), \mathsf{Open}(\cdot,\cdot),$
    $\mathsf{SndToU}(\cdot,\cdot), \mathsf{WReg}(\cdot,\cdot), \mathsf{USK}(\cdot), \mathsf{CrptU}(\cdot,\cdot))$
  $cnt \leftarrow 0$
  $I \xleftarrow{\$} \{1, \ldots, n(k)\}$
  $d \leftarrow B(gpk : \mathsf{Ch}(\cdot,\cdot,\cdot), \mathsf{Open}(\cdot,\cdot),$
    $\mathsf{SndToU}(\cdot,\cdot), \mathsf{WReg}(\cdot,\cdot), \mathsf{USK}(\cdot), \mathsf{CrptU}(\cdot,\cdot))$
  If $cnt < I$ then
    $\sigma \leftarrow \mathsf{Ch}_b(0, 0, \varepsilon)$ [oracle query]
  Return d

Oracle $\mathsf{Ch}(i_0, i_1, m)$
  $gsk_0 \leftarrow \mathsf{USK}(i_0)$ [oracle query]
  $gsk_1 \leftarrow \mathsf{USK}(i_1)$ [oracle query]
  $cnt \leftarrow cnt + 1$
  If $cnt < I$ then $\sigma \leftarrow \mathsf{GSig}(gpk, gsk_0, m)$
  If $cnt > I$ then $\sigma \leftarrow \mathsf{GSig}(gpk, gsk_1, m)$
  If $cnt = I$ then
    $\sigma \leftarrow \mathsf{Ch}_b(i_0, i_1, m)$ [oracle query]
  Return σ

Figure 9: Construction of adversary A

Framing. Framing means a set of group members should not be able to combine their keys to produce a valid signature such that the opening algorithm will attribute it to a different group member. It is clear that framing is a version of coalition resistance, and is therefore captured by our formulation of non-frameability.

Anonymity. The informal notion of anonymity is a weaker form of the anonymity requirement in this paper, where the adversary does not have capabilities as powerful as we give it, and is thus implied by our definition.

Unlinkability. Unlinkability means a party who sees a list of signatures cannot relate two signatures together as being produced by the same user. By similar reasoning to that in [5] we can show that a group signature scheme secure against anonymity is also secure against unlinkability.

B From many queries to one

The following says that in considering anonymity, we may without loss of generality restrict our attention to adversaries that make exactly one query to their $\mathsf{Ch}(b,\cdot,\cdot)$ oracle. This will be useful in later proofs.

Lemma B.1 *Given dynamic group signature scheme $\mathcal{GS}$, for any polynomial-time adversary B attacking the anonymity of $\mathcal{GS}$ that makes at most $n(k)$ queries to the $\mathsf{Ch}(b,\cdot,\cdot)$ oracle, where $n(k)$ is a polynomial, there exists a polynomial-time adversary A, also attacking the anonymity of $\mathcal{GS}$ that makes exactly one query to its $\mathsf{Ch}(b,\cdot,\cdot)$ oracle, and*

$$\mathbf{Adv}^{\mathrm{anon}}_{\mathcal{GS},B}(k) \le n(k) \cdot \mathbf{Adv}^{\mathrm{anon}}_{\mathcal{GS},A}(k) \ . \ \blacksquare$$

Proof of Lemma B.1: The proof is a standard hybrid argument. For completeness we provide the details.

For any $i \in \{0, \ldots, n(k)\}$, we associate to B an oracle $\mathsf{HGuess}^i(\cdot,\cdot,\cdot)$ and an experiment $\mathbf{Exp}^i_{\mathcal{GS}}(B)$, as indicated in Figure 9. Let

$$P(i) = \Pr\left[\, \mathbf{Exp}^i_{\mathcal{GS}}(B) = 1 \,\right]$$

Now, observe that oracles $\mathsf{HGuess}^0(\cdot,\cdot,\cdot)$ and $\mathsf{Ch}^1(\cdot,\cdot,\cdot)$ are equivalent, meaning that on any inputs, their responses are identically distributed. Similarly, oracles $\mathsf{HGuess}^{n(k)}(\cdot,\cdot,\cdot)$ and $\mathsf{Ch}^0(\cdot,\cdot,\cdot)$ are equivalent. Hence,

$$\begin{aligned} P(0) &= \Pr\left[\, \mathbf{Exp}^{\text{anon-1}}_{\mathcal{GS},B}(k) = 1 \,\right] \\ P(n(k)) &= \Pr\left[\, \mathbf{Exp}^{\text{anon-0}}_{\mathcal{GS},B}(k) = 1 \,\right] \end{aligned} \tag{1}$$

The details of A are given in Figure 9. Adversary A is intended to run in the experiment $\mathbf{Exp}^{\text{anon-}b}_{\mathcal{GS}}(k)$. It begins by initializing a counter to 0, and picking I at random from $\{1, \ldots, n(k)\}$. Then it runs adversary B against $\mathcal{GS}$. When answering B's queries to oracles $\mathsf{Open}(\cdot,\cdot), \mathsf{CrptU}(\cdot,\cdot), \mathsf{SndToU}(\cdot,\cdot), \mathsf{WReg}(\cdot,\cdot)$ and $\mathsf{USK}(\cdot)$, A simply queries its own oracles and return the answer to B. When answering B's queries to $\mathsf{Ch}(\cdot,\cdot,\cdot)$, B's first $I-1$ queries are answered by signatures of the first identity in B's query, the I-th query is answered by calling A's $\mathsf{Ch}_b(\cdot,\cdot,\cdot)$ oracle, and the rest by signatures of the second identity in B's query. After B outputs a bit d, A returns it as its own result. (If B makes less than I queries to $\mathsf{Ch}(\cdot,\cdot,\cdot)$, A will make a dummy query to its $\mathsf{Ch}_b(\cdot,\cdot,\cdot)$ oracle so that A makes exactly one Ch_b query.) Regarding I as a random variable taking values in $\{1, \ldots, n(k)\}$, this means for every $i \in \{1, \ldots, n(k)\}$,

$$\begin{aligned} \Pr\left[\, \mathbf{Exp}^{\text{anon-1}}_{\mathcal{GS},A}(k) = 1 | I = i \,\right] &= P(i-1) \\ \Pr\left[\, \mathbf{Exp}^{\text{anon-0}}_{\mathcal{GS},A}(k) = 1 | I = i \,\right] &= P(i) \end{aligned} \tag{2}$$

Since the random variable I is uniformly distributed in the range $\{1, \ldots, n(k)\}$ we have

$$\begin{aligned} \Pr\left[\, \mathbf{Exp}^{\text{anon-1}}_{\mathcal{GS},A}(k) = 1 \,\right] &= \sum_{i=1}^{n(k)} \Pr\left[\, \mathbf{Exp}^{\text{anon-1}}_{\mathcal{GS},A}(k) = 1 | I = i \,\right] \cdot \Pr\left[\, I = i \,\right] = \sum_{i=1}^{n(k)} P(i-1) \cdot \frac{1}{n(k)} \\ \Pr\left[\, \mathbf{Exp}^{\text{anon-0}}_{\mathcal{GS},A}(k) = 1 \,\right] &= \sum_{i=1}^{n(k)} \Pr\left[\, \mathbf{Exp}^{\text{anon-0}}_{\mathcal{GS},A}(k) = 1 | I = i \,\right] \cdot \Pr\left[\, I = i \,\right] = \sum_{i=1}^{n(k)} P(i) \cdot \frac{1}{n(k)} \end{aligned} \tag{3}$$

Using the above equations, we have

$$\begin{aligned} \mathbf{Adv}^{\text{anon}}_{\mathcal{GS},A}(k) &= \Pr\left[\, \mathbf{Exp}^{\text{anon-1}}_{\mathcal{GS},A}(k) = 1 \,\right] - \Pr\left[\, \mathbf{Exp}^{\text{anon-0}}_{\mathcal{GS},A}(k) = 1 \,\right] \\ &= \frac{1}{n(k)} \cdot (P(0) - P(n(k))) \\ &= \frac{1}{n(k)} \cdot \mathbf{Adv}^{\text{anon}}_{\mathcal{GS},B}(k) \end{aligned} \tag{4}$$

This completes the proof. ▌

C Proofs of Security Results

These proofs are very similar to those in [5] and we build considerably on the latter.

Algorithm $A_c(pk_e \,:\, \mathsf{Enc}(pk_e, \mathrm{LR}(\cdot,\cdot,b)), \mathsf{Dec}(sk_e,\cdot))$

$(st_{S_1}, R_1) \xleftarrow{\$} \textsc{Sim}_1(\mathsf{gen}, 1^k)$; $(st_{S_2}, R_2) \xleftarrow{\$} \textsc{Sim}_2(\mathsf{gen}, 1^k)$

$(pk_s, sk_s) \xleftarrow{\$} \mathsf{K_s}(1^k)$; $gpk \leftarrow (1^k, R_1, R_2, pk_e, pk_s)$; $ik \leftarrow sk_s$

$\mathrm{CU} \leftarrow \emptyset$; $\mathrm{HU} \leftarrow \emptyset$; $\mathrm{GSet} \leftarrow \emptyset$; $\mathrm{CLIST} \leftarrow \emptyset$; $d \leftarrow \perp$

$d' \xleftarrow{\$} B(gpk, ik \,:\, \mathsf{Ch}_c(\cdot,\cdot,\cdot), \mathsf{Open}(\cdot,\cdot), \mathsf{SndToU}(\cdot,\cdot), \mathsf{WReg}(\cdot,\cdot), \mathsf{USK}(\cdot), \mathsf{CrptU}(\cdot,\cdot))$

If $d \neq \perp$ then return d else return d'

$\mathsf{Ch}_c(m, i_0, i_1)$

Parse gpk as $(1^k, R_1, R_2, pk_e, pk_s)$; Parse $\boldsymbol{gsk}[i_c]$ as $(i_c, pk_{i_c}, sk_{i_c}, \mathrm{cert}_{i_c})$

$s_c \xleftarrow{\$} \mathsf{Sig}(sk_{i_c}, m)$; $M_c \leftarrow \langle i_c, pk_{i_c}, \mathrm{cert}_{i_c}, s_c\rangle$; $M_{\bar{c}} \leftarrow 0^{|M_c|}$

$C \leftarrow \mathsf{Enc}(pk_e, \mathrm{LR}(M_0, M_1, b))$ [oracle query]

$\mathrm{CLIST} \leftarrow \mathrm{CLIST} \cup \{C\}$

$\pi_1 \leftarrow \textsc{Sim}_1(\mathsf{prove}, st_{S_1}, (pk_e, pk_s, m, C))$

Return (C, π_1)

$\mathsf{Open}(m, \sigma)$

Parse σ as (C, π_1)

If $\mathsf{GVf}(gpk, m, \sigma)) = 1$ and $C \in \mathrm{CLIST}$ then $d \leftarrow c$

Run Open algorithm (using $\mathsf{Dec}(sk_e,\cdot)$ oracle to decrypt C, and using $\textsc{Sim}_2$ in place of P_2), and return the result to B

Figure 10: Adversary A_c ($c = 0, 1$) is against the security of the encryption scheme

---

C.1 Proof of Lemma 5.1

By the assumption that P_1 is computational zero-knowledge for ρ_1 over Dom_1, we can fix a simulator $\textsc{Sim}_1$ such that $\Pi_1 = (P_1, V_1, \textsc{Sim}_1)$ is a simulation sound zero knowledge non-interactive proof system for L_{ρ_1}. Similarly we can fix a simulator $\textsc{Sim}_2$ such that $\Pi_2 = (P_2, V_2, \textsc{Sim}_2)$ is a zero knowledge non-interactive proof system for L_{ρ_2}.

We show that for any polynomial time adversary B mounting an attack against anonymity of $\mathcal{GS}$, one can construct polynomial time IND-CCA adversaries A_0, A_1 attacking $\mathcal{AE}$, an adversary A_s against the simulation soundness of Π_1, a distinguisher D_1 that distinguishes simulated proofs from real proofs for Π_1 and a distinguisher D_2 for Π_2, such that for all $k \in \mathbb{N}$

$$\mathbf{Adv}^{\mathrm{anon}}_{\mathcal{GS},B}(k) \leq$$
$$\mathbf{Adv}^{\text{ind-cca}}_{\mathcal{AE},A_0}(k) + \mathbf{Adv}^{\text{ind-cca}}_{\mathcal{AE},A_1}(k) + \mathbf{Adv}^{\mathrm{ss}}_{\Pi,A_s}(k) + 2\cdot\left(\mathbf{Adv}^{\mathrm{zk}}_{\mathrm{P}_1,\textsc{Sim}_1,D_1}(k) + \mathbf{Adv}^{\mathrm{zk}}_{\mathrm{P}_2,\textsc{Sim}_2,D_2}(k)\right) \quad (5)$$

By the assumption on the security of the building blocks of our group signature scheme, all functions on the right side are negligible, therefore so is the function on the left, i.e. our construction is an anonymous group signature scheme.

Following are the details of the constructed adversaries. Unless explicitly specified, these adversaries will answer the oracle queries from B according to Figure 2.

Adversaries against the encryption scheme. The details of adversaries A_0, A_1 against the encryption scheme $\mathcal{AE}$ is given in Figure 10. They are virtually identical, modulo parameter c by which their construction is parametrized, and so is the following description.

Adversary A_c creates an instance for the group signature scheme by generating all keys. The difference from a real group signature scheme is that the public encryption key corresponding to ok is

Algorithm $D_1(1^k, R_1 : \mathsf{Prove}(\cdot,\cdot))$

$r_e \xleftarrow{\$} \{0,1\}^{r(k)}$

$(pk_e, sk_e) \leftarrow \mathsf{K_e}(1^k; r_e)$; $(pk_s, sk_s) \xleftarrow{\$} \mathsf{K_s}(1^k)$

$(st_{S_2}, R_2) \xleftarrow{\$} \textsc{Sim}_2(\mathsf{gen}, 1^k)$

$gpk \leftarrow (1^k, R_1, R_2, pk_e, pk_s)$

$ok \leftarrow (sk_e, r_e)$; $ik \leftarrow sk_s$

$\mathrm{CU} \leftarrow \emptyset$; $\mathrm{HU} \leftarrow \emptyset$; $\mathrm{GSet} \leftarrow \emptyset$; $b \xleftarrow{\$} \{0,1\}$

$d \xleftarrow{\$} B(gpk, ik : \mathsf{Ch}_b(\cdot,\cdot,\cdot), \mathsf{Open}(\cdot,\cdot),$
$\quad \mathsf{SndToU}(\cdot,\cdot), \mathsf{WReg}(\cdot,\cdot), \mathsf{USK}(\cdot), \mathsf{CrptU}(\cdot,\cdot))$

If $d = b$ then return 1 else return 0

$\mathsf{Ch}_b(m, i_0, i_1)$

Parse gpk as $(1^k, R_1, R_2, pk_e, pk_s)$

Parse $\boldsymbol{gsk}[i_b]$ as $(i_b, pk_{i_b}, sk_{i_b}, \mathrm{cert}_{i_b})$

$r \xleftarrow{\$} \{0,1\}^k$; $s \xleftarrow{\$} \mathsf{Sig}(sk_{i_b}, m)$

$C \leftarrow \mathsf{Enc}(pk_e, \langle i_b, pk_{i_b}, \mathrm{cert}_{i_b}, s\rangle; r)$

$\pi_1 \leftarrow \mathsf{Prove}((pk_e, pk_s, m, C),$
$\quad (i_b, pk_{i_b}, \mathrm{cert}_{i_b}, s, r))$ [oracle query]

Return (C, π_1)

$\mathsf{Open}(m, \sigma)$

Run Open algorithm, using $\textsc{Sim}_2$ in place of P_2, and return the result to B

Algorithm $D_2(1^k, R_2 : \mathsf{Prove}(\cdot,\cdot))$

$r_e \xleftarrow{\$} \{0,1\}^{r(k)}$

$(pk_e, sk_e) \leftarrow \mathsf{K_e}(1^k; r_e)$; $(pk_s, sk_s) \xleftarrow{\$} \mathsf{K_s}(1^k)$

$R_1 \xleftarrow{\$} \{0,1\}^{p(k)}$

$gpk \leftarrow (1^k, R_1, R_2, pk_e, pk_s)$

$ok \leftarrow (sk_e, r_e)$; $ik \leftarrow sk_s$

$\mathrm{CU} \leftarrow \emptyset$; $\mathrm{HU} \leftarrow \emptyset$; $\mathrm{GSet} \leftarrow \emptyset$; $b \xleftarrow{\$} \{0,1\}$

$d \xleftarrow{\$} B(gpk, ik : \mathsf{Ch}_b(\cdot,\cdot,\cdot), \mathsf{Open}(\cdot,\cdot),$
$\quad \mathsf{SndToU}(\cdot,\cdot), \mathsf{WReg}(\cdot,\cdot), \mathsf{USK}(\cdot), \mathsf{CrptU}(\cdot,\cdot))$

If $d = b$ then return 1 else return 0

$\mathsf{Ch}_b(m, i_0, i_1)$

Parse gpk as $(1^k, R_1, R_2, pk_e, pk_s)$

Parse $\boldsymbol{gsk}[i_b]$ as $(i_b, pk_{i_b}, sk_{i_b}, \mathrm{cert}_{i_b})$

$r \xleftarrow{\$} \{0,1\}^k$; $s \xleftarrow{\$} \mathsf{Sig}(sk_{i_b}, m)$

$C \leftarrow \mathsf{Enc}(pk_e, \langle i_b, pk_{i_b}, \mathrm{cert}_{i_b}, s\rangle; r)$

$\pi_1 \leftarrow P_1(1^k, (pk_e, pk_s, m, C),$
$\quad (i_b, pk_{i_b}, \mathrm{cert}_{i_b}, s, r), R_1)$

Return (C, π_1)

$\mathsf{Open}(m, \sigma)$

Run Open algorithm, using Prove oracle in place of P_2, and return the result to B

Figure 11: Adversaries D_1, D_2 are distinguishers against the zero knowledge property of the interactive proof systems Π_1, Π_2, respectively

---

obtained from the environment in which A_c is run (the CCA experiment) and that the random strings R_1 and R_2 in the public key of the encryption scheme are obtained by using the simulators $\textsc{Sim}_1$ and $\textsc{Sim}_2$.

Then, A_c runs B against the group signature scheme created this way. In doing so, it needs to answer all opening queries that B may make. This is possible, using the decryption oracle: when a query to the opening oracle is made by B, adversary A_c intercepts this query and checks to see if the signature is valid (this is easy, since A_c possesses gpk.) Then, A_c submits the encrypted part of the signature to the decryption oracle, and from the plaintext, A extracts the identity of the alleged signer and uses $\textsc{Sim}_2$ to generate a proof, which it passes to B.

When B queries (m, i_0, i_1) to Ch_b oracle, A_c creates two challenge plaintexts M_0, M_1, which are computed as follows: M_c is the plaintext of the encrypted part of a group signature on m produced by i_c and $M_{\bar{c}}$ is an all-zero string of length equal to that of M_c.

A_c then queries $\mathsf{Enc}(pk_e, \mathrm{LR}(\cdot,\cdot,b))$, and receives as input a ciphertext C, which is the encryption of one of the two messages, M_0 and M_1. Next, A_c runs the simulator to obtain a proof π_1 of validity for (pk_e, pk_s, m, C). This is always possible, even in the case when C encrypts $M_{\bar{c}}$. The challenge signature that is returned to B is (C, π_1) which A_c now simulates. The final output of A_c is computed as follows: if during this stage B makes a valid query (C, π') to the opening oracle (i.e. it manages to produce a different proof of validity for C), then the guess bit d is set to c, otherwise it is set to whatever B outputs.

Notice that further opening queries of B can be answered by A_c using the decryption oracle (as

Algorithm $A_s(R_1 \ : \ \textsc{Sim}_1(\mathsf{prove}, st_{S_1}, \cdot))$
$r_e \xleftarrow{\$} \{0,1\}^{r(k)}$; $(pk_e, sk_e) \leftarrow \mathsf{K_e}(1^k; r_e)$; $(pk_s, sk_s) \xleftarrow{\$} \mathsf{K_s}(1^k)$
$(st_{S_2}, R_2) \xleftarrow{\$} \textsc{Sim}_2(\mathsf{gen}, 1^k)$; $gpk \leftarrow (1^k, R_1, R_2, pk_e, pk_s)$; $ok \leftarrow (sk_e, r_e)$; $ik \leftarrow sk_s$
CU $\leftarrow \emptyset$; HU $\leftarrow \emptyset$; GSet $\leftarrow \emptyset$; CLIST $\leftarrow \emptyset$; $y \leftarrow \perp$
$B(gpk, ik \ : \ \mathsf{Ch}(\cdot,\cdot,\cdot), \mathsf{Open}(\cdot,\cdot), \mathsf{SndToU}(\cdot,\cdot), \mathsf{WReg}(\cdot,\cdot), \mathsf{USK}(\cdot), \mathsf{CrptU}(\cdot,\cdot))$
Return y

$\mathsf{Ch}(m, i_0, i_1)$
Parse gpk as $(1^k, R_1, R_2, pk_e, pk_s)$; Parse $\boldsymbol{gsk}[i_1]$ as $(i_1, pk_{i_1}, sk_{i_1}, \mathrm{cert}_{i_1})$
$s_1 \xleftarrow{\$} \mathsf{Sig}(sk_{i_1}, m)$; $M_1 \leftarrow \langle i_1, pk_{i_1}, \mathrm{cert}_{i_1}, s_1 \rangle$; $M_0 \leftarrow 0^{|M_1|}$
$C \leftarrow \mathsf{Enc}(pk_e, M_0))$; CLIST $\leftarrow$ CLIST $\cup \{C\}$
$\pi_1 \leftarrow \textsc{Sim}_1(\mathsf{prove}, \mathrm{St}_{S_1}, (pk_e, pk_s, m, C))$ [oracle query]
Return (C, π_1)

$\mathsf{Open}(m, \sigma)$
Parse σ as (C, π_1)
If $\mathsf{GVf}(gpk, m, \sigma)) = 1$ and $C \in$ CLIST then $y \leftarrow ((pk_e, pk_s, m, C), \pi_1)$
Run Open algorithm, using $\textsc{Sim}_2$ in place of P_2, and return the result to B

Figure 12: Adversary A_s is against simulation-soundness of Π_1

---

described above). However, we have to make sure that the challenge ciphertext C is never queried to the decryption oracle. This is true, since whenever a valid query (C, π') is issued by B to the opening oracle, adversary A_c, instead of submitting C to the decryption oracle, simply outputs c and terminates.

The distinguishers for zero-knowledge. The distinguisher D_1 (given in Figure 11), also starts out by creating an instance of the group signature scheme $\mathcal{GS}$. The keys for the encryption schemes, the individual signing keys and the keys for certifying/verifying individual public keys are obtained by running the respective key generation algorithms. The string R_1 that is part of the public key is supplied to D_1 by the environment in which it is run. D_1 uses simulator $\textsc{Sim}_2$ to generate the reference string R_2.

Then, by running B against the group signature scheme, it obtains a message m and two identities i_0, i_1 for which B claims it can distinguish group signatures on m. When B makes a query to the opening oracle, D_1 runs the Open algorithm normally, except that it uses simulator $\textsc{Sim}_2$ to generate the proof π_2.

The challenge group signature that D_1 passes to to B in response to B's query to Ch is created as follows. One of the two signers i_0 and i_1 is chosen, by flipping uniformly at random a bit b, and the plaintext corresponding to a signature on m created by i_b is encrypted under the public key of the group manager. D_1 queries its oracle $(pk_s, pk_e, m, C) \in L_{\rho_1}$ together with the corresponding witness, and receives a proof π_1. D_1 then creates a group signature (C, π_1) on m and feeds it to B. The final output of D_1 is whatever B outputs.

The distinguisher D_2 has structure similar to D_1 except for three major differences. (1) The string R_2 that is part of the public key is supplied to D_2 by the environment in which it is run. However, the reference string R_1 is a true random string. (2) When B makes a query to the opening oracle, D_2 uses its Prove oracle instead of $\textsc{Sim}_2$ to generate π_2. (3) In Ch oracle, D_2 uses P_1 to generate π_1.

The simulation-soundness adversary. The adversary A_s against the simulation soundness is given in Figure 12. It creates an instance for the group signature scheme $\mathcal{GS}$, by generating *all* keys.

The only difference from a "real" group signature scheme is that the random string R_1 that is part of the public key is obtained from the environment in which A_s runs (i.e. it is generated by the simulator), and R_2 is generated by simulator $\textsc{Sim}_2$. Then A_s runs B against the group signature scheme. When B queries Ch with message m and the two identities i_0 and i_1, the challenge signature (C, π_1) that A_s passes to B is such that C is the encryption of the all-zero string (of appropriate length) and π_1 is a simulated proof of validity. When B queries the opening oracle, A_s runs the Open algorithm normally but uses $\textsc{Sim}_2$ in place of P_2. Finally, A_s tracks the queries that B makes to the opening oracle: if B makes a valid query (C, π') then A_s outputs $((pk_e, pk_s, m, C), \pi')$, otherwise it fails.

Putting it all together. We now explain how to relate the advantages of the four adversaries described above with the advantage of B (the adversary against full-anonymity that they all run as a subroutine.) We start with the distinguisher.

Recall that under experiment $\mathbf{Exp}^{\text{zk-1}}_{P_2,\textsc{Sim}_2,D_2}(k)$, the string R_2 passed to D_2 is an actual random string; so D_2 runs B against a group signature scheme generated according to the key generation algorithm GKg. If (m, i_0, i_1) is the query made by B to Ch, the signature returned to B is the real signature of i_b, where b is chosen at random. So, D_2 outputs 1 exactly when B guesses correctly which user produced the signature, i.e. it wins in $\mathbf{Exp}^{\text{anon-}b}_{\mathcal{GS},B}(k)$ no matter what D_2's choice of b is. We can formalize the above as follows:

$$
\begin{aligned}
&\Pr\left[\mathbf{Exp}^{\text{zk-1}}_{P_2,\textsc{Sim}_2,D_2}(k) = 1\right] \\
&= \Pr[\, B \text{ returns } 1 \mid b = 1\,] \cdot \Pr[\, b = 1\,] + \Pr[\, B \text{ returns } 0 \mid b = 0\,] \cdot \Pr[\, b = 0\,] \\
&= \frac{1}{2}\Pr\left[\mathbf{Exp}^{\text{anon-1}}_{\mathcal{GS},B}(k) = 1\right] + \frac{1}{2}\Pr\left[\mathbf{Exp}^{\text{anon-0}}_{\mathcal{GS},B}(k) = 0\right] \\
&= \frac{1}{2}\Pr\left[\mathbf{Exp}^{\text{anon-1}}_{\mathcal{GS},B}(k) = 1\right] + \frac{1}{2}\left(1 - \Pr\left[\mathbf{Exp}^{\text{anon-0}}_{\mathcal{GS},B}(k) = 1\right]\right) \\
&= \frac{1}{2} + \frac{1}{2}\mathbf{Adv}^{\text{anon}}_{\mathcal{GS},B}(k)
\end{aligned} \tag{6}
$$

Notice that the experiments $\mathbf{Exp}^{\text{zk-1}}_{P_1,\textsc{Sim}_1,D_1}(k)$ and $\mathbf{Exp}^{\text{zk-0}}_{P_2,\textsc{Sim}_2,D_2}(k)$ are identical. In both experiments, the reference string R_1 is a true random string and R_2 is generated by simulator $\textsc{Sim}_2$. In $\mathbf{Exp}^{\text{zk-1}}_{P_1,\textsc{Sim}_1,D_1}(k)$, the Prove oracle D_1 queries is in fact P_1. In $\mathbf{Exp}^{\text{zk-0}}_{P_2,\textsc{Sim}_2,D_2}(k)$, the Prove oracle D_2 queries is $\textsc{Sim}_2$. Therefore,

$$
\Pr\left[\mathbf{Exp}^{\text{zk-1}}_{P_1,\textsc{Sim}_1,D_1}(k) = 1\right] = \Pr\left[\mathbf{Exp}^{\text{zk-0}}_{P_2,\textsc{Sim}_2,D_2}(k) = 1\right] \tag{7}
$$

The success of D_1 under experiment $\mathbf{Exp}^{\text{zk-0}}_{P_1,\textsc{Sim}_1,D_1}(k)$ is related to the advantages of adversaries A_0, A_1 and A_s as follows (readers are referred to [5] for the detailed discussion and derivation):

$$
2 \cdot \Pr\left[\mathbf{Exp}^{\text{zk-0}}_{P_1,\textsc{Sim}_1,D_1}(k) = 1\right] \leq \mathbf{Adv}^{\text{ind-cca}}_{\mathcal{AE},A_1}(k) + \mathbf{Adv}^{\text{ind-cca}}_{\mathcal{AE},A_0}(k) + \mathbf{Adv}^{\text{ss}}_{\Pi_1,A_s}(k)
$$

We can now calculate sum of the advantages of distinguishers D_1 and D_2, by combining the above equation with equations (6) and (7):

$$
\begin{aligned}
&2 \cdot \left(\mathbf{Adv}^{\text{zk}}_{P_1,\textsc{Sim}_1,D_1}(k) + \mathbf{Adv}^{\text{zk}}_{P_2,\textsc{Sim}_2,D_2}(k)\right) \\
&= 2 \cdot \left(\Pr\left[\mathbf{Exp}^{\text{zk-1}}_{P_1,\textsc{Sim}_1,D_1}(k) = 1\right] - \Pr\left[\mathbf{Exp}^{\text{zk-0}}_{P_1,\textsc{Sim}_1,D_1}(k) = 1\right]\right. \\
&+ \left.\Pr\left[\mathbf{Exp}^{\text{zk-1}}_{P_2,\textsc{Sim}_2,D_2}(k) = 1\right] - \Pr\left[\mathbf{Exp}^{\text{zk-0}}_{P_2,\textsc{Sim}_2,D_2}(k) = 1\right]\right) \\
&\geq \mathbf{Adv}^{\text{anon}}_{\mathcal{GS},B}(k) - \mathbf{Adv}^{\text{ind-cca}}_{\mathcal{AE},A_0}(k) - \mathbf{Adv}^{\text{ind-cca}}_{\mathcal{AE},A_1}(k) - \mathbf{Adv}^{\text{ss}}_{\Pi_1,A_s}(k)
\end{aligned}
$$

and by rearranging the terms we obtain Equation (5) as desired.

$$\begin{array}{lcl} F_1 & : & \mathsf{GVf}(gpk, m, \sigma) = 1 \wedge i = 0. \\ F_2 & : & \mathsf{GVf}(gpk, m, \sigma) = 1 \wedge i > 0 \wedge \mathsf{Judge}(gpk, i, \boldsymbol{upk}[i], m, \sigma, \tau) = 0 \\ F & : & F \text{ denotes the event } F_1 \vee F_2 \\ S & : & (pk_e, pk_s, m, C) \in L_{\rho_1} \end{array}$$

Figure 13: Events considered in the proof of traceability

C.2 Proof of Lemma 5.2

Let B be a traceability adversary against $\mathcal{GS}$. We define an adversary A_1 against digital signature scheme $\mathcal{DS}$, as specified in Figure 14. Then, using the assumption that (P_1, V_1) is a sound proof system for ρ_1, we claim that

$$\mathbf{Adv}^{\text{trace}}_{\mathcal{GS},B}(k) \leq 2^{-k} + \mathbf{Adv}^{\text{unforg-cma}}_{\mathcal{DS},A_1}(k) .$$

Since we assume that the digital signature scheme $\mathcal{DS}$ is secure, it follows that the right hand side of the inequality is a negligible function (of the security parameter) so, the advantage function on the left is also negligible. Thus, according to the definition, $\mathcal{GS}$ is a traceable group signature scheme.

Adversary A_1 is intended to run in the experiment $\mathbf{Exp}^{\text{unforg-cma}}_{\mathcal{DS},A_1}(k)$, defining the security of digital signature $\mathcal{DS}$. As such, it has access to a signing oracle $\mathsf{Sig}(sk, \cdot)$, and is given as input the corresponding verification key pk. It starts out by construction an instance of the group signature scheme $\mathcal{GS}$ as follows. It generates the opening key sk_e together with the corresponding key pk_e. A_1 answers most of the oracles according to Figure 2 except for two oracles: SndToI and AddU. When answering these two oracles for identity i, instead of certifying $\langle i, pk_i \rangle$ using ik, A_1 queries its own oracle $\mathsf{Sig}(sk, \cdot)$, and sends the result back to i as the certificate.

Let (m, σ) denote the output of B where $\sigma = (C, \pi_1)$. Let (i, τ) be the output of $\mathsf{Open}(gpk, ok, \boldsymbol{reg}, m, \sigma)$. Consider the events in Figure 13. Notice that the events F_1 and F_2 are disjoint. The advantage of the adversary B can be bounded by

$$\mathbf{Adv}^{\text{trace}}_{\mathcal{GS},B}(k) = \Pr[\,F\,] = \Pr\left[\,F \wedge \overline{S}\,\right] + \Pr[\,F_1 \wedge S\,] + \Pr[\,F_2 \wedge S\,] . \tag{8}$$

We now bound each of these terms.

Claim C.1 $\Pr\left[\,F \wedge \overline{S}\,\right] \leq 2^{-k}$.

Proof: The event F implies $\mathsf{GVf}(gpk, m, \sigma) = 1$, i.e. $V_1((m, C), \pi_1) = 1$. Since (P_1, V_1) is a sound proof system for ρ_1, for any security parameter k, and any polynomial time forger B,

$$\Pr\left[\,F \wedge \overline{S}\,\right] \leq \Pr\left[\,(pk_e, pk_s, m, C) \notin L_{\rho_1} \text{ and } V_1((m, C), \pi_1) = 1\,\right] \leq 2^{-k} \tag{9}$$

as desired. ▮

Claim C.2 $\Pr[\,F_1 \wedge S\,] \leq \mathbf{Adv}^{\text{unforg-cma}}_{\mathcal{DS},A_1}(k)$.

Proof: Suppose B outputs a successful forgery $(m, (C, \pi_1), \tau)$ such that

- $(pk_e, pk_s, m, C) \in L_{\rho_1}$, and
- $\mathsf{Open}(gpk, ok, \boldsymbol{reg}, m, \sigma)$ returns $i = 0$.

Adversary $A_1^{\mathsf{Sig}(sk,\cdot)}(pk)$
  $R_1 \xleftarrow{\$} \{0,1\}^{p_1(k)}$
  $R_2 \xleftarrow{\$} \{0,1\}^{p_2(k)}$
  $r_e \xleftarrow{\$} \{0,1\}^{r(k)}$
  $(pk_e, sk_e) \leftarrow \mathsf{K_e}(1^k; r_e)$
  $pk_s \leftarrow pk$; $sk_s \leftarrow \perp$
  HU $\leftarrow \emptyset$; CU $\leftarrow \emptyset$
  $gpk \leftarrow (1^k, R_1, R_2, pk_e, pk_s)$
  $ok \leftarrow (sk_e, r_e)$
  $ik \leftarrow \perp$
  $(m, \sigma) \leftarrow B(gpk, ok : \mathsf{SndToI}(\cdot,\cdot), \mathsf{AddU}(\cdot),$
    $\mathsf{RReg}(\cdot), \mathsf{USK}(\cdot), \mathsf{CrptU}(\cdot,\cdot))$
  Parse σ as (C, π_1)
  $M \leftarrow \mathsf{Dec}(sk_e, C)$
  Parse M as $\langle i, pk', \mathrm{cert}', s\rangle$
  Return $(\langle i, pk'\rangle, \mathrm{cert}')$

$\mathsf{SndToI}(i, M_{in})$
  If $i \notin$ CU then return ε
  $M_{out} \leftarrow \varepsilon$; $dec' \leftarrow \mathsf{reject}$
  If $dec^i = \mathsf{cont}$ then
    Parse St^i_{iss} as (gpk, ik, i, upk_i)
    Parse M_{in} as (pk_i, sig_i)
    If $\mathsf{Vf}(upk_i, pk_i, sig_i) = 1$ then
      $\mathrm{cert}_i \leftarrow \mathsf{Sig}(sk, \langle i, pk_i\rangle)$ [oracle query]
      $\boldsymbol{reg}[i] \leftarrow (pk_i, sig_i)$
      $M_{out} \leftarrow \mathrm{cert}_i$; $dec' \leftarrow \mathsf{accept}$
  $dec^i \leftarrow dec'$; return M_{out}

$\mathsf{AddU}(i)$
  If $i \in$ CU or $i \in$ HU then return ε
  HU $\leftarrow$ HU $\cup \{i\}$; $(\boldsymbol{upk}[i], \boldsymbol{usk}[i]) \xleftarrow{\$} \mathsf{UKg}(1^k)$
  $(pk_i, sk_i) \xleftarrow{\$} \mathsf{K_s}(1^k)$; $sig_i \leftarrow \mathsf{Sig}(\boldsymbol{usk}[i], pk_i)$
  $\mathrm{cert}_i \leftarrow \mathsf{Sig}(sk, \langle i, pk_i\rangle)$ [oracle query]
  $\boldsymbol{reg}[i] \leftarrow (pk_i, sig_i)$
  $\boldsymbol{gsk}[i] \leftarrow (i, pk_i, sk_i, \mathrm{cert}_i)$
  Return $\boldsymbol{upk}[i]$

Figure 14: Construction of cma-forger A_1 against $\mathcal{DS}$ from an adversary B against traceability of $\mathcal{GS}$.

Let $\mathsf{Dec}(sk_e, C) = \langle i, pk', \mathrm{cert}', s\rangle$ be the plaintext of C, and (pk_i, sig_i) be the contents of $\boldsymbol{reg}[i]$. It follows that $\mathsf{Vf}(pk, \langle i, pk'\rangle, \mathrm{cert}') = 1$, and either $pk' \neq pk_i$ or $\boldsymbol{reg}[i] = \varepsilon$. Since A_1 has never queried $\langle i, pk'\rangle$ to its oracle $\mathsf{Sig}(sk,\cdot)$, $(\langle i, pk'\rangle, \mathrm{cert}')$ is a successful forgery of A_1 in the $\mathbf{Exp}^{\text{unforg-cma}}_{\mathcal{DS},A_1}(k)$ experiment, and

$$\mathbf{Adv}^{\text{unforg-cma}}_{\mathcal{DS},A_1}(k) \quad \geq \quad \Pr[\,F_1 \wedge S\,] \tag{10}$$

as desired. ▮

Claim C.3 $\Pr[\,F_2 \wedge S\,] = 0$.

Proof: Let (i, τ) be the output of $\mathsf{Open}(gpk, ok, \boldsymbol{reg}, m, \sigma)$, where $\sigma = (C, \pi_1)$. The event F_2 implies $i > 0$. Let $M = \mathsf{Dec}(sk_e, C)$ be the plaintext of C, and M is parsed as $\langle i, pk, \mathrm{cert}, s\rangle$. Consider the pseudocode of the Open algorithm. Since Open returns $i > 0$, we have $\boldsymbol{reg}[i] \neq \varepsilon$, and $\boldsymbol{reg}[i]$ can be parsed as (pk_i, sig_i). Furthermore, it must be true that $pk = pk_i$, otherwise Open will return $i = 0$.

Now consider the Judge algorithm given parameters $gpk, i, \boldsymbol{upk}[i], m, \sigma, \tau$. Notice that the specific parameters m, σ are the same as the input of Open in the previous paragraph, and τ is the proof Open outputs.

First, Judge checks whether $V_2(1^k, (C, i, pk, \mathrm{cert}, s), \pi_2, R_2) = 1$. Since $(pk_e, C, i, pk, \mathrm{cert}, s) \in L_{\rho_2}$, and Open generated π_2 by running the prover P_2 on $(pk_e, C, i, pk, \mathrm{cert}, s)$, this satisfies. Next, Judge will check whether the identity i in its own parameter list is identical to the identity i' in τ. Again, this is true, since the i' that Open put in τ is exactly the i that Open returns. Third, Judge will check whether $\mathsf{Vf}(\boldsymbol{upk}[i], pk_i, sig_i) = 1$. This is true since pk_i, sig_i were written to the $\boldsymbol{reg}$ table by the

Adversary $A_2^{\mathsf{Sig}(sk,\cdot)}(pk)$
$R_1 \xleftarrow{\$} \{0,1\}^{p_1(k)}$
$R_2 \xleftarrow{\$} \{0,1\}^{p_2(k)}$
$r_e \xleftarrow{\$} \{0,1\}^{r(k)}$
$(pk_s, sk_s) \xleftarrow{\$} \mathsf{K_s}(1^k)$
$(pk_e, sk_e) \leftarrow \mathsf{K_e}(1^k; r_e)$
$t \xleftarrow{\$} \{1, 2, \ldots, n(k)\}$
$cnt \leftarrow 0$; $u \leftarrow \perp$
$\mathrm{HU} \leftarrow \emptyset$; $\mathrm{CU} \leftarrow \emptyset$
$gpk \leftarrow (1^k, R_1, R_2, pk_e, pk_s)$
$ok \leftarrow (sk_e, r_e)$; $ik \leftarrow sk_s$
$(m, \sigma, i, \tau) \leftarrow B(gpk, ok, ik : \mathsf{SndToU}(\cdot,\cdot),$
$\quad \mathsf{WReg}(\cdot,\cdot), \mathsf{GSig}(\cdot,\cdot), \mathsf{USK}(\cdot), \mathsf{CrptU}(\cdot,\cdot))$
Parse τ as $(\overline{pk}, \overline{sig}, i, pk', \mathrm{cert}', s, \pi_2)$
Return (m, s)

$\mathsf{USK}(i)$
If $i = u$ then fail
Run the normal USK oracle

$\mathsf{SndToU}(i, M)$
If $i \notin \mathrm{HU}$ then
$\quad \mathrm{HU} \leftarrow \mathrm{HU} \cup \{i\}$; $cnt \leftarrow cnt + 1$
$\quad (\boldsymbol{upk}[i], \boldsymbol{usk}[i]) \xleftarrow{\$} \mathsf{K_s}(1^k)$
$\quad$ If $cnt = t$ then
$\quad\quad u \leftarrow i$; $pk_i \leftarrow pk$; $sk_i \leftarrow \perp$
$\quad$ Else $(pk_i, sk_i) \xleftarrow{\$} \mathsf{K_s}(1^k)$
$\quad sig_i \leftarrow \mathsf{Sig}(\boldsymbol{usk}[i], pk_i)$; $\mathrm{St}_{jn} \leftarrow (pk_i, sk_i)$
$\quad$ Return $(\langle pk_i, sig_i \rangle, \mathsf{true})$
Else run the normal SndToU oracle

$\mathsf{GSig}(i, m)$
Parse gpk as $(1^k, R_1, R_2, pk_e, pk_s)$
If $i = u$ then
$\quad$ Parse $\boldsymbol{gsk}[i]$ as $(i, pk_i, sk_i, \mathrm{cert}_i)$
$\quad s \xleftarrow{\$} \mathsf{Sig}(sk, m)$ [oracle query]
$\quad r \xleftarrow{\$} \{0,1\}^k$; $C \xleftarrow{\$} \mathsf{Enc}(pk_e, \langle i, pk, \mathrm{cert}_i, s \rangle; r)$
$\quad \pi_1 \xleftarrow{\$} P_1(1^k, (pk_e, pk_s, m, C),$
$\quad\quad (i, pk, \mathrm{cert}_i, s, r), R_1)$
$\quad$ Return (C, π_1)
Else run the normal GSig oracle

Figure 15: Construction of cma-forger A_2 against $\mathcal{DS}$ from an adversary B against non-frameability of $\mathcal{GS}$. Here $n(\cdot)$, a polynomial in k, is the number of honest users that B creates.

---

issuer during the Join protocol only if $\mathsf{Vf}(\boldsymbol{upk}[i], pk_i, sig_i) = 1$. Finally, Judge will check if $pk_i = pk$, which we already know is true from our previous discussion about Open. After checking all these conditions, it is necessary that Judge return 1. This proves the claim. ▮

Using the Claims, we can bound the advantage of B as desired:

$$\mathbf{Adv}^{\mathrm{trace}}_{\mathcal{GS},B}(k) \le 2^{-k} + \mathbf{Adv}^{\mathrm{unforg\text{-}cma}}_{\mathcal{DS},A_1}(k) .$$

C.3 Proof of Lemma 5.3

Let B be a non-frameability adversary against $\mathcal{GS}$ who creates at most $n(k)$ honest users, where n is a polynomial. We define adversaries A_2, A_3 against digital signature scheme $\mathcal{DS}$, as specified in Figures 15 and 16. Then, using the assumption that $(P_1, V_1), (P_2, V_2)$ are sound proof systems for ρ_1, ρ_2 respectively, we claim that

$$\mathbf{Adv}^{\mathrm{nf}}_{\mathcal{GS},B}(k) \le 2^{-k+1} + n(k) \cdot \left(\mathbf{Adv}^{\mathrm{unforg\text{-}cma}}_{\mathcal{DS},A_2}(k) + \mathbf{Adv}^{\mathrm{unforg\text{-}cma}}_{\mathcal{DS},A_3}(k) \right) .$$

Since we assume that the digital signature scheme $\mathcal{DS}$ is secure, it follows that the right hand side of the inequality is a negligible function (of the security parameter) so, the advantage function on the left is also negligible. Thus, according to the definition, $\mathcal{GS}$ is a non-frameable group signature scheme.

Adversary A_2 is intended to run in the experiment $\mathbf{Exp}^{\mathrm{unforg\text{-}cma}}_{\mathcal{DS},A_2}(k)$, defining the security of digital signature $\mathcal{DS}$. As such, it has access to a signing oracle $\mathsf{Sig}(sk, \cdot)$, and is given as input the corresponding verification key pk. It starts out by construction an instance of the group signature

Adversary $A_3^{\mathsf{Sig}(sk,\cdot)}(pk)$

$R_1 \xleftarrow{\$} \{0,1\}^{p_1(k)}$; $R_2 \xleftarrow{\$} \{0,1\}^{p_2(k)}$; $r_e \xleftarrow{\$} \{0,1\}^{r(k)}$

$(pk_e, sk_e) \leftarrow \mathsf{K_e}(1^k; r_e)$; $(pk_s, sk_s) \xleftarrow{\$} \mathsf{K_s}(1^k)$

$t \xleftarrow{\$} \{1, 2, \ldots, n(k)\}$; $cnt \leftarrow 0$; $u \leftarrow \perp$

$\mathrm{HU} \leftarrow \emptyset$; $\mathrm{CU} \leftarrow \emptyset$; $gpk \leftarrow (1^k, R_1, R_2, pk_e, pk_s)$

$ok \leftarrow (sk_e, r_e)$; $ik \leftarrow sk_s$

$(m, \sigma, i, \tau) \leftarrow B(gpk, ok, ik : \mathsf{SndToU}(\cdot,\cdot),$
$\mathsf{WReg}(\cdot,\cdot), \mathsf{GSig}(\cdot,\cdot), \mathsf{USK}(\cdot), \mathsf{CrptU}(\cdot,\cdot))$

Parse τ as $(\overline{pk}, \overline{sig}, i, pk', \mathrm{cert}', s, \pi_2)$

Return $(\overline{pk}, \overline{sig})$

$\mathsf{USK}(i)$

If $i = u$ then fail else run the normal USK oracle

$\mathsf{SndToU}(i, M)$

If $i \notin \mathrm{HU}$ then

$\mathrm{HU} \leftarrow \mathrm{HU} \cup \{i\}$; $cnt \leftarrow cnt + 1$

$(pk_i, sk_i) \xleftarrow{\$} \mathsf{K_s}(1^k)$

If $cnt = t$ then

$u \leftarrow i$; $\boldsymbol{upk}[i] \leftarrow pk$

$sig_i \leftarrow \mathsf{Sig}(sk, pk_i)$ [oracle query]

Else

$(\boldsymbol{upk}[i], \boldsymbol{usk}[i]) \xleftarrow{\$} \mathsf{K_s}(1^k)$

$sig_i \leftarrow \mathsf{Sig}(\boldsymbol{usk}[i], pk_i)$

$\mathrm{St}_{jn} \leftarrow (pk_i, sk_i)$

Return $(\langle pk_i, sig_i \rangle, \mathsf{true})$

Else run the normal SndToU oracle

Figure 16: Construction of cma-forger A_3

scheme $\mathcal{GS}$ as follows. It generates the certification key sk_s together with the corresponding key pk_s and also the opening key sk_e together with the corresponding key pk_e. Suppose adversary B creates at most $n(k)$ honest users, then A_2 will randomly choose an integer $t \in \{1, 2, \ldots, n(k)\}$. Let u be the identity of the t-th honest user that B creates. When answering B's SndToU queries, A_2 generates $(\boldsymbol{upk}[j], \boldsymbol{usk}[j])$ for all honest users. A_2 also generates all signing-verifying keys (pk_j, sk_j) except those for user u. The individual signing key for u will be the signing key sk of the oracle to which A_2 has access.

Next A_2 runs B against $\mathcal{GS}$ created this way and thus needs to be able to answer all oracle queries that B may make. Requests for signatures on messages can be easily answered. Requests of the type (j, m), with $j \neq u$, are answered by using $\boldsymbol{gsk}[j]$ (which it A_2 created by itself). Requests of the type (j, m) with $j = u$ are handled by first submitting m to the signing oracle, thus obtaining $s = \mathsf{Sig}(sk, m)$, and then by creating the rest of the group signature by itself. Finally, A_2 can also answer all requests for the group signing secret keys of group members (notice that $\boldsymbol{gsk}[i]$ is not requested, since otherwise opening the signature would lead to a member of the set of corrupted members and the forgery would not be successful).

Adversary A_3 is intended to run in the experiment $\mathbf{Exp}^{\text{unforg-cma}}_{\mathcal{DS},A_3}(k)$, defining the security of digital signature $\mathcal{DS}$. A_3 has structure similar to that of A_2 except for three major differences. (1) When answering B's SndToU queries, A_3 generates (pk_j, sk_j) for all honest users j. A_3 also generates all user signing-verifying keys $(\boldsymbol{upk}[j], \boldsymbol{usk}[j])$ and user signatures sig_j except those for user u. The user signing key $\boldsymbol{usk}[u]$ will be the signing key sk of the oracle to which A_3 has access. The user signature sig_u will be obtained by querying A_3's signing oracle. (2) When answering B's GSig queries, A_3 will run the normal GSig oracle in Figure 2. (3) After B outputs a successful forgery (m, σ, i, τ), A_3 extracts $(\overline{pk}, \overline{sig})$ from τ and returns it.

Let $(m, (C, \pi_1), i, \tau)$ denote the output of B, where $\tau = (\overline{pk}, \overline{sig}, i, pk', \mathrm{cert}', s, \pi_2)$. Consider the events listed in Figure 17. Then the advantage of the adversary B can be bounded by

$$\mathbf{Adv}^{\text{nf}}_{\mathcal{GS},B}(k) = \Pr[F] \le \Pr\left[F \wedge \overline{S_1}\right] + \Pr\left[F \wedge \overline{S_2}\right] + \Pr[F \wedge P \wedge S_1 \wedge S_2] + \Pr\left[F \wedge \overline{P} \wedge S_1 \wedge S_2\right] .$$

We now bound each of these terms.

Claim C.4 $\Pr\left[F \wedge \overline{S_1}\right] \le 2^{-k}$.

$$\begin{array}{rcl} F & : & \begin{array}{l}\mathsf{GVf}(gpk, m, \sigma) = 1 \wedge i \in \mathrm{HU} \wedge \boldsymbol{gsk}[i] \neq \varepsilon \wedge \\ B \text{ did not query } \mathsf{USK}(i) \text{ or } \mathsf{GSig}(i, m) \wedge \mathsf{Judge}(gpk, 1, i, upk[i], m, \sigma, \tau) = 1\end{array} \\ S_1 & : & (pk_e, pk_s, m, C) \in L_{\rho_1} \\ S_2 & : & (pk_e, C, i, pk', \mathrm{cert}', s) \in L_{\rho_2} \\ P & : & \begin{array}{l}\text{Given that } \boldsymbol{gsk}[i] \neq \varepsilon, \text{ it must have the form } (i, pk_i, sk_i, \mathrm{cert}_i). \\ \text{Define the event } P \text{ as } pk' = pk_i\end{array} \end{array}$$

Figure 17: Events considered in the proof of non-frameability

---

Proof: The event F implies $\mathsf{GVf}(gpk, m, \sigma) = 1$, i.e. $V_1((m, C), \pi_1) = 1$. Since (P_1, V_1) is a sound proof system for ρ_1, for any security parameter k, and any polynomial time forger B,

$$\Pr\left[\, F \wedge \overline{S_1} \,\right] \leq \Pr\left[\, (pk_e, pk_s, m, C) \notin L_{\rho_1} \text{ and } V_1((m, C), \pi_1) = 1 \,\right] \leq 2^{-k} \tag{11}$$

as desired. ▮

Claim C.5 $\Pr\left[\, F \wedge \overline{S_2} \,\right] \leq 2^{-k}$.

Proof: The event F implies $\mathsf{Judge}(gpk, 1, i, upk[i], m, \sigma, \tau) = 1$, i.e. $V_2((pk_e, C, i, pk', \mathrm{cert}', s), \pi_2) = 1$. Since (P_2, V_2) is a sound proof system for ρ_2, for any security parameter k, and any polynomial time forger B,

$$\Pr\left[\, F \wedge \overline{S_2} \,\right] \leq \Pr\left[\, (pk_e, C, i, pk', \mathrm{cert}', s) \notin L_{\rho_2} \text{ and } V_2((pk_e, C, i, pk', \mathrm{cert}', s), \pi_2) = 1 \,\right] \leq 2^{-k} \tag{12}$$

as desired. ▮

Claim C.6 $\Pr\left[\, F \wedge P \wedge S_1 \wedge S_2 \,\right] \leq n(k) \cdot \mathbf{Adv}^{\text{unforg-cma}}_{\mathcal{DS}, A_2}(k)$.

Proof: Suppose B outputs a successful forgery $(m, (C, \pi_1), i, \tau)$ where $\tau = (\overline{pk}, \overline{sig}, i, pk', \mathrm{cert}', s, \pi_2)$, such that $(pk_e, pk_s, m, C) \in L_{\rho_1}$, $(pk_e, C, i, pk', \mathrm{cert}', s) \in L_{\rho_2}$ and $pk' = pk_i$, and at the same time, A_2 correctly guesses the identity i. This implies that s is valid signature on m under pk_i. Furthermore, (i, m) is not submitted by B to its signing oracle $\mathsf{GSig}(\cdot, \cdot)$, (otherwise it is not a valid forgery), which in particular means m was not queried to the signing oracle $\mathsf{Sig}(sk, \cdot)$ to which A_2 has access. Altogether, this amounts to the fact that (m, s) is a successful forgery of A_2 in the $\mathbf{Exp}^{\text{unforg-cma}}_{\mathcal{DS}, A_2}(k)$ experiment.

By observing that A_2 guesses the identity i correctly with probability $1/n(k)$, which is independent of the events F, S_1, S_2 and P, we obtain:

$$\mathbf{Adv}^{\text{unforg-cma}}_{\mathcal{DS}, A_2}(k) \quad \geq \quad \frac{1}{n(k)} \cdot \Pr\left[\, F \wedge P \wedge S_1 \wedge S_2 \,\right] \tag{13}$$

as desired. ▮

Claim C.7 $\Pr\left[\, F \wedge \overline{P} \wedge S_1 \wedge S_2 \,\right] \leq n(k) \cdot \mathbf{Adv}^{\text{unforg-cma}}_{\mathcal{DS}, A_3}(k)$.

Proof: Suppose B outputs a successful forgery $(m, (C, \pi_1), i, \tau)$ where $\tau = (\overline{pk}, \overline{sig}, i, pk', \mathrm{cert}', s, \pi_2)$, such that $(pk_e, pk_s, m, C) \in L_{\rho_1}$, $(pk_e, C, i, pk', \mathrm{cert}', s) \in L_{\rho_2}$ and $pk' \neq pk_i$, and at the same time, A_3 correctly guesses the identity i. B wins the non-frameability experiment, therefore

$$\mathsf{Judge}(gpk, 1, i, upk[i], m, \sigma, \tau) = 1$$

which implies $\overline{pk} = pk'$, and $(\overline{pk}, \overline{sig})$ in τ satisfies $\mathsf{Vf}(pk, \overline{pk}, \overline{sig}) = 1$. Furthermore, since pk_i is the only query made by A_3 to its signing oracle, $\overline{pk}$ $(= pk', \neq pk_i)$ was not queried to the $\mathsf{Sig}(sk, \cdot)$. Altogether, this amounts to the fact that $(\overline{pk}, \overline{sig})$ is a successful forgery of A_3 in the $\mathbf{Exp}^{\text{unforg-cma}}_{\mathcal{DS},A_3}(k)$ experiment.

By observing that A_3 guesses the identity i correctly with probability $1/n(k)$, which is independent of the events F, S_1, S_2 and $\overline{P}$, we obtain:

$$\mathbf{Adv}^{\text{unforg-cma}}_{\mathcal{DS},A_3}(k) \quad \geq \quad \frac{1}{n(k)} \cdot \Pr\left[\, F \wedge \overline{P} \wedge S_1 \wedge S_2 \,\right] \tag{14}$$

as desired. ▮

Using the Claims we can bound the advantage of B as desired:

$$\mathbf{Adv}^{\text{nf}}_{\mathcal{GS},B}(k) \leq 2^{-k+1} + n(k) \cdot \left(\mathbf{Adv}^{\text{unforg-cma}}_{\mathcal{DS},A_2}(k) + \mathbf{Adv}^{\text{unforg-cma}}_{\mathcal{DS},A_3}(k)\right) \ .$$