An extended abstract of this paper appears in *Advances in Cryptology – EUROCRYPT '03*, Lecture Notes in Computer Science Vol. 2656 , E. Biham ed., Springer-Verlag, 2003. This is the full version.

Foundations of Group Signatures: Formal Definitions, Simplified Requirements, and a Construction Based on General Assumptions

Mihir Bellare* Daniele Micciancio† Bogdan Warinschi‡

May 2003

Department of Computer Science and Engineering,
University of California, San Diego
9500 Gilman Drive, CA 92093
{mihir,daniele,bogdan}@cs.ucsd.edu
http://www-cse.ucsd.edu/users/{mihir, daniele,bogdan}

Abstract

This paper provides theoretical foundations for the group signature primitive. We introduce strong, formal definitions for the core requirements of anonymity and traceability. We then show that these imply the large set of sometimes ambiguous existing informal requirements in the literature, thereby unifying and simplifying the requirements for this primitive. Finally we prove the existence of a construct meeting our definitions based only on the assumption that trapdoor permutations exist.

Keywords: Foundations, theory, definitions, trapdoor permutations, group-signature schemes, anonymity.

*Supported in part by NSF grant CCR-0098123, NSF grant ANR-0129617 and an IBM Faculty Partnership Development Award.

†Supported in part by NSF CAREER Award CCR-0093029.

‡Supported in part by NSF CAREER Award CCR-0093029.

Contents

1 Introduction

A central line of work in theoretical cryptography is to provide formal (and strong) definitions of security for cryptographic primitives, and then provide constructions, based on general computational-complexity assumptions (such as the existence of one-way or trapdoor functions), that satisfy the definitions in question. The value and difficulty of such "foundational" work is acknowledged and manifest. A classical example is public-key encryption. Although it might seem like an intuitive goal, much work has been required to formally define and provably achieve it [20, 22, 19, 23, 26, 17], and these advances now serve as the basis for new schemes and applications. This paper provides such foundations for the group signatures primitive.

1.1 Background

In the group signature setting introduced by Chaum and Van Heyst [14] there is a group having numerous members and a single manager. Associated to the group is a single signature-verification key *gpk* called the group public key. Each group member i has its own secret signing key based on which it can produce a signature relative to *gpk*. The core requirements as per [14] are that the group manager has a secret key *gmsk* based on which it can, given a signature σ, extract the identity of the group member who created σ (traceability) and on the other hand an entity not holding *gmsk* should be unable, given a signature σ, to extract the identity of the group member who created σ (anonymity).

Since [14], more requirements, that refine or augment the core ones, have been introduced (eg. unlinkability, unforgeability, collusion resistance [5], exculpability [5], and framing resistance [15]) so that now we have a large set of unformalized, overlapping requirements whose precise meaning, and relation to each other, is neither always clear nor even always agreed upon in the existing literature.

The state of the art is represented by [5, 2] that identify weaknesses in previous works and present new schemes. The schemes in [2] are claimed to be proven-secure (in the random oracle model). However, while the work in question establishes certain properties, such as zero-knowledge, of certain subprotocols of its scheme, there is no actual definition of the security of a group signature scheme. For example, the anonymity requirement is formulated simply as the phrase "given a valid signature of some message, identifying the actual signer is computationally hard for everyone but the group manager." This is like formulating the privacy requirement for an encryption scheme as the phrase "the ciphertext should not reveal information about the message to anyone except the owner of the secret key," yet to truly capture privacy requires significantly more precise and non-trivial definitions, as evidenced by [20, 22, 19, 23, 26, 17]. In particular the informal requirement of [2] leaves open the issue of what exactly is the attack model and definition of adversarial success. Can the attacker see, or request, previous signatures? Can it call on the group manager to "open" some previous signatures? Can it have partial information ruling out some signers a priori? With such questions unanswered, we believe it is premature to say that proven-security has been achieved.

Furthermore, all claims of proven secure schemes so far have been in the random-oracle model. As far as we know, there are no constructions even claimed to be proven secure in the standard model.

1.2 Formal definitions

Providing appropriate definitions has required significantly more than merely formalizing the intuitive informal requirements of previous works. We consider novel attack capabilities and success

measures, and then formulate strong versions of the core requirements that we call full-anonymity and full-traceability. Perhaps surprisingly, we are then able to show that these two requirements are enough, in the sense that all the other requirements are implied by them. Our formalisms build on definitional ideas used for encryption [20, 22, 19, 23, 26, 17] and digital signatures [21].

FULL-ANONYMITY. We adopt an indistinguishability based formalization under which the adversary produces a message and a pair of group-member identities, is returned a target signature of the given message under a random one of the two identities and then is required to have negligible advantage over one-half in determining under which of the two identities the target signature was produced. Within this framework, we define a strong adversary that may corrupt all the members of the group, including the one issuing the signature. (Formally, the adversary is given the secret keys of all group members.) We also capture (in analogy to the definition of encryption secure against chosen-ciphertext attack [26]) the possibility that the adversary can see the outcome of opening attempts conducted by the group manager on arbitrary signatures of its choice (except of course the challenge signature).

FULL-TRACEABILITY. Our formulation of traceability is much stronger than what was called traceability in the past, and can be viewed also as a strong form of collusion-resistance. It asks that a group of colluding group members who pool their secret keys cannot create a valid signature that the opening algorithm would not catch as belonging to some member of the colluding group, and this is true even if the colluding group knows the secret key of the group manager under which signatures are opened.

1.3 Implications

As indicated above, there is a large and growing list of informal security requirements for group signatures. We show however that all existing requirements are implied by full-anonymity plus full-traceability. This provides a conceptual simplification with a clear advantage: having to check only two security properties makes it easier to give formal proofs of security when new group signature schemes are invented.

These implications might seem surprising at first glance, particularly because the implied properties include seemingly unrelated notions like unforgeability (nobody outside the group can produce valid signatures) or security against framing attacks (no one can produce signatures that will be later attributed to a honest group member that never signed the corresponding document). However the fact that a small number of strong formal notions imply a large number of informal requirements should be viewed, based on historical evidence, as an expected rather than a surprising benefit, and even as a test of the definitions in question. As an analogy, in the absence of a strong formal notion, one might formulate the security of encryption via a list of requirements, for example security against key-recovery (it should be hard to recover the secret key from the public key), security against inversion (it should be hard to recover the plaintext from the ciphertext), security under repetition (it should be hard to tell whether the messages corresponding to two ciphertexts are the same) and so on, and we can imagine these requirements being discovered incrementally and being thought to be quite different from each other. However, strong notions like indistinguishability [20], semantic security [20] and non-malleability [17] imply not only all these, but much stronger properties, and in particular put the different informal requirements under a common umbrella. We believe we are doing the same for group signatures.

1.4 Our scheme

With such strong notions of security, a basic theoretical question emerges, namely whether or not a scheme satisfying them even exists, and, if so, what are the minimal computational-complexity assumptions under which this existence can be proven. We answer this by providing a construction of a group signature scheme that provably achieves full-anonymity and full-traceability (and thus, by the above, also meets all previous security requirements) assuming only the existence of trapdoor permutations. We stress that this result is not in the random oracle model. Additionally we note that our construction is "non-trivial" in the sense that the sizes of all keys depend only logarithmically (rather than polynomially) on the number of group members.

The construction uses as building blocks an IND-CCA secure asymmetric encryption scheme, known to exist given trapdoor permutations via [17]; simulation-sound adaptive non-interactive zero-knowledge (NIZK) proofs for NP, known to exist given trapdoor permutations via [18, 28]; and a digital signature scheme secure against chosen-message attack, known to exist given trapdoor permutations via [6]. As often the case with constructs based on general assumptions, our scheme is polynomial-time but not practical, and our result should be regarded as a plausibility one only.

The basic framework of our construction builds on ideas from previous works (eg. [2]). Roughly, the secret signing key of a group member includes a key-pair for a standard digital signature scheme that is certified by the group manager. The group member's signature is an encryption, under a public encryption key held by the group manager, of a standard signature of the message together with certificate and identity information, and accompanied by a non-interactive zero-knowledge proof that encryption contains what it should.

Previous works did not try to achieve security notions as strong as we target, nor to pin down what properties of the building blocks suffice to actually prove security. For example we found that the encryption scheme had to be secure against chosen-ciphertext attack and not just chosen-plaintext attack. Further subtleties are present regarding the NIZK proofs. We require them to be simulation-sound and also have a strong, adaptive zero-knowledge property [28]. On the other hand, we only require NIZK proofs for a single theorem rather than multiple theorems. We highlight this because at first glance it can sound impossible, but it is due to the strong ZK property, and the situation is not without precedent: single-theorem adaptive ZK proofs have sufficed also for applications in [28].

1.5 Related work

As indicated above, the notion of group signature was introduced by Chaum and Heyst in [14]. They also gave the first schemes. Since then, many other schemes were proposed, including [15, 11, 25, 13, 4]. These schemes improve on the performance of the original group signature scheme of [14], but leave open some important security issues, most notably security against coalitions of group members. The importance of achieving provable security against coalitions of group members is pointed out in [5], where a (partial) coalition attack on the scheme of [13] is also described. A subsequent work trying to address the issue of securing group signature schemes against coalition attacks is [2]. On a separate research line, [10, 3, 12] investigate issues related to the dynamics of the group, to support membership revocation, and independent generation of group member keys. Still another extension is that of [29], that combines group signature schemes with forward security [1, 7].

The definitions and results of this paper are for the setting in which the group is static, meaning the number and identities of members is decided at the time the group is set up and new members cannot be added later. We consider it important to begin with this setting because, even though

dynamic groups have been considered (as indicated above), proper definitions of security have not been provided even for the basic static-group case, and the important definitional issues arise already in this context.

Our work provides a basis from which to treat the case of dynamic groups, but the latter do give rise to new issues. Section 5 discusses dynamic groups and other extensions. Providing formal definitions of security and provably-secure constructions for dynamic group signatures is the subject of on-going work [8].

2 Definitions of the security of group signature schemes

NOTATION AND TERMINOLOGY. 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 A is a randomized algorithm then $z \xleftarrow{\$} A(x, y, \ldots)$ denotes the operation of running A on inputs $x, y, \ldots$ and letting z be the output. If A is a randomized algorithm then $[A(x, y, \ldots)]$ denotes the set of all points having positive probability of being output by A on inputs $x, y, \ldots$.

We say that a function $f\colon \mathbb{N} \to \mathbb{N}$ is *nice* if it is polynomially bounded (i.e. there exists a polynomial $p(\cdot)$ such that $f(k) \leq p(k)$ for all $k \in \mathbb{N}$) and computable in poly(k) time. The notion of a function $\nu\colon \mathbb{N} \to \mathbb{N}$ being negligible is standard. In this paper we will need a notion of negligibility of a two-argument function $\mu\colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}$. We say such a μ is negligible if for every nice function $n\colon \mathbb{N} \to \mathbb{N}$, the function $\mu_n\colon \mathbb{N} \to \mathbb{N}$ is negligible, where $\mu_n(k) = \mu(k, n(k))$ for all $k \in \mathbb{N}$.

SYNTAX OF GROUP SIGNATURE SCHEMES. A *group signature scheme* $\mathcal{GS} = (\mathsf{GKg}, \mathsf{GSig}, \mathsf{GVf}, \mathsf{Open})$ consists of four polynomial-time algorithms:

- The randomized *group key generation* algorithm GKg takes input $1^k, 1^n$, where $k \in \mathbb{N}$ is the security parameter and $n \in \mathbb{N}$ is the group size (ie. the number of members of the group), and returns a tuple $(gpk, gmsk, \boldsymbol{gsk})$, where gpk is the *group public key*, $gmsk$ is the *group manager's secret key*, and $\boldsymbol{gsk}$ is an n-vector of keys with $\boldsymbol{gsk}[i]$ being a *secret signing key* for player $i \in [n]$.
- The randomized *group signing* algorithm GSig takes as input a secret signing key $\boldsymbol{gsk}[i]$ and a message m to return a signature of m under $\boldsymbol{gsk}[i]$ $(i \in [n])$.
- The deterministic *group signature verification* algorithm GVf takes as input the group public key gpk, a message m, and a candidate signature σ for m to return either 1 or 0.
- The deterministic *opening* algorithm Open takes as input the group manager secret key $gmsk$, a message m, and a signature σ of m to return an identity i or the symbol $\perp$ to indicate failure.

For simplicity we are assigning the members consecutive integer identities $1, 2, \ldots, n$. We say that σ is a *true* signature of m if there exists $i \in [n]$ such that $\sigma \in [\mathsf{GSig}(\boldsymbol{gsk}[i], m)]$. We say that σ is a *valid* signature of m with respect to gpk if $\mathsf{GVf}(gpk, m, \sigma) = 1$.

CORRECTNESS. The scheme must satisfy the following *correctness* requirement: For all $k, n \in \mathbb{N}$, all $(gpk, gmsk, \boldsymbol{gsk}) \in [\mathsf{GKg}(1^k, 1^n)]$, all $i \in [n]$ and all $m \in \{0,1\}^*$

$$\mathsf{GVf}(gpk, m, \mathsf{GSig}(\boldsymbol{gsk}[i], m)) = 1 \quad \text{and} \quad \mathsf{Open}(gmsk, m, \mathsf{GSig}(\boldsymbol{gsk}[i], m)) = i\ .$$

The first says that true signatures are always valid. The second asks that the opening algorithm correctly recovers the identity of the signer from a true signature.

DISCUSSION. The definitions above are for the setting in which the group is static, meaning the number and identities of members is decided at the time the group is set up and new members cannot be added later. For information about the dynamic group case refer to Section 5 and [8].

Experiment $\mathbf{Exp}^{\text{anon-b}}_{\mathcal{GS},A}(k,n)$

$(gpk, gmsk, \boldsymbol{gsk}) \xleftarrow{\$} \mathsf{GKg}(1^k, 1^n)$

$(\text{St}, i_0, i_1, m) \xleftarrow{\$} A^{\mathsf{Open}(gmsk,\cdot,\cdot)}(\mathsf{choose}, gpk, \boldsymbol{gsk})$; $\sigma \xleftarrow{\$} \mathsf{GSig}(\boldsymbol{gsk}[i_b], m)$

$d \xleftarrow{\$} A^{\mathsf{Open}(gmsk,\cdot,\cdot)}(\mathsf{guess}, \text{St}, \sigma)$

If A did not query its oracle with m, σ in the guess stage then return d EndIf

Return 0

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

$(gpk, gmsk, \boldsymbol{gsk}) \xleftarrow{\$} \mathsf{GKg}(1^k, 1^n)$

$\text{St} \leftarrow (gmsk, gpk)$; $\mathcal{C} \leftarrow \emptyset$; $K \leftarrow \varepsilon$; $\text{Cont} \leftarrow \mathsf{true}$

While $(\text{Cont} = \mathsf{true})$ do

$(\text{Cont}, \text{St}, j) \xleftarrow{\$} A^{\mathsf{GSig}(\boldsymbol{gsk}[\cdot],\cdot)}(\mathsf{choose}, \text{St}, K)$

If $\text{Cont} = \mathsf{true}$ then $\mathcal{C} \leftarrow \mathcal{C} \cup \{j\}$; $K \leftarrow \boldsymbol{gsk}[j]$ EndIf

Endwhile

$(m, \sigma) \xleftarrow{\$} A^{\mathsf{GSig}(\boldsymbol{gsk}[\cdot],\cdot)}(\mathsf{guess}, St)$

If $\mathsf{GVf}(gpk, m, \sigma) = 0$ then return 0 ; If $\mathsf{Open}(gmsk, m, \sigma) = \perp$ return 1

If there exists $i \in [n]$ such that the following are true then return 1 else return 0:

1. $\mathsf{Open}(gmsk, m, \sigma) = i$
2. $i \notin \mathcal{C}$
3. i, m was not queried by A to its oracle

Figure 1: Experiments used, respectively, to define full-anonymity and full-traceability of group signature scheme $\mathcal{GS} = (\mathsf{GKg}, \mathsf{GSig}, \mathsf{GVf}, \mathsf{Open})$. Here A is an adversary, $b \in \{0,1\}$, and St denotes state information passed by the adversary between stages.

Compactness. In practice it is preferable that sizes of keys and signatures in a group signature scheme do not grow proportionally to the number of members n. Actually, a polynomial dependency of these sizes on $\log(n)$ can be shown to be unavoidable, but ideally one would not want more than that. In our context, this leads to the following efficiency criterion for a group signature scheme. We call a group signature scheme $\mathcal{GS} = (\mathsf{GKg}, \mathsf{GSig}, \mathsf{GVf}, \mathsf{Open})$ *compact* if there exist polynomials $p_1(\cdot,\cdot)$ and $p_2(\cdot,\cdot,\cdot)$ such that

$$|gpk|, |gmsk|, |\boldsymbol{gsk}[i]| \leq p_1(k, \log(n)) \text{ and } |\sigma| \leq p_2(k, \log(n), |m|)$$

for all $k, n \in \mathbb{N}$, all $(gpk, gmsk, \boldsymbol{gsk}) \in [\mathsf{GKg}(1^k, 1^n)]$, all $i \in [n]$, all $m \in \{0,1\}^*$ and all $\sigma \in [\mathsf{GSig}(\boldsymbol{gsk}[i], m)]$.

Full-anonymity. Informally, anonymity requires that an adversary not in possession of the group manager's secret key find it hard to recover the identity of the signer from its signature. As discussed in the introduction, our formalization is underlain by an indistinguishability requirement, on which is superimposed an adversary with strong attack capabilities. To capture the possibility of an adversary colluding with group members we give it the secret keys of all group members. To capture the possibility of its seeing the results of previous openings by the group manager, we give it access to an *opening oracle*, $\mathsf{Open}(gmsk, \cdot, \cdot)$, which when queried with a message m and signature σ, answers with $\mathsf{Open}(gmsk, m, \sigma)$.

Let us now proceed to the formalization. To any group signature scheme $\mathcal{GS} = (\mathsf{GKg}, \mathsf{GSig}, \mathsf{GVf}, \mathsf{Open})$, adversary A and bit b we associate the first experiment given in Figure 1. Here, A is an adversary that functions in two stages, a choose stage and a guess stage. In the choose stage A takes as input the group members secret keys, $\boldsymbol{gsk}$, together with the group public key gpk. During this stage, it can also query the opening oracle $\mathsf{Open}(gmsk, \cdot)$ on group signatures of his choice, and it is required that at the end of the stage A outputs two valid identities $1 \leq i_0, i_1 \leq n$, and a message m. The adversary also outputs some state information to be used in the second stage of the attack. In the second stage, the adversary is given the state information, and a signature on m produced using the secret key of one of the two users i_0, i_1, chosen at random. The goal is to guess which of the two secret keys was used. The adversary can still query the opening oracle, but not on the challenge signature. We denote by

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

the advantage of adversary A in breaking the full-anonymity of $\mathcal{GS}$. We say that a group signature scheme is *fully-anonymous* if for any polynomial-time adversary A, the two-argument function $\mathbf{Adv}^{\text{anon}}_{\mathcal{GS},A}(\cdot,\cdot)$ is negligible in the sense of negligibility of two-argument functions defined at the beginning of this section.

In the above experiment, the adversary issues a request to sign its message m under one of two identities i_0, i_1 that it specifies, and wins if it can guess which was chosen. One might feel that allowing just one such request is restrictive, and the adversary should be allowed a sequence of such requests. (The challenge bit b remains the same in answering all requests). However, a standard hybrid argument shows that, under polynomial-time reductions, allowing a polynomial number of requests is equivalent to allowing just one request, so our definition is not restrictive. This hybrid argument depends on the fact that the adversary has the group public key and the secret signing keys of all members.

Full-traceability. In case of misuse, signer anonymity can be revoked by the group manager. In order for this to be an effective deterrence mechanism, we require that no colluding set S of group members (even consisting of the entire group, and even being in possession of the secret key for opening signatures) can create signatures that cannot be opened, or signatures that cannot be traced back to some member of the coalition. We remark that giving the opening key to the adversary does not model corruption of the group manager,[1] but rather compromise of the group manager's key by the adversary. We call our requirement *full-traceability*.

We formally define full-traceability using the second experiment of Figure 1. Here, adversary A runs in two stages, a choose stage and a guess stage. On input the group public key gpk and the secret of the group manager, $gmsk$, the adversary starts its attack by adaptively corrupting a set $\mathcal{C}$ of group members. The identity of the group members that are corrupted and their number is entirely up to it. At the end of the choose stage the set $\mathcal{C}$ contains the identities of the corrupted members. In the guess stage, the adversary attempts to produce a forgery (m, σ), and we say it wins (meaning the experiment returns 1), if σ is a valid group signature on m, but the opening algorithm returns $\perp$ or some valid user identity i such that $i \notin \mathcal{C}$. Otherwise, the experiment returns 0. We define the advantage of adversary A in defeating full-traceability of the group signature scheme $\mathcal{GS}$ by:

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

and say that $\mathcal{GS}$ is *fully-traceable* if for any polynomial-time adversary A, the two-argument function $\mathbf{Adv}^{\text{trace}}_{\mathcal{GS},A}(\cdot,\cdot)$ is negligible, as per the definition of negligibility of two-argument functions given at

[1] See section 5 for a detailed discussion of this case

the beginning of this section.

3 Relations to Existing Security Notions

We show that our formulations of anonymity and traceability are strong enough to capture all existing informal security requirements in the literature. This highlights the benefits of strong notions.

UNFORGEABILITY. A basic requirement of any digital signature scheme is that signatures cannot be forged, i.e., it is computationally unfeasible to produce message signature pairs (m, σ) that are accepted by the verification algorithm, without knowledge of the secret key(s). We did not include unforgeability among the main security properties of group signature schemes, since it immediately follows from full-traceability.

In order to define unforgeability, one can restrict the adversary used in the definition of full-traceability to just the second stage: in the first stage it does not ask for any secret key, and also, the secret key of the group manager is not part of its initial state St. Still, the adversary can obtain digital signatures of its choice using the signing oracle. In this case a successful attack against the full-traceability requirement reduces to producing a valid message signature pair (m, σ) such that message m was not queried to the signing oracle. The condition about the opening algorithm tracing the signature to a nonmember of the set of corrupted users is vacuously satisfied because this set is empty.

EXCULPABILITY. Exculpability (first introduced in [5]) is the property that no member of the group and not even the group manager can produce signatures on behalf of other users.

Clearly, a group signature scheme secure against full-traceability has also the exculpability property. If either the group manager or a user defeats the exculpability of a group signature scheme, then an adversary against full-traceability can be easily constructed. In the first case, the adversary simply follows the strategy of the group manager and produces a signature that is a forgery in the sense of full-traceability: it can not be traced to the one who produced it. In the second case, say user i can defeat the exculpability property. Then, an adversary as in the full-traceability experiment, which in the first stage makes a single request for $\boldsymbol{gsk}[i]$, and then follows the strategy of the user to produce the forgery, is successful against full-traceability.

TRACEABILITY. Originally [14], the name "traceability" was used to denote the *functional* property that if a message is signed with key $\boldsymbol{gsk}[i]$ and the opening algorithm is applied to the resulting signature, the output of the opening algorithm must be i. Later, the term has been overloaded to include an actual *security* requirement, namely that it is not possible to produce signatures which can not be traced to one of the group that has produced the signature. The two requirements seem to be first separated in [2] where the latter was identified with coalition resistance (see below). Nevertheless, a group signature scheme that is fully traceable, is also traceable (under either definition of traceability).

COALITION RESISTANCE. The possibility of a group of signers colluding together to generate signatures that cannot be traced to any of them was not part of the original formulation of secure group signature schemes. The requirement of traceability even in the face of attacks by a coalition of group members was explicitly considered for the first time only recently in [2], and termed coalition resistance. In the descriptions of the property, details such as whether the coalition is dynamically chosen or not are left unspecified. A strong formalization of coalition resistance can be obtained using the experiment for full-traceability in which the adversary is not given the secret key of the group manager. The rest remains essentially unchanged. It is then immediate that fully-traceable

group signature schemes are also coalition resistant.

FRAMING. Framing is a version of coalition resistance that was first considered in [15]. Here, a set of group members combine their keys to produce a valid signatures in such a way that the opening algorithm will attribute the signature to a different member of the group. As in the case of coalition resistance, framing has not been formally defined, issues similar to the one discussed above being left unspecified. A strong formalization for framing is the following: consider an experiment in which a user's identity u is chosen at random from the set of all users, and all group secret keys, except the secret key of u, together with the secret key of the group manager are given to the adversary. The adversary wins if it manages to produce a signature which will open as user u, and we call a scheme secure against framing if no efficient adversary can win with non-negligible probability.

A group signature scheme that is fully-traceable, is also secure against framing. Indeed, an adversary B against framing can be turned into an adversary A against full-traceability as follows. It generates a random user identity and requests the secret keys of all other users. Then it runs adversary B with input these secret keys and the secret key of the group manager and outputs the forgery attempted by B. If B is successful in its framing attempt, then A defeats full-traceability.

ANONYMITY. As we have already discussed, anonymity is just a weaker form of full anonymity, in which the adversary does not have access to the opening oracle, and also, has no information of the member's secret keys. Clearly, a scheme that is fully-anonymous is also anonymous.

UNLINKABILITY. This is related to anonymity rather than full-traceability. The intuition is that a party after seeing a list of signatures, can not relate two signatures together as being produced by the same user. As for anonymity, it was implicit in previous definitions that the attacker was assumed not to be a group member. Again, realistic scenarios exist where this is not necessarily the case. It is thus necessary to consider distinct security notions for the two cases i.e. *insider/outsider unlinkability*. In either case, formalizing unlinkability is not an easy task because it is not clear how (possibly linked) signatures should be produced. For example we can require that it is computationally hard to distinguish between a box that each time receives an input message produces a signature using a fixed secret key $\boldsymbol{gsk}[i]$, or a box where i is chosen uniformly at random for every invocation of the oracle. However, it is not clear why the uniform distribution should be used in the second case. Alternatively, we could let the adversary choose the distribution in the second case, but still in real applications it seems unjustifiable to assume that the signer is chosen each time independently at random and the choice of the signer at different times might be correlated. We considered various possible formalization of the notion of unlinkability, and in all cases we could prove that it follows from anonymity. Also, it seems that for any reasonable formulation of unlinkability, one can prove that anonymity also follows. We conclude that anonymity and unlinkability are technically the same property, and only the easier to define anonymity property needs to be included in the security definition.

4 Our construction

We begin by describing the primitives we use, and then describe our construction.

4.1 Primitives

DIGITAL SIGNATURE SCHEMES. A digital signature scheme $\mathcal{DS} = (\mathsf{K_s}, \mathsf{Sig}, \mathsf{Vf})$ is specified, as usual, by algorithms for key generation, signing and verifying. Our construction uses a digital signature

scheme that satisfies the standard notion of unforgeability under chosen message attack [21]. We now recall the definition.

Consider the following experiment $\mathbf{Exp}^{\text{unforg-cma}}_{\mathcal{DS},A}(k)$, 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 $(pk, sk) \xleftarrow{\$} \mathsf{K_s}(1^k)$. 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.

It is known that digital signature schemes, unforgeable under chosen message attack, exist assuming one-way functions exist [27], and hence certainly assuming the existence of a family of trapdoor permutations.

Encryption schemes. A public-key encryption scheme $\mathcal{AE} = (\mathsf{K_e}, \mathsf{Enc}, \mathsf{Dec})$ is specified, as usual, by algorithms for key generation, encryption and decryption. Our group signature construction utilizes an encryption scheme that satisfies the standard notion of indistinguishability under chosen-ciphertext attack (IND-CCA) [26]. For completeness, we now recall the definition.

An IND-CCA adversary against $\mathcal{AE}$ is an algorithm A that operates in two stages, a choose stage and a guess stage. For a a fixed bit b, the adversary works as follows. In the first stage the algorithm is given a public key pk_e for encryption. It is also given access to a decryption oracle which decrypts submitted ciphertexts using the secret key sk_e that corresponds to pk_e. At the end of the first stage it outputs a pair of messages M_0 and M_1. The input of the algorithm to the second stage is some state information, also produced at the end of the first stage, and a challenge ciphertext C that is an encryption of M_b. At the end of the second stage (in which A still has access to the decryption oracle but is not allowed to submit C as a query) the adversary outputs a guess bit d that selects one or the other message. The adversary wins if he guesses successfully which of the messages was encrypted. Let $\mathbf{Exp}^{\text{ind-cca}\_b}_{\mathcal{AE},A}(k)$ denote the random variable representing the outcome of A in the above experiment, when pk_e is obtained by running the key generation algorithm (with fresh coins) on security parameter k. 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.

It is known that the existence of trapdoor permutations implies that such encryption schemes exists [17].

Simulation-sound Non-interactive zero knowledge proof systems. The last building block we need are simulation-sound NIZK proofs of membership in NP languages. The following presentation is along the lines of [18, 28].

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 a 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) form a *non-interactive proof system* for ρ [18, 9] over domain Dom if there exists a polynomial p such that the following two conditions are satisfied:

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

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

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

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

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 [18, 9] but has been highlighted for example in [16]. The construction of [18] does have this property, and it is noted and used in other places too.

Zero-knowledge is defined by means of a *distinguisher* D which essentially tries to distinguish between proofs 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 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 $x \in$ 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.) Having produced $(x, w) \in \rho$ with $x \in$ Dom, the simulator 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. 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 such that 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 [18], 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 [28]. 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 $\mathbf{Exp}^{\text{ss}}_{\Pi,A}(k)$ that we describe now. The experiment involves a simulation-soundness adversary A working in two stages, choose and forge.

First, a "fake common" random string R, together with the associated trap-door information St_S is generated by running the simulator: $(R, \mathrm{St}_S) \xleftarrow{\$} \textsc{Sim}(\mathsf{gen}, k)$. The string is passed to the first stage of the simulation-soundness adversary, which outputs an element $y \in \mathrm{Dom}$ together with some information St_A. The experiment uses the simulator to produce π, a proof of validity of y. Then (y, π, St_A) are passed to the forge stage of A, at the end of which the adversary is required to output a pair (x, π'). The experiment returns 1 (i.e. the adversary wins) if the conditions (1) $\pi' \neq \pi$, (2) $x \notin$ and (3) $L_\rho, V(k, x, \pi', R) = 1$ are satisfied. Otherwise the experiment returns 0. 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.

From [18, 28], if trapdoor permutations exist, then any NP-relation has a simulation-sound, non-interactive zero knowledge proof system.

4.2 Our construction

Overview. We now show that the building blocks above can be used to construct a group signature scheme $\mathcal{GS} = (\mathsf{GKg}, \mathsf{GSig}, \mathsf{GVf}, \mathsf{Open})$ that is both fully-anonymous and fully-traceable. 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 begin with an overview.

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 a reference string R. We denote by sk_s the signing key corresponding to pk_s, and call it the *certificate creation* key. The secret key $\boldsymbol{gsk}[i]$ of group member i contains a signature key sk_i with matching verification key pk_i and a certificate (i.e. signature) cert_i of $\langle i, pk_i\rangle$ under the certificate creation key. (For convenience the public key of the group gpk is also thrown into $\boldsymbol{gsk}[i]$.) The manager secret key $gmsk$ is the decryption key sk_e corresponding to pk_e. The certificate creation key sk_s is however denied to the group manager. (This prevents the latter from issuing certificates for keys it generates itself, and is important to attain full-traceability). As far as our abstract key generation mechanisms goes, we imagine that this key, although created and used to produce certificates, is simply not given to the group manager. In "real life," we would expect that a key-issuing entity, distinct from the group manager or players, maintains the certification keys. It can add members to the group dynamically via a protocol in which an aspiring group member i creates its own signature and verification keys sk_i, pk_i via $\mathsf{K_s}(k)$, and sends pk_i to the key issuer, who signs $\langle i, pk_i\rangle$ under pk_s to obtain cert_i and returns this to i along with k, R, pk_e, pk_s. The key issuer does not give the group manager the secret certificate creation key sk_s.

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 the certificate verification key pk_s present in the group public key. All proofs are with respect to reference string R of the group public key.

Group signature verification comes down to verification of the NIZK proofs. Opening is possible because the group manager has the decryption key sk_e.

Specification. We need to begin by specifying the witness relation ρ underlying the zero-knowledge proofs. We will then fix a proof system (P, V) for ρ and define the four algorithms

Algorithm $\mathsf{GKg}(k, n)$
$R \xleftarrow{\$} \{0,1\}^{p(k)}$; $(pk_e, sk_e) \xleftarrow{\$} \mathsf{K_e}(1^k)$; $(pk_s, sk_s) \xleftarrow{\$} \mathsf{K_s}(1^k)$
$gpk \leftarrow (k, R, pk_e, pk_s)$; $gmsk \leftarrow (k, pk_e, sk_e, pk_s)$;
For $i \leftarrow 1$ to n do
$(pk_i, sk_i) \xleftarrow{\$} \mathsf{K_s}(1^k)$; $\mathrm{cert}_i \leftarrow \mathsf{Sig}(sk_s, \langle i, pk_i \rangle)$
$\boldsymbol{gsk}[i] \leftarrow (i, pk_i, sk_i, \mathrm{cert}_i, gpk)$
Endfor
Return $(gpk, gmsk, \boldsymbol{gsk})$

Algorithm $\mathsf{GVf}(gpk, (m, \sigma))$
Parse gpk as (k, R, pk_e, pk_s) ; Parse σ as (C, π)
Return $V(k, (pk_e, pk_s, M, C), \pi, R)$

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

Algorithm $\mathsf{Open}(gmsk, gpk, m, \sigma)$
Parse $gmsk$ as (k, pk_e, sk_e, pk_s) ; Parse σ as (C, π)
If $V(k, (m, C), \pi, R) = 0$ then return $\perp$
Parse $\mathsf{Dec}(sk_e, C)$ as $\langle i, pk, \mathrm{cert}, s \rangle$
Return i

Figure 2: **Our construction:** Group signature scheme $\mathcal{GS} = (\mathsf{GKg}, \mathsf{GSig}, \mathsf{GVf}, \mathsf{Open})$ associated to digital signature scheme $\mathcal{DS} = (\mathsf{K_s}, \mathsf{Sig}, \mathsf{Vf})$, public-key encryption scheme $\mathcal{AE} = (\mathsf{K_e}, \mathsf{Enc}, \mathsf{Dec})$, and non-interactive proof system (P, V).

---

constituting the group signature scheme in terms of P, V and the algorithms of $\mathcal{DS}$ and $\mathcal{AE}$. Consider the relation ρ defined as follows: $((pk_e, pk_s, M, C), (i, pk', \mathrm{cert}, s, r)) \in \rho$ iff

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

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 a message m under the key pk_e using the coins r, and assume that $|r| = k$. The domain Dom corresponding to ρ 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 is a NP relation over Dom, and consequently we can fix a non-interactive zero knowledge proof system (P, V) for it. Based on this, the algorithms defining the group signature scheme are depicted in Figure 2.

4.3 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-relation ρ over domain Dom, and its non-interactive proof system (P, V) as above, and let $\mathcal{GS} =$

(GKg, GSig, GVf, Open) denote the signature scheme associated to them as per our construction. We derive our main result (Theorem 4.3) via the following two lemmas.

Lemma 4.1 *If $\mathcal{AE}$ is an IND-CCA secure encryption scheme and (P, V) is a simulation sound, computational zero-knowledge proof system for ρ over* Dom *then group signature scheme $\mathcal{GS}$ is fully-anonymous.* ▮

Lemma 4.2 *If digital signature scheme $\mathcal{DS}$ is secure against forgery under chosen-message attack and (P, V) is a sound non-interactive proof system for ρ over* Dom *then group signature scheme $\mathcal{GS}$ is fully-traceable.* ▮

The scheme we have described is compact. Indeed, keys, as well as sizes of signatures of k-bit messages, are of size $\text{poly}(k, \log(n))$. Since it is known that the existence of a family of trapdoor permutations implies the existence of the primitives we require [17, 6, 18, 28], we get:

Theorem 4.3 *If there exists a family of trapdoor permutations, then there exists a compact group signature scheme that is fully-anonymous and fully-traceable.* ▮

We start with the proof of Lemma 4.1.

Proof: By the assumption that P is computational zero-knowledge for ρ over Dom, we can fix a simulator Sim such that (P, V, Sim) is a simulation sound zero knowledge non-interactive proof system for L_ρ.

We show that for any given nice function $n(\cdot)$, and any polynomial time adversary B mounting an attack against full-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 and a distinguisher D that distinguishes simulated proofs from real proofs, such that for all $k \in \mathbb{N}$

$$\mathbf{Adv}^{\text{anon}}_{\mathcal{GS},B}(k, n(k)) \leq \mathbf{Adv}^{\text{ind-cca}}_{\mathcal{AE},A_0}(k) + \mathbf{Adv}^{\text{ind-cca}}_{\mathcal{AE},A_1}(k) + \mathbf{Adv}^{\text{ss}}_{\Pi,A_{ss}}(k) + 2 \cdot \mathbf{Adv}^{\text{zk}}_{\text{P},\text{Sim},D}(k) \qquad (1)$$

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. It follows from our definition of negligibility of 2-argument functions, that $\mathbf{Adv}^{\text{anon}}_{\mathcal{GS},B}(\cdot, \cdot)$ is negligible too, i.e. our construction is a fully anonymous group signature scheme.

Adversaries against the encryption scheme. The details of adversaries A_0, A_1 against the encryption scheme $\mathcal{AE}$ is given in Figure 3. They are virtually identical, modulo parameter b by which their construction is parametrized, and so is the following description.

In the choose stage, A_b creates a group signature scheme. It generates all individual secret keys as well as the certification/verification keys. The difference from a real group signature scheme is that the group manager public key (used for encryption) is obtained from the environment in which A is run (the CCA experiment) and that the random string R in the public key of the encryption scheme is obtained by using the simulator Sim.

Then, A_b 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_b intercepts this query and checks to see if the signature is valid (this is easy, since A_b possesses gpk.) Then, A_b submits the encrypted part of the signature to the decryption oracle, and from the plaintext, A extracts the identity of the alleged signer which it passes to B.

Algorithm $A_b^{\mathsf{Dec}(sk_e,\cdot)}(\mathsf{choose}, pk_e)$
$(pk_s, sk_s) \xleftarrow{\$} \mathsf{K_s}(1^k)$
$(st_S, R) \xleftarrow{\$} \textsc{Sim}(\mathsf{gen}, k)$
$gpk \leftarrow (k, R, pk_e, pk_s)$
For $i \leftarrow 1$ to $n(k)$ do
$(pk_i, sk_i) \xleftarrow{\$} \mathsf{K_s}(1^k)$
$\mathrm{cert}_i \xleftarrow{\$} \mathsf{Sig}(sk_s, \langle i, pk_i \rangle)$
$\boldsymbol{gsk}[i] \leftarrow (i, pk_i, sk_i, \mathrm{cert}_i, gpk)$
EndFor
$(st, m, i_0, i_1) \xleftarrow{\$} B^{(\cdot)}(\mathsf{choose}, gpk, \boldsymbol{gsk})$
$s_b \xleftarrow{\$} \mathsf{Sig}(sk_{i_b}, m)$
$\mathrm{St} \leftarrow (m, st, st_S, pk_e, pk_s)$
$M_b \leftarrow \langle i_b, pk_{i_b}, \mathrm{cert}_{i_b}, s_b \rangle$
$M_{\bar{b}} \leftarrow 0^{|M_b|}$
Return (M_0, M_1, St)

Adversary $A^{\mathsf{Dec}(sk_e,\cdot)}(\mathsf{guess}, St, C)$
Parse St as $(m, st, st_S, pk_e, pk_s)$
$\pi \leftarrow \textsc{Sim}(\mathsf{prove}, st_S, (pk_e, pk_s, m, C))$
Run $B^{(\cdot)}(\mathsf{guess}, st, (C, \pi))$
If at any point B makes a valid query (C, π')
Then $d \leftarrow b$
Else d is the output of B
Return d

Algorithm $D(\mathsf{choose}, k, R)$
$(pk_e, sk_e) \leftarrow \mathsf{K_e}(1^k)$
$(pk_s, sk_s) \leftarrow \mathsf{K_s}(1^k)$
$gpk \leftarrow (k, R, pk_e, pk_s)$
For $i \leftarrow 1$ to $n(k)$ do
$(pk_i, sk_i) \leftarrow \mathsf{K_s}(1^k)$
$\mathrm{cert}_i \leftarrow \mathsf{Sig}(sk_i, \langle i, pk_i \rangle)$
$\boldsymbol{gsk}[i] \leftarrow (i, pk_i, sk_i, \mathrm{cert}_i, gpk)$
EndFor
$(st, m, i_0, i_1) \xleftarrow{\$} B^{(\cdot)}(\mathsf{choose}, gpk, \boldsymbol{gsk})$
$b \xleftarrow{\$} \{0,1\}$; $r \xleftarrow{\$} \{0,1\}^k$
$s \leftarrow \mathsf{Sig}(sk_{i_b}, m)$
$C \leftarrow \mathsf{Enc}(pk_e, \langle i_b, pk_{i_b}, \mathrm{cert}_{i_b}, s \rangle; r)$
$St \leftarrow (st, b, C)$
Return $(St, (pk_e, pk_s, m, C), (i_b, pk_{i_b}, \mathrm{cert}_{i_b}, s, r))$

Algorithm $D(\mathsf{guess}, St, \pi)$
Parse St as (st, b, C)
$d \leftarrow B^{(\cdot)}(\mathsf{guess}, st, (C, \pi))$
If $d = b$ Then Return 1
Else Return 0

Figure 3: Adversary A_b $(b = 0, 1)$ is against the security of the encryption scheme ; D is a distinguishser against the zero knowledge property of the interactive proof system

When B outputs a message m and two identities i_0, i_1, A_b creates two challange plaintexts M_0, M_1, (to be returned to the IND-CCA experiement at the end of the choose stage of A_b). The plaintexts are computed as follows: M_b is the plaintext of the encrypted part of a group signature on m produced by i_b and $M_{\bar{b}}$ is an all-zero string of length equal to that of M_b.

We now move to the guess stage of A_b in which the adversary receives as input a ciphertext C, which is the encryption of one of the two messages, M_0 and M_1. Next, A_b runs the simulator to obtain a proof π of validity for (pk_e, pk_s, m, C). This is always possible, event in the case when C encrypts $M_{\bar{b}}$. The challange signature that is passed to the second stage of B is (C, π) which A_b now simulates. The final output of A_b 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 b, otherwise it is set to whatever B outputs.

Notice that further opening queries of B can be answered by A_b using the decryption oracle (as 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_b, instead of submitting C to the decryption oracle, simply outputs b and terminates.

THE DISTINGUISHER FOR ZERO-KNOWLEDGE. The distinguisher D (given in Figure 3), 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 certyfing/veryfing individual public keys are obtained by running the resepctive key generation algorithms. The string R that is part of the public key is supplied to D by the environment in which it is run.

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. Notice that D can easily answer any query that B may make to the opening oracle, since it possesses $gmsk$. The challange group signature that D passes to the second stage of B 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. It then outputs $(pk_s, pk_e, m, C) \in L_\rho$ together with the corresponding witness. The input to the second stage of D contains π, which is a proof of membership of (pk_s, pk_e, m, C) to L_ρ (which depending on the environment is either simulated or real). In this stage, D creates a group signature (C, π) on m and feeds it to the second stage of B. The final output of D is whatever B outputs.

Algorithm $A_s^{\text{SIM}(\mathsf{prove}, st_S, \cdot)}(\mathsf{choose}, R)$
- $(pk_e, sk_e) \xleftarrow{\$} \mathsf{K_e}(1^k)$
- $(pk_s, sk_s) \xleftarrow{\$} \mathsf{K_s}(1^k)$
- $gpk \leftarrow (R, pk_e, pk_s)$
- For $i \leftarrow 1$ to $n(k)$ do
  - $(pk_i, sk_i) \xleftarrow{\$} \mathsf{K_s}(1^k)$
  - $\text{cert}_i \xleftarrow{\$} \mathsf{Sig}(sk_s, \langle i, pk_i \rangle)$
  - $\boldsymbol{gsk}[i] \leftarrow (i, pk_i, sk_i, \text{cert}_i, gpk)$
- EndFor
- $(st, m, i_0, i_1) \xleftarrow{\$} B^{(\cdot)}(\mathsf{choose}, gpk, \boldsymbol{gsk})$
- $s_b \xleftarrow{\$} \mathsf{Sig}(sk_{i_b}, m)$
- $St \leftarrow (m, st, st_S, pk_e, pk_s)$
- $M_1 \leftarrow \langle i_1, pk_{i_1}, \text{cert}_{i_1}, s_1 \rangle$
- $M_0 \leftarrow 0^{|M_1|}$
- $C \xleftarrow{\$} \mathsf{Enc}(pk_e, M_0)$
- $\pi \leftarrow \text{SIM}(\mathsf{prove}, st_S, (pk_e, pk_s, m, C))$ [oracle query]
- Run $B^{(\cdot)}(\mathsf{guess}, st, (C, \pi))$
  - If B makes a valid query (C, π')
  - Output $((pk_e, pk_s, m, C), \pi')$

Figure 4: Adversary A_s is against simulation-soundness

THE SIMULATION-SOUNDNESS ADVERSARY. The adversary A_s against the simulation soundness is given in Figure 4. 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 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). Then A_s runs B against the group signature scheme until B outputs the message and the two identities i_0 and i_1 for which it can distinguish signatures. The challenge signature (C, π) that A_s passes to the second stage of B is such that C is the encryption of the all-zero string (of

appropriate length) and π is a simulated proof of validity. Finally, A 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 $\mathbf{Exp}^{\text{zk\_1}}_{P,\text{SIM},D}(k)$, the string R passed to the distinguisher is an actual random string; so D runs B against a group signature scheme, generated according to the key generation algorithm GKg. If (m, i_0, i_1) is the output of the choose stage of B, the signature passed to the guess stage of B is the real signature of i_b, where b is chosen at random. So, D outputs 1, exactly when B guesses correctly which user produced the signature, i.e. it wins in the $\mathbf{Exp}^{\text{anon-b}}_{\mathcal{GS},B}(k)$, no matter what D's choice of b is. If $n(\cdot)$ is any fixed nice function, we can formalize the above as follows:

$$\begin{aligned}
&\Pr\left[\,\mathbf{Exp}^{\text{zk_1}}_{P,\text{SIM},D}(k) = 1\,\right] \\
&= \Pr\left[\,B(\mathsf{guess}) = 1 \mid b = 1\,\right]\cdot\Pr\left[\,b = 1\,\right] + \Pr\left[\,B(\mathsf{guess}) = 0 \mid b = 0\,\right]\cdot\Pr\left[\,b = 0\,\right] \\
&= \frac{1}{2}\Pr\left[\,\mathbf{Exp}^{\text{anon-ia_1}}_{\mathcal{GS},B}(k, n(k)) = 1\,\right] + \frac{1}{2}\Pr\left[\,\mathbf{Exp}^{\text{anon-ia_0}}_{\mathcal{GS},B}(k) = 0\,\right] \\
&= \frac{1}{2}\Pr\left[\,\mathbf{Exp}^{\text{anon-ia_1}}_{\mathcal{GS},B}(k, n(k)) = 1\,\right] + \frac{1}{2}\left(1 - \Pr\left[\,\mathbf{Exp}^{\text{anon-ia_0}}_{\mathcal{GS},B}(k, n(k)) = 1\,\right]\right) \\
&= \frac{1}{2} + \frac{1}{2}\mathbf{Adv}^{\text{anon}}_{\mathcal{GS},B}(k, n(k))
\end{aligned}$$

Eliminating the fractions, we have that:

$$2\cdot\Pr\left[\,\mathbf{Exp}^{\text{zk\_1}}_{P,\text{SIM},D}(k) = 1\,\right] = 1 + \mathbf{Adv}^{\text{anon}}_{\mathcal{GS},B}(k, n(k)) \tag{2}$$

We now analyze the success of D under experiment $\mathbf{Exp}^{\text{zk\_0}}_{P,\text{SIM},D}(k)$, and relate it to the advantages of adversaries A_0, A_1 and A_s. The key observation is that in all these adversaries, B is run as a subroutine against a group signature scheme in which keys are generated according the same distribution: the only difference from a real group signature scheme is that the random string that is part of the group public key, is obtained from the simulator. The second important observation is related to the input to the guess stage of B. Let (st, m, i_0, i_1) be the output of the first stage of B. The challenge signature (C, π) passed to the second stage of B is sampled from one of distributions $\mathcal{D}_0, \mathcal{D}_1$ and $\mathcal{D}_{0^k}$ that we now describe. For $b = 0, 1$, the elements of distribution $\mathcal{D}_b$ are signature-looking pairs, (C, π), where C is the encryption (under the public key of the group manager) of a plaintext that i_b would normally encrypt when creating a group signature on m, and π is a proof of validity obtained by running the simulator on (pk_e, pk_s, m, C). The distribution $\mathcal{D}_{0^k}$ is obtained similarly, but C is the encryption of the all-zero message. We will write $B(0), B(1)$ and $B(0^k)$ for the outcome of the second stage of B when the challenge signature comes from distribution $\mathcal{D}_0, \mathcal{D}_1$ and $\mathcal{D}_{0^k}$, respectively. Finally, we consider the following events concerning the behavior of B. We will denote by $\mathsf{ASK}_0, \mathsf{ASK}_1$ and ASK_{0^k} the event that when the challenge signature (C, π) is sampled from distribution $\mathcal{D}_0, \mathcal{D}_1$ and $\mathcal{D}_{0^k}$ respectively, then during the second guess stage B makes a valid query (C, π') to the opening oracle. By $\mathsf{NASK}_0, \mathsf{NASK}_1$ and NASK_{0^k} we denote the corresponding opposite events.

We can now analyze, using the above notation, the success of the above adversaries when run under the various experiments for which they are intended.

1. From the description of A_1, it can be seen that in the experiment $\mathbf{Exp}^{\text{ind-cca\_1}}_{\mathcal{AE},A_1}(k)$, adversary A_1 always passes to the second stage of B a challenge signature sampled from D_1. From the definition of A_1, it is immediate that there are two cases in which A_1 outputs 1 in this experiment: if event $\mathsf{ASK}(1)$ happens, or if B outputs 1 and $\mathsf{NASK}(1)$ happens. Formally:

$$\Pr\left[\,\mathbf{Exp}^{\text{ind-cca\_1}}_{\mathcal{AE},A_1}(k)=1\,\right]=\Pr\left[\,B(1)=1\wedge\mathsf{NASK}_1)\,\right]+\Pr\left[\,\mathsf{ASK}_1\,\right] \tag{3}$$

   The relation that we actually need (and immediately follows the above) is that:

$$\Pr\left[\,\mathbf{Exp}^{\text{ind-cca\_1}}_{\mathcal{AE},1}(k)=1\,\right]\geq\Pr\left[\,B(1)=1\,\right] \tag{4}$$

2. When A_1 is run in experiment $\mathbf{Exp}^{\text{ind-cca\_0}}_{\mathcal{AE},A_1}(k)$, the challenge signature passed to B is sampled from the distribution $\mathcal{D}_{0^k}$. A similar reasoning as above leads to the relation:

$$\Pr\left[\,\mathbf{Exp}^{\text{ind-cca\_0}}_{\mathcal{AE},A_1}(k)=1\,\right]=\Pr\left[\,B(0^k)=1\wedge\mathsf{NASK}_{0^k}\,\right]+\Pr\left[\,\mathsf{ASK}_{0^k}\,\right] \tag{5}$$

3. Now we focus on the experiment $\mathbf{Exp}^{\text{ind-cca\_1}}_{\mathcal{AE},A_0}(k)$. The challenge signature passed to B comes from distribution $\mathcal{D}_{0^k}$, so the experiment returns 1, exactly when $B(0^k)=1$ and event NASK_{0^k} occurs (otherwise the experiment returns 0):

$$\Pr\left[\,\mathbf{Exp}^{\text{ind-cca\_1}}_{\mathcal{AE},A_0}(k)=1\,\right]=\Pr\left[\,B(0^k)=1\wedge\mathsf{NASK}_{0^k}\,\right] \tag{6}$$

4. When A_0 is run under experiment $\mathbf{Exp}^{\text{ind-cca\_0}}_{\mathcal{AE},A_0}(k)$, the challenge signature is sampled from $\mathcal{D}_0$, so the experiment returns 1 only when both $B(0)=1$ and event NASK_0 happen (otherwise the experiment returns 0):

$$\Pr\left[\,\mathbf{Exp}^{\text{ind-cca\_0}}_{\mathcal{AE},A_0}(k)=1\,\right]=\Pr\left[\,B(0)=1\wedge\mathsf{NASK}_0\,\right]\leq\Pr\left[\,B(0)=1\,\right] \tag{7}$$

5. Notice that the simulation-soundness adversary A_s passes to the second stage of B a sample from distribution $B(0^k)$. Since, $(pk_e,pk_s,m,C)\notin L_\rho$ (here C is the encryption of the all-zero message), A_s wins in the simulation soundness experiment exactly when B makes a valid query (C,π') to the decryption oracle in which $\pi'\neq\pi$; so:

$$\mathbf{Adv}^{\text{ss}}_{\Pi,A_s}(k)=\Pr\left[\,\mathsf{ASK}_{0^k}\,\right] \tag{8}$$

   Combining this last relation with equations (5) and (6) we obtain that

$$\Pr\left[\,\mathbf{Exp}^{\text{ind-cca\_0}}_{\mathcal{AE},A_1}(k)=1\,\right]-\Pr\left[\,\mathbf{Exp}^{\text{ind-cca\_1}}_{\mathcal{AE},A_0}(k)=1\,\right]=\mathbf{Adv}^{\text{ss}}_{\Pi,A_s}(k) \tag{9}$$

6. The success probability of D under experiment $\mathbf{Exp}^{\text{zk\_0}}_{P,\text{SIM},D}(k)$ is determined as follows. From the description of D, the signature passed to the second stage of B is sampled from $\mathcal{D}_0$, or $\mathcal{D}_1$, the choice of distribution being determined by the random bit b. So, D outputs 1 exactly when B correctly guesses which is the distribution from which the challenge signature was chosen. Formally:

$$\begin{aligned}
&\Pr\left[\,\mathbf{Exp}^{\text{zk_0}}_{P,\text{SIM},D}(k)=1\,\right]\\
&\quad=\quad\Pr\left[\,B(0)=0\,\right]\cdot\Pr\left[\,b=0\,\right]+\Pr\left[\,B(1)=1\,\right]\cdot\Pr\left[\,b=1\,\right]\\
&\quad=\quad\frac{1}{2}\left(1-\Pr\left[\,B(0)=1\,\right]+\Pr\left[\,B(1)=1\,\right]\right)
\end{aligned} \tag{10}$$

Using equations (4) and (7) it immediately follows that:

$$2 \cdot \Pr\left[\mathbf{Exp}^{\text{zk\_0}}_{P,\text{SIM},D}(k) = 1\right] \leq 1 + \mathbf{Exp}^{\text{ind-cca\_1}}_{\mathcal{AE},A_1}(k) - \mathbf{Exp}^{\text{ind-cca\_0}}_{\mathcal{AE},A_0}(k) \tag{11}$$

Now, in the right hand side of the above inequality, we add and substract the right and the left sides of Equation (9) and obtain:

$$\begin{aligned}
& 2 \cdot \Pr\left[\mathbf{Exp}^{\text{zk_0}}_{P,\text{SIM},D}(k) = 1\right] \\
& \leq \quad 1 \\
& + \quad \Pr\left[\mathbf{Exp}^{\text{ind-cca_1}}_{\mathcal{AE},A_1}(k) = 1\right] - \Pr\left[\mathbf{Exp}^{\text{ind-cca_0}}_{\mathcal{AE},A_1}(k) = 1\right] \\
& + \quad \Pr\left[\mathbf{Exp}^{\text{ind-cca_1}}_{\mathcal{AE},A_0}(k) = 1\right] - \Pr\left[\mathbf{Exp}^{\text{ind-cca_0}}_{\mathcal{AE},A_0}(k) = 1\right] \\
& + \quad \mathbf{Adv}^{\text{ss}}_{\Pi,A_s}(k) \\
& = \quad \mathbf{Adv}^{\text{ind-cca}}_{\mathcal{AE},A_1}(k) + \mathbf{Adv}^{\text{ind-cca}}_{\mathcal{AE},A_0}(k) + \mathbf{Adv}^{\text{ss}}_{\Pi,A_s}(k)
\end{aligned}$$

We can now calculate the advantage of distinguisher D, by combining the above equation with Equation 2:

$$\begin{aligned}
2 \cdot \mathbf{Adv}^{\text{zk}}_{P,\text{SIM},D}(k) &= 2 \cdot \left(\Pr\left[\mathbf{Exp}^{\text{zk_1}}_{P,\text{SIM},D}(k) = 1\right] - \Pr\left[\mathbf{Exp}^{\text{zk_0}}_{P,\text{SIM},D}(k) = 1\right]\right) \\
&\geq \mathbf{Adv}^{\text{anon}}_{\mathcal{GS},B}(k, n(k)) - \mathbf{Adv}^{\text{ind-cca}}_{\mathcal{AE},A_0}(k) - \mathbf{Adv}^{\text{ind-cca}}_{\mathcal{AE},A_1}(k) - \mathbf{Adv}^{\text{ss}}_{\Pi,A_s}(k)
\end{aligned}$$

and by rearranging the terms we obtain

$$\mathbf{Adv}^{\text{anon}}_{\mathcal{GS},B}(k, n(k)) \leq \mathbf{Adv}^{\text{ind-cca}}_{\mathcal{AE},A_0}(k) + \mathbf{Adv}^{\text{ind-cca}}_{\mathcal{AE},A_1}(k) + \mathbf{Adv}^{\text{ss}}_{\Pi,A_s}(k) + 2 \cdot \mathbf{Adv}^{\text{zk}}_{\text{P,SIM},D}(k)$$

as desired. ∎

We now prove Lemma 4.2.

Proof: We obtain our result as follows. We show how to construct adversaries A_1 and A_2 against digital signature scheme $\mathcal{DS}$, and by using the assumption that (P, V) is a sound proof system for ρ, we show that for any adversary B against full-traceability of $\mathcal{GS}$, and any nice function $n : \mathbb{N} \to \mathbb{N}$,

$$\mathbf{Adv}^{\text{trace}}_{\mathcal{GS},B}(k, n(k)) \leq 2^{-k} + \mathbf{Adv}^{\text{unforg-cma}}_{\mathcal{DS},A_1}(k) + n(k) \cdot \mathbf{Adv}^{\text{unforg-cma}}_{\mathcal{DS},A_2}(k)$$

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 fully-traceable group signature scheme.

So, let B be a full-traceability adversary against our group signature scheme and let $\mathcal{C}$ be the set of group members that B corrupts during its attack. We first give a classification of the types of forgeries that B can produce.

Case 0. The simplest case is when the forgery $(m, (C, \pi))$ is such that $(pk_e, pk_s, m, C) \notin L_\rho$, yet the signature appears to be valid, i.e. π is a proof for a false statement. We call these forgeries of type 0.

Otherwise, the plaintext underlying C is of the form $\langle i, pk, \text{cert}, s\rangle$ for some i, pk, cert, s, user i is not part of the corrupted set, i.e. $i \notin \mathcal{C}$ and relations

1. $\mathsf{Vf}(pk_s, \langle i, pk\rangle, \text{cert}) = 1$ and
2. $\mathsf{Vf}(pk, m, s) = 1$

hold. Depending on the membership certificate, we distinguish the following two cases:

Case 1. The pair i, pk was not certified by the group manager (this is the case, for instance, when i is not a valid user identity.) In this case, the coalition manages to forge a group membership certificate cert for identity i. We call these forgeries of type 1.

Case 2. The last case is when i, pk was certified by the group manager, in which case i is a valid user identity. If this is true, then it must be the case that $i \notin \mathcal{C}$, and the (i, m) is not submitted to the signing oracle. Which in particular implies that the coalition produces a valid signature s on m. We call these forgeries of type 2.

We can immediately bound the probability that B produces a forgery of type 0. Indeed, since (P, V) is a sound proof system for ρ, for any security parameter k, and any polynomial time forger B,

$$\Pr\left[\, B \text{ outputs } (C, \pi) : (pk_e, pk_s, m, C) \notin L_\rho \text{ and } V((m, C), \pi) = 1 \,\right] \leq 2^{-k} \tag{12}$$

The details of adversaries A_0, A_1 are given in Figure 5. Adversary A_1 is intended to run in the experiment $\mathbf{Exp}^{\text{unforg-cma}}_{\mathcal{DS}, A_1}(k)$, and as such, it has access to a signing oracle $\mathsf{Sig}(sk_s, \cdot)$. The adversary creates an instance of the group signature scheme, and uses the signing oracle to certify individual public keys. More precisely, it starts by generating the public/secret pair used for encryption, (pk_e, sk_e) by itself, and then produces individual certificates using the signing oracle, i.e. for each user j, it generates a pair of individual signing/verifying keys (sk_j, pk_j) and submits $\langle j, pk_j\rangle$ to the oracle. The answers returned by the oracle are individual certificates $\text{cert}_j = \mathsf{Sig}(sk_s, \langle j, pk_j\rangle)$. Finally, it runs adversary B against $\mathcal{GS}$. In order to perfectly emulate the environment in which B is run, adversary A_1 needs to answer the queries that B makes, i.e. it needs to produce group signatures on messages of B's choice, and also it needs to return to B the secret group signing key of users that it corrupts. This can be easily done, since A_1 possesses the secret keys of all the users.

When B stops and produces an attempted forgery $(m, (C, \pi))$, A_1 decrypts C and obtains the certificate cert, and the pair identity/individual public key, i, pk, used to create the group signature. If the forgery that is output is of type 1 (the group manager never certifies $\langle i, pk\rangle$), then the message $\langle i, pk\rangle$ was never queried to the signing oracle and $\mathsf{Vf}(pk_s, \langle i, pk\rangle, \text{cert}) = 1$. The pair $(\langle i, pk\rangle, \text{cert})$ is therefore a successful forgery in the $\mathbf{Exp}^{\text{unforg-cma}}_{\mathcal{DS}, A_1}(k)$ experiment. Therefore, for a fixed security parameter k, if E_1 is the event that B outputs a forgery of type 1 in the experiment describing an attack on full-traceability of $\mathcal{GS}$, then

$$\mathbf{Adv}^{\text{unforg-cma}}_{\mathcal{DS}, A_1}(k) = \Pr\left[\, E_1 \,\right] \tag{13}$$

We now show how to construct an adversary that exploits the second type of forgeries. Adversary A_2 is intended to run in the experiment $\mathbf{Exp}^{\text{unforg-cma}}_{\mathcal{DS}, A_2}(k)$, 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 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. Then, it generates all but one individual signing-verifying keys $\{sk_j, pk_j\}_{\{j \neq i\}}$. The individual signing key for user i will be the signing key sk of the oracle to which A_2 has access. Furthermore, A_2 creates individual membership certificates simply by signing $\langle j, pk_j\rangle$ for users $j \neq i$, and $\langle i, pk\rangle$ for user i.

Algorithm $A_1^{\mathsf{Sig}(sk,\cdot)}(pk)$
$R \xleftarrow{\$} \{0,1\}^{p(k)}$
$(pk_e, sk_e) \xleftarrow{\$} \mathsf{K_e}(1^k)$
$gpk \leftarrow (k, R, pk_e, pk_s)$
$gmsk \leftarrow (k, n, pk_e, sk_e, pk_s)$
For $j \leftarrow 1$ to $n(k)$ do
$(sk_j, pk_j) \xleftarrow{\$} \mathsf{K_s}(k)$
$\mathrm{cert}_j \xleftarrow{\$} \mathsf{Sig}(sk, \langle j, pk_j \rangle)$
$\boldsymbol{gsk}[j] \leftarrow (j, pk_j, sk_j, \mathrm{cert}_j, gpk)$
EndFor
Run B
Until B stops and outputs $(m, (C, \pi))$
Parse $\mathsf{Dec}(sk_e, C)$ as $\langle i, pk', \mathrm{cert}', s \rangle$
Return $(\langle i, pk' \rangle, \mathrm{cert}')$

Adversary $A_2^{\mathsf{Sig}(sk,\cdot)}(pk)$
$R \xleftarrow{\$} \{0,1\}^{p(k)}$
$(pk_e, sk_e) \xleftarrow{\$} \mathsf{K_e}(k)$; $(pk_s, sk_s) \xleftarrow{\$} \mathsf{K_s}(k)$
$i \xleftarrow{\$} \{1, 2, ..., n(k)\}$
For $j \leftarrow 1$ to $n(k)$ (except i) do
$(sk_j, pk_j) \xleftarrow{\$} \mathsf{K_s}(k)$
$\mathrm{cert}_j \xleftarrow{\$} \mathsf{Sig}(sk_s, \langle j, pk_j \rangle)$
$\boldsymbol{gsk}[j] \leftarrow (j, pk_j, sk_j, \mathrm{cert}_j, gpk)$
Endfor
$\mathrm{cert} \xleftarrow{\$} \mathsf{Sig}(sk_s, \langle i, pk \rangle)$
$gpk \leftarrow (R, pk_e, pk_s)$
$gmsk \leftarrow (pk_e, sk_e, pk_s)$
Run B
- if in the choose stage B requests $\boldsymbol{gsk}[i]$ the algorithm fails;
- when B makes an oracle query (j, m) do
$\sigma \xleftarrow{\$} \mathsf{GSig}(\boldsymbol{gsk}[j], m)$; return σ to B;
- when B makes an oracle query (i, m) do
$s \xleftarrow{\$} \mathsf{Sig}(sk, m)$;
$C \xleftarrow{\$} \mathsf{Enc}(pk_e, \langle i, pk, \mathrm{cert}, s \rangle; r)$;
$\pi \xleftarrow{\$} P(k, (pk_e, pk_s, m, C), (i, pk, \mathrm{cert}, s, r))$;
return (C, π) to B
Until B stops and returns $(m, (C, \pi))$
Parse $\mathsf{Dec}(sk_e, C)$ as $\langle (i, pk', \mathrm{cert}', s) \rangle$
Return (m, s)

Figure 5: Construction of two cma-forger A_1 and A_2 against $\mathcal{DS}$ from an adversary B against full-traceability of $\mathcal{GS}$. Note that $n(\cdot)$ is a fixed nice function.

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 $j \neq i$, are answered by using secret group signing key of j (which it A_2 created by itself). Requests of the type (M, i) 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).

Furthermore, the message (i, m) was not queried to the group signing oracle (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. We can now relate the probability that B outputs a forgery of type 2 in the full-traceability experiment with the success probability of A_2 in the $\mathbf{Exp}^{\text{unforg-cma}}_{\mathcal{DS}, A_2}(k)$ experiment, by observing that A_2 wins precisely when B outputs a successful forgery *and* A_2

correctly guesses the identity i that underlies the forged signature. Since the latter event happens with probability $1/n(k)$ and is independent of the former we obtain:

$$\mathbf{Adv}^{\text{unforg-cma}}_{\mathcal{DS},A_2}(k) = \frac{1}{n}(k) \cdot \Pr[\, E_2 \,] \tag{14}$$

Using Equations (12),(13) and (14) we can bound the advantage of B as desired:

$$\begin{aligned}
\mathbf{Adv}^{\text{trace}}_{\mathcal{GS},B}(k, n(k)) &= \Pr[\, B \text{ wins in the experiment } \mathbf{Exp}^{\text{trace}}_{\mathcal{GS},B}(k) \,] \\
&= \Pr[\, B \text{ outputs a successful forgery } (m, (C, \pi)) \,] \\
&= \Pr[\, ((pk_e, pk_s, m, C) \notin L_\rho \text{ and } V((m, C), \pi) = 1) \text{ or } E_1 \text{ or } E_2 \,] \\
&\leq 2^{-k} + \mathbf{Adv}^{\text{unforg-cma}}_{\mathcal{DS},A_1}(k) + n(k) \cdot \mathbf{Adv}^{\text{unforg-cma}}_{\mathcal{DS},A_2}(k)
\end{aligned}$$

∎

5 Dynamic Groups and Other Extensions

In the previous sections we considered static group signatures schemes. i.e., schemes where the size and membership of the group do not change over time. In this section we discuss various extensions of the basic definition, including schemes where the group is dynamic. i.e., members can join and leave the group over time. Following standard terminology [24] we refer to these groups as partially or fully dynamic.

Partially dynamic groups. These are groups supporting either join (incremental) or leave (decremental) operation. Here we concentrate on incremental groups, and postpone the discussion of leave operation to the next paragraph. In an incremental group signature scheme, the key generation algorithm produces (beside the group public key *gpk*) two secret keys: an issuing key *gisk* and an opening key *gmsk*. No signing keys $\boldsymbol{gsk}$ are output by the key generation algorithm. The two keys *gisk*, *gmsk* are given to two different group managers, one of which has authority on the group membership, and the other has authority on traceability. Using *gisk*, the key issuer can generate (possibly via an interactive process) signing keys $\boldsymbol{gsk}[i]$ and distribute them to perspective group members (which we identify with the index i). The basic definition of security is essentially the same as described in the previous sections. Key *gisk* is given to the adversary in the definition of anonymity, but not in the definition of traceability, as knowledge of this key would allow to generate "dummy" group members and use their keys to sign untraceable messages. Alternatively, one can postulate that if the signature opener cannot trace a signature, then he will blame the key issuer. Our construction (and proof of security) from section 4 can be easily extended to support incremental group operations, with the certificate creation key sk_s used as *gisk*.

Although the above definition allows the group to change over time, the security properties are still static: a signer i that join the group at time t, can use the newly acquired key $\boldsymbol{gsk}[i]$ to sign documents that predate time t. This problem can be easily solved enhancing the signatures with an explicit time counter, and using the technique of forward security. Before explaining the connection with incremental groups, we discuss forward security, which may be a desirable security feature on its own. As for standard digital signature schemes [7], forward security for group signatures is defined using a key evolution paradigm. The lifetime of the public key is divided into time periods, and signing keys of the group members change over time, with the key of user i at time j denoted

$\boldsymbol{gsk}_j[i]$. At the end of each time period, each user updates his key using an update algorithm $\boldsymbol{gsk}_{j+1}[i] = \mathsf{GUpd}(\boldsymbol{gsk}_j[i])$. Although the forward security requirement for group signature schemes was already considered before [29], that definition has a serious security flaw: [29] only requires that no adversary, given $\boldsymbol{gsk}_t[i]$, can efficiently recover $\boldsymbol{gsk}_j[i]$ for any $j < t$. This is not enough.[2] We define forward secure group signature schemes requiring that an attacker should not be able to produce *valid signatures* for any earlier time period.

Our scheme from Section 4 can be modified to achieve forward security, by replacing the signature scheme used by group members with a forward secure one $\mathcal{DS} = (\mathsf{K_s}, \mathsf{Vf}, \mathsf{Sig}, \mathsf{Upd})$.time period j, user i has the group signing key $\boldsymbol{gsk}_j[i] = (k, R, i, pk_i, sk_{i,j}, \mathrm{cert}_i, pk_e, pk_s)$, where $sk_{i,0}$ is obtained by running the key generation algorithm of $\mathcal{DS}$ and $sk_{i,j}$ is the result of evolving $sk_{i,0}$ j times. The key update algorithm GUpd for the group signature scheme is $\mathsf{GUpd}(\boldsymbol{gsk}_j[i]) = (k, R, i, pk_i, \mathsf{Upd}(sk_{i,j}), \mathrm{cert}_i, pk_e, pk_s)$, and group membership certificates are preserved across different time periods, since the individual public keys remain unchanged. If the individual signature scheme is forward secure, and the signature scheme used to certify individual scheme is secure against chosen message attack, then we postulate that the group signature scheme described above is a secure group signature scheme.

In the context of dynamic group signature schemes, signing keys are not stolen by an adversary, but given by the key issuer to the group members. Still, if the signature scheme is forward secure, then the group member cannot use the (legitimately obtained) key to sign documents the predate the joining time. A (forward) secure partially dynamic group signature scheme supporting the join operation is obtained letting the key issuer generate key $\boldsymbol{gsk}_t[i]$ when user i joins the group at time t.

FULLY DYNAMIC GROUPS. Now consider a group where members can both join and leave the group. The issue of members leaving the group is much more delicate than the one of members joining the group, and several alternative (and incompatible) definitions are possible. As for partially dynamic groups, the signing keys $\boldsymbol{gsk}_j[i]$ may vary over time, but this time also the public key gpk_j is allowed to change. Consider a signature σ produced (using key $\boldsymbol{gsk}_a[i]$) at time a by some user i who belonged to the group at time 1, but not at times 0 and 2, Now, say σ is verified at time b (using key gpk_b). When should σ be accepted by the signature verification algorithm? There are two possible answers: (1) if i was a group member when then signature was generated, or (2) if i belonged to the group at the time verification algorithm is invoked.[3] Clearly, there is no "right" answer, and what definition should be used depends on the application. In either case, different kinds of inefficiency are necessarily introduced in the protocol. In case (2), σ should be accepted if $b = 1$, but not if $b = 0$ or $b = 2$. In particular, the public keys gpk_j must be different. This is undesirable because it requires the verifier to continuously communicate with the group manager to update the group public key.[4] Moreover, this definition raises potential anonymity problems: verifying the same signature against different public keys gpk_j, one can determine when the signer joined and left the group, and possibly use this information to discover the signer identity. Now consider case (1), where signatures are valid only if signer belongs to the group at the time the

[2]Consider a scheme where $\boldsymbol{gsk}_j[i] = (k_j; h_j)$ $\mathsf{GUpd}(k_j; h_j) = k_j; g(h_j)$ for some one way permutation g, and only the first part of the key k_j is used by the signing algorithm. Such a scheme would certainly satisfy the definition of [29], but it is clearly not forward secure: even if the past keys cannot be recovered, the adversary can still forge messages in the past!

[3]Notice that in the first case, signatures should remain valid (and the signer anonymous) even after the signer leaves the group, while in the second case removing the signer from the group should immediately invalidate all of its signatures.

[4]Alternatively, one can consider public keys of the form $gpk_j = (gpk, j)$, where the time dependent part can be supplied autonomously by the verifier. But it is easy to see that in this case the life time of secret key $\boldsymbol{gsk}[i]$ needs to be decided in advance when the user i joins the group.

signature is issued. This time, the public key may stay the same throughout the lifetime of the group, but the key update function $\boldsymbol{gsk}_j[i] = \mathsf{GUpd}(\boldsymbol{gsk}_{j-1}[i])$ should not be publicly computable by the group members. This introduces inefficiency, as the group members now need to interact with the key issuer to update their signing key from one time period to the next.

Our construction can be easily adapted to satisfy either definition of security for fully dynamic groups, e.g., by rekeying the entire group at the end of each time period. Due to the high (but unavoidable) cost of rekeying operations, fully dynamic group signatures should be used only if required by the application, and in all other situations the simpler incremental groups are clearly preferable.

DISHONEST GROUP MANAGERS. The last extension to the definition we discuss pertains the case where the group manager holding *gmsk* is not honest. The experiments we gave in Section 2, only capture the situation where the adversary obtained the group manager's secret key. If the group manager is truly dishonest, one should also assume that he does not behave as prescribed when applying the opening algorithm. For example, when asked to open a signature he can falsely accuse an arbitrary user (i.e. he does not use his secret key), or claim that the signature can not be open (e.g. he changes the secret key).

We consider a solution in which when opening a signature the group manager (on input opening key *gmsk*, message m and signature σ) outputs not only a user identity i, but also a "proof" τ. This proof can be (publicly) verified by a "judging" algorithm GJudge (which is added to the syntax of a group signature scheme,) such that if $\mathsf{Open}(gmsk, m, \sigma) = (i, \tau)$ then $\mathsf{GJudge}(m, \sigma, i, \tau) = \mathsf{true}$. Within this framework, it is possible to formally capture security requirements regarding dishonest behavior of the group manager, as described at the beginning of this section.

Our scheme from Section 4 can be easily modified accordingly. We use an authenticated encryption scheme $\mathcal{AE}$ in which when decrypting a ciphertext, one also recovers the randomness that was used to create it. Opening a signature (C, π) is done as follows: the group manager recovers the plaintext underlying C as $\langle i, pk_i, \mathrm{cert}_i, s\rangle$ and also recovers the randomness r that was used to create C. Then, it outputs (i, τ), where $\tau = (\langle i, pk_i, \mathrm{cert}_i, s\rangle, r)$, is the "proof" that the signature was created by user i. The judging algorithm GJudge simply checks whether C is the result of encrypting $\langle i, sk_i, pk_i, \mathrm{cert}_i, s\rangle$ with pk_e using randomness r.

References

[1] R. Anderson. Invited talk. Fourth Annual Conference on Computer and Communications Security,1997.

[2] G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A practical and provably secure coalition-resistant group signature scheme. In M. Bellare, editor, *CRYPTO'00*, volume 1880 of *LNCS*, pages 255–270. Springer-Verlag, 2000.

[3] G. Ateniese and G. Tsudik. Quasi-efficient revocation in group signature schemes. Available at http://eprint.iacr.org/2001/101.pdf.

[4] G. Ateniese and G. Tsudik. Group signatures à la carte. In *ACM Symposium on Discrete Algorithms*, pages 848–849. ACM Press, 1999.

[5] G. Ateniese and G. Tsudik. Some open issues and directions in group signature. In *Financial Crypto'99*, volume 1648 of *LNCS*, pages 196–211. Springer-Verlag, 1999.

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

[7] M. Bellare and S. Miner. A forward-secure digital signature scheme. In M. Wiedner, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 431–448. Springer-Verlag, 1999.

[8] M. Bellare, H. Shi, and C. Zhang. Foundations of group signatures: the case of dynamic groups.

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

[10] E. Bresson and J. Stern. Efficient revocation in group signatures. In *PKC'2001*, volume 1992 of *LNCS*, pages 190–206. Springer-Verlag, 2001.

[11] J. Camenisch. Efficient and generalized group signature. In *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 465–479. Springer-Verlag, 1997.

[12] J. Camenisch and M. Michels. A group signature scheme with improved efficiency. In K. Ohta and D. Pei, editors, *ASIACRYPT'98*, volume 1514 of *LNCS*, pages 160–174. Springer-Verlag, 1999.

[13] J. Camenisch and M. Stadler. Efficient group signatures schemes for large groups. In B.Kaliski, editor, *CRYPTO'97*, volume 1294 of *LNCS*, pages 410–424. Springer-Verlag, 1997.

[14] D. Chaum and E. van Heyst. Group signatures. In D. W. Davis, editor, *EUROCRYPT'91*, volume 547 of *LNCS*, pages 257–265. Springer-Verlag, 1991.

[15] L. Chen and T. P. Pedersen. New group signature schemes. In A. DeSantis, editor, *EUROCRYPT'94*, volume 950 of *LNCS*, pages 171–181. Springer-Verlag, 1994.

[16] A. DeSantis and M. Yung. Cryptographic applications of the non-interactive metaproof and many-prover systems. In A. Menezes and S. Vanstone, editors, *CRYPTO'90*, number 537 in LNCS, pages 366–377. Springer-Verlag, 1990.

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

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

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

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

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

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

[23] M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *STOC'90*, pages 427–437, 1990.

[24] N. I. of Standards and Technology. Dictionary of algorithms and data structures. http://www.nist.gov/dads/.

[25] H. Petersen. How to convert any digital signature scheme into a group signature scheme. In *Proceedings of Security Protocols Workshop'97*, pages 177–190, 1997.

[26] C. Rackoff and D. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In *CRYPTO'91*, pages 433–444, 1992.

[27] J. Rompel. One-way functions are necessary and sufficient for secure signatures. In *22nd Annual Symposium on Theory of Computing*, pages 387–394. ACM, ACM Press, 1990.

[28] A. Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *FOCS'99*, pages 543–553, 1999.

[29] D. Song. Practical forward-secure group signature schemes. In *ACM Symposium on Computer and Communication Security*, pages 225–234, November 2001.