Group Signatures with Efficient Concurrent Join

Aggelos Kiayias[1]** and Moti Yung[2]

[1] Computer Science and Engineering, University of Connecticut
Storrs, CT, USA. aggelos@cse.uconn.edu
[2] RSA Laboratories, Bedford, MA, and
Computer Science, Columbia University
New York, NY, USA. moti@cs.columbia.edu

Abstract. A group signature is a basic privacy mechanism. The group joining operation is a critical component of such a scheme. To date all secure group signature schemes either employed a trusted-party aided join operation or a complex joining protocol requiring many interactions between the prospective user and the Group Manager (GM). In addition no efficient scheme employed a join protocol proven secure against adversaries that have the capability to dynamically initiate multiple concurrent join sessions during an attack.

This work presents the first efficient group signature scheme with a simple Joining protocol that is based on a "single message and signature response" interaction between the prospective user and the GM. This single-message and signature-response registration paradigm where no other actions are taken, is the most efficient possible join interaction and was originally alluded to in 1997 by Camenisch and Stadler, but its efficient instantiation remained open till now.

The fact that joining has two short communication flows and does not require secure channels is highly advantageous: for example, it allows users to easily join by a proxy (i.e., a security officer of a company can send a file with all registration requests in his company and get back their certificates for distribution back to members of the company). It further allows an easy and non-interactive global system re-keying operation as well as straightforward treatment of multi-group signatures. We present a strong security model for group signatures (the first explicitly taking into account concurrent join attacks) and an efficient scheme with a single-message and signature-response join protocol.

1 Introduction

Group signatures is a useful anonymous non-repudiable multi-use credential primitive that was introduced by Chaum and Van Heyst [17]. It involves a group of users, each holding a membership certificate that allows a user to issue a publicly verifiable signature while hiding the identity of the actual signer within the group. The public-verification procedure employs only the public-key of the group. Furthermore, in the case of any dispute or abuse, it is possible for the

** Research partly supported by NSF CAREER Award CNS-0447808.

group manager (GM) to "open" an individual signature and reveal the identity of its originator.

Constructing an efficient group signature has been a research target for many years, see e.g., [18, 16, 13, 14, 9, 26, 3, 1, 11, 24, 8, 2, 10, 12]. A scalable scheme that provides constant signature size and has resistance to attacks by coalitions of users was given in [1]. Earlier constructions were designed without a formal model and definition of security of such schemes, and thus with partial security proofs at the best case (while many were actually broken).

A central issue in group signatures has been the way by which users join the group. Recently, [5] gave the first formal model of a somewhat "relaxed" group signature primitive where a trusted party generates and hands out all users' keys. They also produced a generic solution thus demonstrating the polynomial-time plausibility of their notion of trusted-party aided join group signatures. This is in contrast with users who dynamically join the system and get their individual keys by interacting with the group manager (as in the protocol of [1]). Dynamic joins that allow users to register sequentially were studied formally in [23, 25] where efficient constructions were given and in [5, 6] where a generic plausibility proof was provided.

The most efficient and conceptually simple joining procedure for a group signature scheme (what we will call the "single-message and signature-response paradigm") was illustrated by Camenisch and Stadler [16] who sketched a generic solution (which was followed in careful details in [5, 6]). In this type of joining protocol, the prospective user has an appropriately distributed secret x' and it computes a one way function f on it to obtain $x = f(x')$. The user sends x to the GM who, in turn, signs x and returns the signature v to the user using an appropriate signing algorithm. This completes the interaction of the join protocol. The possession ofthe signature v on $x = f(x')$ enables a user to sign anonymously a message m by simply encrypting x probabilistically into ψ (under the GM's public key or whatever entity is supposed to execute the opening algorithm) and by providing a zero-knowledge proof of (i) the fact that the ψ is an encryption of some x known to the prover, (ii) the fact that the prover knows x' a preimage of that x under f, (iii) the fact that the prover knows a signature issued by the GM on that x.

While the Camenisch-Stadler approach is elegant and advantageous (as we argue below), its instantiation by an efficient scheme turned out to be elusive, since the many schemes that have been suggested in the last eight years approximated it but none really employed it. In fact, all the efficient schemes in the non-trusted-party-aided joining setting that were not broken used additional communications during the join protocol usually to assure that certain constraints and certain knowledge of the joining user is present, i.e., the prospective user had to engage in an interactive zero-knowledge proof with the GM. It was not at all apparent whether the single-message and signature-response join would actually be instantiable in an *efficient* manner in a *provably secure scheme*. Moreover the employment of such proofs of knowledge has the usual shortcomings with respect to adversaries operating in the concurrent setting (namely, rewinding cannot be employed and a "straight-line" approach needs to be followed that makes the joining protocols even more involved).

To conclude the motivation for our result, we summarize the advantages of a group signature employing a single-message and signature-response joining protocol:

1. *Concurrency:* Joining of users can be done concurrently where a batch of users join at the same time. This enables group managers over the Internet (where servers are multi-thread machines).
2. *Proxy Join:* Users can be joined by a proxy collecting all their requests and then collecting the responses from the group manager; this is a very effective way to enroll companies and organizations by delegating collection and distribution to security officers. It is highly effective in enrolling to an identity escrow scheme without the need for random oracle proofs.
3. Multi-Group Scenario: There may be a number of groups; since single-message and signature-response joins require essentially no interaction between the GM and the prospective user users may accumulate many GM membership signatures on the same x value non-interactively thus easily becoming members of multiple-groups.

1.1 Our result

In this work we implement the first group signature scheme with a single-message and signature-response join protocol to be exploited for concurrent joins and other advantages as above, thus implementing efficiently the Camenisch-Stadler approach for the first time[3].

We start by presenting the first model of "group signature with concurrent joins" which builds on the recent formal models and consists of a set of attacks. We note that in a privacy primitive interacting users may be conducting simultaneous attacks against each other and these need to be captured formally. We call our attacks: misidentification attack, framing attack and anonymity attack and is an extension of our sequential-join formal model for group signatures in [25]. We then implement a scheme based on specific assumptions and prove its security. The scheme allows adversarial opening of signatures and its signature size is only about twice the size of the scheme of [1] (that did not allow for adversarial opening or concurrent join attacks).

From a technical viewpoint we employ a number of complex primitives including the digital signature scheme of Boneh and Boyen [7] (hence referred to as the BB signature) as well as verifiable encryption for discrete-logarithms that are based on the Paillier encryption function [27, 20, 15, 22].

[3] In some recent schemes of group signatures and related primitives based on dynamic accumulators [28, 19], a simple two message join was implemented; nevertheless this was to be followed by local modifications of keys of *all existing users*; we do not consider such a protocol efficient. In our solution, keys of other users are unaffected when new members are introduced to the group.

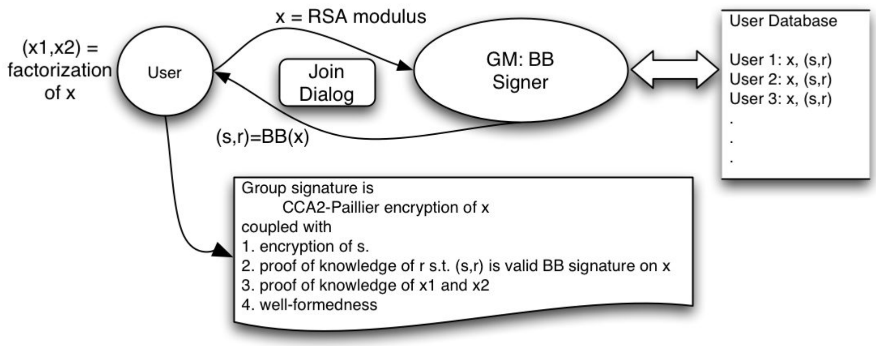

Fig. 1. Overview of our general group signature design. The BB signature can be substituted by potentially other signatures that are suitable for algebraic encryption with efficient validity proof.

A novelty of our technical approach (and perhaps an explanation why we manage to achieve an efficient single-message and signature-response join) is that we deviate from most of group signature literature by instantiating the one-way function employed by the prospective user during the join with multiplication instead of exponentiation. Our general design approach is outlined in figure 1: users sample an RSA modulus and merely obtain a BB certificate on it. This modest interaction (which is simply a PKI registration in a domain employing RSA moduli with a BB signature for certification) allows users to sign as group members.

Our security proofs follow a modular approach: in a nutshell, a misidentification adversary is turned into a BB-forger, a framing adversary is turned into a factoring algorithm and an anonymity attacker is turned into a CCA2 adversary against the encryption algorithm we employ. The group signature itself is based on the Fiat-Shamir paradigm, by essentially turning an identity escrow (anonymous identification) system into a signature and employing a random oracle. We note that the interactive version of our group signature yields an identity escrow scheme in a straightforward manner that can also have concurrent group signing by employing general transformation techniques for Σ-protocols, e.g. [21].

2 Preliminaries

Interactive Turing Machines and Concurrent Executions. A two-party protocol is a pair of probabilistic polynomial-time bounded Interactive Turing machines $\langle \mathsf{A}, \mathsf{B} \rangle$. Each of A, B has a private input tape, work-tapes, a (joint) communication tape and a private output tape. An execution of a protocol $\langle \mathsf{A}, \mathsf{B} \rangle$ on inputs x, y for the two players will be denoted by $[\mathsf{A}(x), \mathsf{B}(y)]$. For an execution of a protocol we will consider the following random variables: (i) $\mathsf{Trans}[\mathsf{A}(x), \mathsf{B}(y)]$ is the contents of the communication tape after the two parties terminate. (ii) $\mathsf{Out}_{\mathsf{A}}[\mathsf{A}(x), \mathsf{B}(y)]$ is the contents of the private output tape of player A after termination. (iii) $\mathsf{Out}_{\mathsf{B}}[\mathsf{A}(x), \mathsf{B}(y)]$ is the contents of the private output tape of player B after termination.

Now suppose that $\mathcal{P} = \langle \mathsf{A}, \mathsf{B} \rangle$ is a protocol. An "interface oracle" for concurrent simulation of player B, denoted by $\mathcal{I}[\mathcal{P}_{\leftrightarrow \mathsf{B}(y)}]$, is an oracle that accepts the following queries:

Q1. $\mathtt{Start-Session}$: The interface oracle $\mathcal{I}[\mathcal{P}_{\leftrightarrow \mathsf{B}(y)}]$ initiates a session for the protocol $\mathcal{P}$: it selects a session identifier s and if B is the player that goes first in the protocol $\mathcal{P}$, the interface simulates the first move of B on input y; the interface returns as answer to the $\mathtt{Start-Session}$ query the session identifier s and the output of the simulation of player B's first move (if any). The interface keeps a database with the state of player B for the session identifier s; the state includes all coin tosses of B, and the contents of all tapes including the communication tape.

Q2. $\mathtt{Advance-Session}(s, msg)$ The interface oracle looks up the table of sessions and recovers the state of player B for the session with identifier s (if there is no such session the interface returns $\perp$ as answer to the oracle query). If session s exists the interface appends msg to the communication tape of the session and continues the simulation of player B (as if msg is the message that is written to the communication tape of player B by player A).

We will use the notation $M^{\mathcal{I}[\mathcal{P}_{\leftrightarrow \mathsf{B}(\cdot)}]}$ to denote any probabilistic Turing machine M that has access to an interface oracle as defined above. Note that the interface oracle $\mathcal{I}[\mathcal{P}_{\leftrightarrow \mathsf{A}(x)}]$ (for concurrent executions of player A in the protocol $\mathcal{P}$) can be defined in the same fashion as above. Frequently protocol executions are stateful, e.g. there is a database, or state St in general that an instantiation of the protocol $\mathcal{P}$ may consult. This state St will be maintained by the interface oracle $\mathcal{I}$. In this case we will write $\mathcal{I}_{St}[\mathcal{P}_{\leftrightarrow \mathsf{B}(\cdot)}]$. In the case that a TM M has access to a stateful interface oracle $\mathcal{I}$ we will write $M^{\mathcal{I}_{St}[\mathcal{P}_{\leftrightarrow \mathsf{B}(\cdot)}]}$. Depending on the case, $\mathcal{I}$ may modify the state St or even allow read and write access to St by M.

Bilinear Maps. Let $\mathbb{G}_1, \mathbb{G}_2$ two groups of prime order p so that (i) $\mathbb{G}_1 = \langle g_1 \rangle$ and $\mathbb{G}_2 = \langle g_2 \rangle$; (ii) $\tau : \mathbb{G}_2 \to \mathbb{G}_1$ is an isomorphism with $\tau(g_2) = g_1$ and (iii) $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a bilinear map. We remark that in many cases it can be that $\mathbb{G}_1 = \mathbb{G}_2$ (and in this case ψ would be the identity mapping). Let $\mathbb{G}_1 = \langle g_1 \rangle, \mathbb{G}_2 = \langle g_2 \rangle$ groups as above with $|\mathbb{G}_1| = |\mathbb{G}_2| = p$; a bilinear map is a map e s.t. for all $(u, v) \in \mathbb{G}_1 \times \mathbb{G}_2$ it holds that $e(u^x, v^y) = e(u, v)^{xy}$ and $e(g_1, g_2) \neq 1$.

Intractability Assumptions. We will employ the following four intractability assumptions:

The *Strong Diffie Hellman Assumption* (SDH) was put forth by Boneh and Boyen [7]. The q-SDH problem over two groups $\mathbb{G}_1, \mathbb{G}_2$ is defined as follows:

given a $(q+2)$-tuple $\langle g_1, g_2, g_2^{\gamma}, \ldots, g_2^{(\gamma)^q} \rangle$ as input, output a pair $(g_1^{\frac{1}{\gamma+x}}, x)$ where $x \in \mathbb{Z}_p^*$. The q-SDH assumption suggests that any probabilistic polynomial-time (PPT) algorithm solving the q-SDH problem has negligible success probability. When q is any polynomial-time function on the security parameter we will write simply SDH.

The *Strong-RSA* problem [4] is as follows: given $n, z \in QR(n)$, where $QR(n)$ is the group of quadratic residues of $\mathbb{Z}_n^*$ asks for two integers $u, e > 1$ so that $u^e \equiv_n z$. The Strong-RSA assumption suggests that any PPT algorithm solving the Strong-RSA problem has negligible success probability.

The *Linear Decisional Diffie Hellman* assumption (Linear-DDH) [8] is as follows: the distribution of tuples of the form $(u, v, h, u^{\alpha}, v^{\beta}, h^{\alpha+\beta})$ where $u, v, h \leftarrow_R \mathbb{G}_1$ and $\alpha, \beta \leftarrow_R \mathbb{Z}_p$, is computationally indistinguishable from the distribution of tuples of the form $(u, v, h, u^{\alpha}, v^{\beta}, \eta)$ where where $u, v, h, \eta \leftarrow_R \mathbb{G}_1$ and $\alpha, \beta \leftarrow_R \mathbb{Z}_p$. The Linear-DDH is assumed to be true, even in groups where DDH fails (e.g., groups $\mathbb{G}_1$ for which we have a bilinear mapping).

The *Decisional Composite Residuosity* (DCR) assumption [27] is defined as follows: it is computationally hard to distinguish between the distributions of tuples of the form $(N, u^N \bmod N^2)$ where N is an RSA safe composite modulus and $u \leftarrow_R \mathbb{Z}_{N^2}^*$ and the distribution of tuples of the form (N, v) where N is an RSA safe composite modulus and $v \leftarrow_R \mathbb{Z}_N^*$.

3 Group Signatures with Concurrent Join : Modeling

In this section we give the formal definition of group signatures with concurrent join. First we start with the syntax of the signature. The parties that are involved in the scheme include the Group Manager (GM), the Users and the Verifiers.

Definition 1. *A group signature scheme with concurrent joins is a digital signature scheme that is comprised of the following five procedures:*

`SETUP`*: it is a probabilistic algorithm that on input a security parameter 1^{ν}, it outputs the group public key $\mathcal{Y}$ (including all system parameters) and the secret key $\mathcal{S}$ for the GM.* `SETUP` *initializes a public state string $St = (St_{users}, St_{join-trans})$ with two components $St_{users} = \epsilon$ and $St_{join-trans} = \epsilon$. The public state string St will hold the user identity database and the database of the Join protocol transcripts. This information will be publicly available and will grow as more users are introduced into the system.*

`JOIN`*: A* protocol *between the GM and a user that results in the user becoming a new group member. The user's output is a* membership certificate *and a* membership secret. *We will denote the i-th user's membership certificate by* cert_i *and the corresponding membership secret by* sec_i.
Since `JOIN` *is a two-party protocol, its specification includes the description of two interactive Turing Machines (ITM)* $\mathsf{J}_{\mathsf{user}}, \mathsf{J}_{\mathsf{GM}}$. *Only* $\mathsf{J}_{\mathsf{user}}$ *will have a private output.*
According to the notations of section 2 an execution of the protocol is denoted as $[\mathsf{J}_{\mathsf{user}}(St, \mathcal{Y}) \leftrightarrow \mathsf{J}_{\mathsf{GM}}(St, \mathcal{Y}, \mathcal{S})]$ *and has two "output" components:*

1. the user private output, $\langle i, \mathsf{cert}_i, \mathsf{sec}_i\rangle \leftarrow \mathsf{User}[\mathsf{J}_{\mathsf{user}}(St, \mathcal{Y}) \leftrightarrow \mathsf{J}_{\mathsf{GM}}(St, \mathcal{Y}, \mathcal{S})]$, *and*

2. the public transcript, $\mathsf{transcript}_i \leftarrow \mathsf{Trans}[\ \mathsf{J}_{\mathsf{user}}(St, \mathcal{Y}) \leftrightarrow \mathsf{J}_{\mathsf{GM}}(St, \mathcal{Y}, \mathcal{S})]$.

After a successful execution of `JOIN` *the following updates are made:* $St_{users} = St_{users}||\langle i\rangle$ *and* $St_{join-trans} = St_{join-trans}||\ \langle i, \mathsf{transcript}_i\rangle$. *The identity-tag* i *will be selected from a space of possible identity tags denoted by* ID.

`SIGN`: *A probabilistic algorithm that given the group's public-key, a membership certificate, a membership secret and a message* m*, it outputs a group signature for the message* m*. We write* $\mathtt{SIGN}(\mathcal{Y}, \mathsf{cert}_i, \mathsf{sec}_i, m)$ *to denote the application of the signing algorithm on the message* m.

`VERIFY`: *An algorithm for establishing the validity of an alleged group signature on a message with respect to a group public-key. If* σ *is a signature on a message* m*, then we have* $\mathtt{VERIFY}(\mathcal{Y}, m, \sigma) \in \{\top, \bot\}$.

`OPEN`: *An algorithm that, given a message, a valid group signature on it, a group public-key, the GM's secret-key and the public-state it determines the identity of the signer. In particular* $\mathtt{OPEN}(m, \sigma, St, \mathcal{Y}, \mathcal{S}) \in St_{users} \cup \{\bot\}$.

Notation. Below we will introduce a helpful notation for describing the relationship between transcripts and membership certificates and secrets. Given $\langle \mathcal{Y}, \mathcal{S}\rangle \leftarrow \mathtt{SETUP}(1^\nu)$ we define the following relations over strings based on $\mathcal{Y}$ and some public state St,

$\langle i, \mathsf{cert}, \mathsf{sec}\rangle \leftrightharpoons_{(\mathcal{Y}, St)} \mathsf{transcript}$ if there exist coin tosses ρ for J_{GM} and $\mathsf{J}_{\mathsf{User}}$ so that

$$\langle i, \mathsf{cert}, \mathsf{sec}\rangle = \mathsf{User}[\mathsf{J}_{\mathsf{User}}(St, \mathcal{Y}) \leftrightarrow \mathsf{J}_{\mathsf{GM}}(St, \mathcal{Y}, \mathcal{S})](\rho)$$

and

$$\mathsf{transcript} = \mathsf{Trans}[\mathsf{J}_{\mathsf{User}}(St, \mathcal{Y}) \leftrightarrow \mathsf{J}_{\mathsf{GM}}(St, \mathcal{Y}, \mathcal{S})](\rho)$$

Similarly we will define $\mathsf{cert} \leftrightharpoons_{\mathcal{Y}} \mathsf{sec}$, if there exist coin tosses ρ for J_{GM} and $\mathsf{J}_{\mathsf{User}}$ and a state St so that

$$\langle i, \mathsf{cert}, \mathsf{sec}\rangle = \mathsf{User}[\mathsf{J}_{\mathsf{User}}(St, \mathcal{Y}) \leftrightarrow \mathsf{J}_{\mathsf{GM}}(St, \mathcal{Y}, \mathcal{S})](\rho)$$

Finally we define the set of all valid public states Valid as follows: $St_0 \in \mathsf{Valid}$ if there exists a PPT Turing machine M and $\langle \mathcal{Y}, \mathcal{S}\rangle \leftarrow \mathtt{SETUP}(1^\nu)$ so that when $M^{\mathcal{I}_{St}[\mathtt{JOIN}_{\leftrightarrow GM(St,\mathcal{Y},\mathcal{S})}, \mathsf{READ}_{St}]}$ terminates it holds that $St = St_0$ and the interface oracle $\mathcal{I}$ given to M initializes $St = (\epsilon, \epsilon)$ and allows M to have read access to St through `READ` queries (that $\mathcal{I}_{St}$ allowes to M in addition to $\mathtt{Start} - \mathtt{Session}$ and $\mathtt{Advance} - \mathtt{Session}$ queries). If $\mathcal{I}_{St}$ initializes St to some $St_0 \in \mathsf{Valid}$ that is not (ϵ, ϵ) then this defines the set of all *valid extensions* of the public-state St_0 that will be denoted by Valid_{St_0}. Obviously $\mathsf{Valid} = \mathsf{Valid}_{(\epsilon,\epsilon)}$.

Correctness. Below we define the correctness of a group signature scheme that satisfies the above syntax. Note that a group signature is a tuple $\langle$`SETUP`, `JOIN`, `SIGN`, `VERIFY`, `OPEN`$\rangle$ with $\mathtt{JOIN} = \langle \mathsf{J}_{\mathsf{User}}, \mathsf{J}_{\mathsf{GM}}\rangle$.

Definition 2. *A group signature with concurrent join is correct if the following are true:*

C1. *(users are assigned unique names) For any* $St \in \mathsf{Valid}$ *it holds that* St_{users} *contains no multiply defined identity-tags, i.e., if* $St_{users} = \langle i_1 \rangle || \ldots || \langle i_K \rangle$ *it holds that* $j \neq j' \Rightarrow i_j \neq i_{j'}$.

C2. *(signing is correct) For any* $\langle \mathcal{Y}, \mathcal{S} \rangle \leftarrow \mathtt{SETUP}(1^\nu)$, *any strings* $\mathsf{cert} \leftrightharpoons_{\mathcal{Y}} \mathsf{sec}$ *and any* $m \in \{0,1\}^*$, *it holds that* $\mathtt{VERIFY}(\mathcal{Y}, m, \mathtt{SIGN}(\mathcal{Y}, \mathsf{cert}, \mathsf{sec}, m)) = \top$.

C3. *(open is correct) For any* $\langle \mathcal{Y}, \mathcal{S} \rangle \leftarrow \mathtt{SETUP}(1^\nu)$, *any* $St \in \mathsf{Valid}$, *any* $m \in \{0,1\}^*$, *and any* $\langle i, \mathsf{cert}, \mathsf{sec} \rangle \leftrightharpoons_{(\mathcal{Y}, St)} \mathsf{transcript}$ *it holds that* $\mathtt{OPEN}(m, \mathtt{SIGN}(\mathcal{Y}, \mathsf{cert}, \mathsf{sec}, m), St'', \mathcal{Y}, \mathcal{S}) = i$, *where* $St'' \in \mathsf{Valid}_{St'}$ *and* St' *is defined as follows:* $St'_{users} = St_{users} || \langle i \rangle$ *and* $St'_{join-trans} = St_{join-trans} || \langle i, \mathsf{transcript} \rangle$.

Property *C1* requires that the JOIN protocol assigns a different identity tag to all users. Property *C2* ensures the correctness of the underlying signing and verification for any valid signing key (that includes a membership secret and a membership certificate). Finally, property *C3* ensures that the OPEN algorithm correctly identifies all signers: in particular it says that if a user is introduced at some moment in the system's operation and the public-state St is updated with the user's identity tag resulting to state St' then it holds that whenever this user issues a group signature the user will be correctly identified for every public state St'' that succeeds the public-state St' of the system. We note that it may be viable to collapse *C1* and *C3* but, given the intuitiveness of the formulation, we keep them as separate properties.

Definition of Security. Security against group signatures with concurrent join, will be broken into three basic properties following the model designs of [23, 25]. The properties are formalized as games between the adversary and an entity called the interface, denoted by $\mathcal{I}$ that represents the *uncorrupted aspect of the system* in each attack.

Misidentification. In a misidentification attack, the adversary joins the system through possibly many concurrent sessions of the JOIN protocol and it attempts to produce a signature that cannot be opened to any of the users that are adversarially controlled. We note that without loss of generality we will assume that *all* users introduced in the system are adversarially controlled; this means that the goal of the adversary is to simply make the OPEN algorithm to fail. We remark that adversaries that make the OPEN algorithm to point to an innocent user will be handled in the framing attack (next paragraph).

Below, $\mathcal{I}_{St}[\mathtt{JOIN}_{\leftrightarrow GM}]$ will denote the interface oracle for concurrent simulation of the GM party in the protocol JOIN (refer to section 2 for the definition). Note that the interface $\mathcal{I}$ has access to the public state string St and it updates it accordingly whenever a new user (the adversary that is) successfully completes the JOIN dialog. Also, an oracle $\mathtt{READ}_{St}$ is provided to the adversary that allows him to read the contents of the public state database that contains the identification transcripts and user identity tags. Finally, an oracle OPEN is provided to the adversary that allows him to submit signatures and obtain the output of the opening algorithm.

The Misidentification-Attack Game $G^{\mathcal{A}}_{\mathsf{mis}}$ (denoted by $G^{\mathcal{A}}_{\mathsf{mis}}(1^\nu)$):
1. $\langle \mathcal{Y}, \mathcal{S} \rangle \leftarrow \mathtt{SETUP}(1^\nu)$; $St = (St_{users}, St_{join-trans}) = (\epsilon, \epsilon)$;
2. $\langle m, \sigma \rangle \leftarrow \mathcal{A}^{\mathcal{I}_{St}[\mathtt{JOIN}_{\leftrightarrow GM(St,\mathcal{Y},\mathcal{S})}, \mathtt{READ}_{St}, \mathtt{OPEN}]}(\mathcal{Y})$;
3. If $(\mathtt{VERIFY}(\mathcal{Y}, m, \sigma) = \top) \wedge (\mathtt{OPEN}(m, \sigma, \mathcal{Y}, \mathcal{S}, St) = \bot)$ then return $\top$ else $\bot$;

We will say that a group signature is secure against misidentification attacks with concurrent join provided that for all $\mathcal{A}$ it holds that $\mathbf{Prob}[G^{\mathcal{A}}_{\mathsf{mis}}(1^\nu) = \top] = 1 - \mathsf{negl}(\nu)$.

Framing. In a framing attack the adversary plays the role of the system where the interface represents a handful of innocent users. A framing attack is meant to capture any adversarial behavior that allows the adversary to make the open algorithm point to an innocent user. We remark that this captures the notion of exculpability as well as any other adversarial behavior that frames an innocent user. In the concurrent setting, we allow the adversary to initiate many concurrent executions of the JOIN dialog playing the role of a malicious GM. The goal of the adversary now is to produce a signature that opens to one of the innocent users.

Naturally in modeling such an attack we cannot allow to the adversary to do all the bookkeeping for the user database himself (otherwise an OPEN operation would be without meaning). Every time the adversary successfully terminates a JOIN dialog with an innocent user that is controlled by the interface $\mathcal{I}$, the interface will add the user identity into the St_{users} and will append the whole communication transcript to $St_{join-trans}$. Moreover it will keep a private database containing the secrets of the innocent users that will have the format $\langle i, \mathsf{sec}_i \rangle$ (these will not be accessible to the adversary). In addition to the above, we will allow the adversary to submit queries to a SIGN oracle that will be handled by the interface oracle $\mathcal{I}$ and accepts the identity of one of the innocent users and a message and returns a signature of this message with the signing mechanism of the named user.

We allow the adversary to have appropriately restricted modify access to the public-state St; this access will be handled by $\mathcal{I}$ in the form of the $\mathtt{MODIFY}_{St}$ oracle query. As mentioned already we will not give to the adversary full write capability to the public state St since if he is allowed to this, opening any signature correctly would be meaningless (e.g., if the adversary erases the database of JOIN transcripts it is straightforward that the opening capability is cancelled). The restrictions are as follows: $\mathtt{MODIFY}_{St}$ will not permit the adversary to insert a join transcript that reuses an identity tag (this restriction is essential to maintain the semantics of the OPEN unambiguous) and will not allow the adversary to modify any of the identity tags or join transcripts of the innocent users (to these the adversary will have read-only access). Any other modification of the public-state will be allowed by $\mathcal{I}$ (in particular the adversary is allowed to introduce users to the public-state as well as erase them — for this reason there is no need for a "corrupt" oracle).

We will use the notation $St^{\mathcal{I}}_{users}$ to denote all innocent users in the system that are introduced by the execution of the concurrent JOIN oracle and are managed by the interface oracle $\mathcal{I}$.

The Framing-Attack Game $G^{\mathcal{A}}_{\mathsf{fra}}$ (denoted by $G^{\mathcal{A}}_{\mathsf{fra}}(1^\nu)$):
1. $\langle \mathcal{Y}, \mathcal{S}\rangle \leftarrow \mathtt{SETUP}(1^\nu)$; $St = (St_{users}, St_{join-trans}) = (\epsilon, \epsilon)$;
2. $\langle m, \sigma\rangle \leftarrow \mathcal{A}^{\mathcal{I}[\mathtt{JOIN}_{\leftrightarrow User(\mathcal{Y})}, \mathtt{SIGN}, \mathtt{READ}_{St}, \mathtt{MODIFY}_{St}]}(\mathcal{Y}, \mathcal{S})$
3. $i = \mathtt{OPEN}(m, \sigma, St, \mathcal{Y}, \mathcal{S})$;
4. If $(\mathtt{VERIFY}(\mathcal{Y}, m, \sigma) = \top) \wedge (i \in St^{\mathcal{I}}_{users})$ then return $\top$ else return $\bot$;

We say that a group signature satisfies security against framing attacks with concurrent join provided that for all $\mathcal{A}$ it holds that $\mathbf{Prob}[G^{\mathcal{A}}_{\mathsf{fra}}(1^\nu) = \top] = 1 - \mathsf{negl}(\nu)$.

Anonymity. Finally, anonymity is modeled as a sort of CCA2 attack against the identity encryption embedding mechanism of the group signature.

The Anonymity-attack Game $G^{\mathcal{A}}_{\mathsf{anon}}$ (denoted by $G^{\mathcal{A}}_{\mathsf{anon}}(1^\nu)$):
1. $\langle \mathcal{Y}, \mathcal{S}\rangle \leftarrow \mathtt{SETUP}(1^\nu)$; $St = (St_{users}, St_{join-trans}) = (\epsilon, \epsilon)$;
2. $\langle aux, m, \mathsf{cert}_1, \mathsf{sec}_1, \mathsf{cert}_2, \mathsf{sec}_2, \rangle \leftarrow \mathcal{A}^{\mathcal{I}[\mathtt{JOIN}_{\leftrightarrow GM(St,\mathcal{Y},\mathcal{S})}, \mathtt{READ}_{St}, \mathtt{OPEN}]}(\mathsf{play}, \mathcal{Y})$
3. if $\neg((\mathsf{cert}_1 \leftrightharpoons_{\mathcal{Y}} \mathsf{sec}_1) \wedge (\mathsf{cert}_2 \leftrightharpoons_{\mathcal{Y}} \mathsf{sec}_2))$ or $\mathsf{cert}_1 = \mathsf{cert}_2$ then stop; return $\bot$;
4. Choose $b \leftarrow_R \{1, 2\}$;
5. $\psi \leftarrow \mathtt{SIGN}(\mathcal{Y}, \mathsf{cert}_b, \mathsf{sec}_b, m)$;
6. $b^* \leftarrow \mathcal{A}^{\mathcal{I}[\mathtt{JOIN}_{\leftrightarrow GM(St,\mathcal{Y},\mathcal{S})}, \mathtt{READ}_{St}, \mathtt{OPEN}^{\neg\psi}]}(\mathsf{guess}, aux)$;
7. if $b = b^*$ return $\top$ else return $\bot$;

We note that the $\mathtt{OPEN}^{\neg\psi}$ oracle operates as the $\mathtt{OPEN}$ oracle with the usual restriction that it should return $\bot$ if the adversary submits ψ as the signature to be opened.

A group signature is said to be secure against anonymity attacks with concurrent join provided that for all $\mathcal{A}$ it holds that $2\mathbf{Prob}[G^{\mathcal{A}}_{\mathsf{anon}}(1^\nu) = \top] - 1 = \mathsf{negl}(\nu)$.

Based on all the above we will say that a group signature with concurrent join is *secure* provided that it is secure against misidentification, framing and anonymity attacks.

4 Group Signatures with Efficient Concurrent Join : Construction

In this section we describe our efficient group signature construction. A number of primitives proved to be instrumental in our construction, namely: BB signatures [7], Linear ElGamal encryption [8], and a CCA2 variant of Pailier encryption [27, 22, 15]. We first begin by describing the public-parameters our system will employ.

Public-parameters. The public parameters of the scheme are as follows:

- p1 two groups of order p where p is a ℓ_p-bit prime, $p > 2^{\ell_p - 1}$, denoted by $\mathbb{G}_1 = \langle g_1\rangle$ and $\mathbb{G}_2 = \langle g_2\rangle$, so that there is e and $\mathbb{G}_T$ and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a bilinear map.
- p2 an RSA-modulus n, of ℓ_n bits; n is selected so that Strong-RSA will be infeasible over $QR(n)$.

p3 three integer ranges S, S', S''. We define the integer range $S =_{\mathsf{df}} S(2^{\ell-1}, 2^{\ell_p-2}) = \{2^{\ell-1}-2^{\ell_p-2}+1, \ldots, 2^{\ell-1}+2^{\ell_p-2}-1\}$. Observe that if $x, y \in S(2^{\ell-1}, 2^{\ell_p-2})$ and $x \equiv_p y$ then it holds that $x = y$; indeed, $p \mid x - y$ means that $x = y + kp$; assume without loss of generality that $k \geq 0$. Now, since $2^{\ell_p-1} < p < 2^{\ell_p}$ we have that $y \geq x + 2^{\ell_p-1}$; this is a contradiction, since even if $x = \min S = 2^{\ell-1} - 2^{\ell_p-2} + 1$ we have that $y \geq 2^{\ell-1} + 2^{\ell_p-1} - 2^{\ell_p-2} + 1 \geq 2^{\ell-1} + 2^{\ell_p-2} + 1 > \max S$. It follows that $k = 0$ and as a result $x = y$.
Now let $k, \epsilon > 1$ be parameters and select the ranges S', S'' as follows: $S' =_{\mathsf{df}} S(2^{\ell'}, 2^{\mu'}) = S(2^{\ell-1}, 2^{\lfloor(\ell_p-4)/\epsilon\rfloor-k})$ and $S'' =_{\mathsf{df}} S(2^{\ell''}, 2^{\mu''}) = S(2^{\ell'/2}, 2^{\mu'/2})$, so that $S(2^{\ell''}, 2^{\epsilon\mu''+k+2})$ does not contain an integer smaller than 2 and is disjoint from the range S. The ranges S, S', S'' are assumed subsets of $\{1, \ldots, \phi(n)\}$.

p4 a safe RSA-modulus N of ℓ_N bits with $N = PQ$ and $P = 2P'+1, Q = 2Q'+1$, so that in the group $\mathbb{Z}^*_{N^2}$ it holds that the DCR assumption is hard, and the value $G = (G_0)^{2N} (\mathrm{mod} N^2)$ is selected with $G_0 \leftarrow_R \mathbb{Z}^*_{N^2}$. Note that with overwhelming probability $\langle G \rangle$ is the subgroup of quadratic residues modulo N^2 that are simultaneously N-th residues; note that $\#\langle G \rangle = P'Q'$.

Regarding the size of parameters we observe the following: ℓ_p can be quite small, e.g., 170 bits is sufficient to achieve security that is equivalent to security of 1020 bits in multiplicative groups for the discrete-log problem (cf. also [8]). On the other hand ℓ_n, ℓ, ℓ_N will be selected so that an RSA modulus with this number of bits is hard to factor and thus $\ell_n, \ell, \ell_N \geq 1024 \gg \ell_p$. The above public parameters will be selected by the setup procedure of the group signature system as described below.

SETUP. The procedure first generates the public-parameters p1 and p2 and p3 as described above. Then, it executes the following steps:

It selects two values $\gamma, \delta \leftarrow_R \mathbb{Z}_p$ and sets $w = g_2^\gamma$ and $v = g_2^\delta$; this is the setup for BB signatures, cf. [7].

It selects two values $\alpha, \beta \leftarrow_R \mathbb{Z}_p$ and $u \leftarrow_R \mathbb{G}_1$ and sets $u' = u^{\alpha/\beta}$ and $h = u^\alpha (u')^\beta$; observe that it holds that $u^\alpha = (u')^\beta$. This is the key-setup for linear ElGamal encryption, cf. [8].

It selects $g, f_1, f_2, f_3 \leftarrow_R QR(n)$. These will be used for commitments.

(Opening functionality) the public parameters N, G according to p4 are selected as well as $H_1, H_2, H_3 \in \langle G \rangle$ with $H_i = G^{a_i}$, $a_i \leftarrow_R \mathbb{Z}_{\lfloor N/4 \rfloor}$ for $i = 1, 2, 3$ and a hash-key hk for a universal one-way hash function family UOHF. We remark that this step can be entirely separated from the GM's setup phase and executed by an opening authority. Nevertheless for convenience and simplification of the presentation we do not make further distinction in the present version of the paper.

The public-key $\mathcal{Y}$ is set to $\langle g_1, g_2, u, u', h, w, v, \mathrm{desc}(\mathbb{G}_1 || \mathbb{G}_2 || \mathbb{G}_T || e || \mathsf{UOHF}), g, f, n, N, G, H_1, H_2, H_3, \mathsf{hk} \rangle$ and the secret key $\mathcal{S}$ is set to to $\langle \gamma, \delta; a_1, a_2, a_3 \rangle$. Note that the factorization of n as well as the values α, β (the decryption key for the linear ElGamal encryption) are not needed and thus they are discarded.

`JOIN`. In the join protocol execution, the user will obtain a BB signature on an RSA modulus that he selects. A user's membership certificate is the signature itself together with the RSA modulus; a user's membership secret on the other hand is the factorization of the modulus. The join procedure between a prospective user and the GM is described in detail below:

- (User→GM) The user initiates the procedure and selects random $x \in S'$ to be an ℓ-bit RSA modulus with x_1, x_2 its two prime divisors, so that $x_1 \in S''$. The User transmits x.
- (GM→User) The GM checks whether $x \in S'$ and whether x was submitted by another user in a previous `JOIN`; if the check fails the GM terminates the `JOIN` protocol; otherwise (i) it reads the public-state St, selects $i \in \mathsf{ID}$ so that $i \notin St_{users}$ and in such a manner that i is distinct from any other concurrent executions and writes to its communication tape the values $\langle i, \sigma, r \rangle$ where $r \leftarrow_R \mathbb{Z}_p$ and $\sigma = g_1^{1/(\gamma+x+\delta r)}$; finally it updates $St_{join-trans}$ by appending to it the tuple $\langle i, \sigma, r \rangle$ and sets $St_{users} = St_{users} || \langle i \rangle$.
- The user verifies that $e(\sigma, w g_2^x v^r) = e(g_1, g_2)$ and that $i \notin St_{users}$; if either test fails the user fails the `JOIN` dialog. Otherwise, it terminates successfully by setting his membership certificate to $\mathsf{cert} = \langle x, \sigma, r \rangle$ and his membership secret to $\mathsf{sec} = \langle x_1, x_2 \rangle$.

Observe that the user *does not prove* that x was selected appropriately; Perhaps surprisingly, we show that this is still sufficient for security in the concurrent setting. Naturally if the user chooses x inappropriately two things may happen: (i) the user may not be able to issue group signatures, e.g., this may happen when x is a prime; this naturally is of no concern to the GM, (ii) the user selects x as an integer that is easy to factor; while this is of concern there is nothing that can be done about it: this case is conceptually the same as the case that the user just leaks its secret-key; while this possibility is annoying there is little that can be done to prevent this in any group signature scheme.

As a side-note the reader perhaps would be concerned with the fact that the BB-signature above (that typically operates over short messages of, say, about 170 bits) will be used to sign RSA moduli that are over 1000 bits. Our scheme prevents any kind of naive forgery based on the wrap-around by employing range proofs that ensure that the integers employed by users, while they are large they all fall within an integer range S' that contains sufficiently less than 2^{170} elements (cf. the parameter selection $\mathsf{p3}$).

`SIGN`. We present the signing algorithm. The user possesses the following: a membership certificate $\langle x, \sigma, r \rangle$ and the corresponding membership secret x_1, x_2. The signing algorithm will be obtained by applying the Fiat-Shamir Heuristic on an appropriately designed proof of knowledge. First, the signer computes the following values:

$T_1 = u^z$	$z \leftarrow_R \mathbb{Z}_p$ in $\mathbb{G}_1$
$T_2 = (u')^{z'}$	$z' \leftarrow_R \mathbb{Z}_p$ in $\mathbb{G}_1$
$T_3 = h^{z+z'}\sigma$	in $\mathbb{G}_1$
$T_4 = g^y f_1^{x_1}$	$y \leftarrow_R S(1, 2^{\ell_n - 2})$ in $QR(n)$
$T_5 = g^{y'} f_2^{x_2} f_3^t$	$y' \leftarrow_R S(1, 2^{\ell_n - 2})$ in $QR(n)$
$C_0 = G^t$	$t \leftarrow_R S(1, 2^{\ell_N - 2})$ in $\mathbb{Z}^*_{N^2}$
$C_1 = H_1^t(1+N)^x$	in $\mathbb{Z}^*_{N^2}$
$C_2 = \|\|(H_2 H_3^{\mathcal{H}(\mathsf{hk}, C_0, C_1)})^t\|\|$	in $\mathbb{Z}^*_{N^2}$

Note that $||x|| = x$ if $x \leq N^2/2$ and $||x|| = N^2 - x$ otherwise. Also recall that $S(a, b) =_{\mathsf{df}} \{a - b, \ldots, a + b\}$. Subsequently the signer will construct the signature "of knowledge" on the given message m by providing a proof of knowledge for the relations given in figure 2 that involve the fourteen witnesses $\theta_z, \theta_{z'}, \theta_x, \theta_{xz}, \theta_{xz'}, \theta_r, \theta_{rz}, \theta_{rz'}, \theta_{x_1}, \theta_{x_2}, \theta_y, \theta_{y'}, \theta_{yx_2}, \theta_t$.

$$
\begin{array}{cc}
T_1^{-1} u^{\theta_z} = 1 & T_2^{-1} (u')^{\theta_{z'}} = 1 \\
T_1^{-\theta_x} u^{\theta_{xz}} = 1 & T_2^{-\theta_x} (u')^{\theta_{xz'}} = 1 \\
T_1^{-\theta_r} u^{\theta_{rz}} = 1 & T_2^{-\theta_r} (u')^{\theta_{rz'}} = 1
\end{array}
$$

$$e(T_3, v)^{\theta_r} e(T_3, g_2)^{\theta_x} e(h, g_2)^{-\theta_{xz} - \theta_{xz'}} e(h, v)^{-\theta_{rz} - \theta_{rz'}} e(h, w)^{-\theta_z - \theta_{z'}} e(T_3, w) = e(g_1, g_2)$$

$$
\begin{array}{cc}
T_4^{-1} g^{\theta_y} f^{\theta_{x_1}} = 1 & T_4^{-\theta_{x_2}} g^{\theta_{yx_2}} f^{\theta_x} = 1
\end{array}
$$

$$T_5^{-1} g^{\theta_{y'}} f^{\theta_{x_2}} f_2^{\theta_t} = 1$$

$$C_0 = G^{\theta_t} \qquad C_1 = H_1^{\theta_t}(1+N)^{\theta_x} \qquad (C_2)^2 = (H_2 H_3^{\mathcal{H}(\mathsf{hk}, C_0, C_1)})^{2\theta_t}$$

$$\theta_x \in S' \qquad \theta_{x_1} \in S''$$

Fig. 2. The relations defining the signature of knowledge

Given the coin tosses of the signer for the selection of $T_1, T_2, T_3, T_4, T_5, C_0, C_1$, the witnesses needed in figure 2 are selected as follows: $\theta_z = z, \theta_{z'} = z', \theta_x = x, \theta_{xz} = x \cdot z (\text{mod} p)$, $\theta_{xz'} = x \cdot z' (\text{mod} p)$, $\theta_r = r$, $\theta_{rz} = r \cdot z (\text{mod} p), \theta_{rz'} = r \cdot z' (\text{mod} p), \theta_{x_1} = x_1, \theta_{x_2} = x_2, \theta_y = y, \theta_{y'} = y', \theta_{yx_2} = y \cdot x_2$ in $\mathbb{Z}$, $\theta_t = t$. Now, given a message m, the signature will be constructed as follows:

1. (choose blindings) the values $\rho_z, \rho_{z'}, \rho_{xz}, \rho_{xz'}, \rho_r, \rho_{rz}, \rho_{rz'} \leftarrow_R \mathbb{Z}_p$ and $\rho_x \leftarrow_R \pm\{0,1\}^{\epsilon\mu' + k}$, $\rho_{x_1} \leftarrow_R \pm\{0,1\}^{\epsilon\mu'' + k}$ and $\rho_{x_2}, \rho_y, \rho_{y'} \leftarrow_R \pm\{0,1\}^{\epsilon(\ell_n - 2) + k}$, $\rho_{yx_2} \leftarrow_R \pm\{0,1\}^{2\epsilon(\ell_n - 2) + k}$ and $\rho_t \leftarrow_R \pm\{0,1\}^{\epsilon(\ell_N - 2) + k}$ are selected. Using these values the following are computed:

$$
\begin{array}{cc}
R_1 = u^{\rho_z} & R_2 = (u')^{\rho_{z'}} \\
R_3 = T_1^{\rho_x} u^{-\rho_{xz}} & R_4 = T_2^{\rho_x} (u')^{-\rho xz'} \\
R_5 = T_1^{\rho_r} u^{-\rho_{rz}} & R_6 = T_2^{\rho_r} (u')^{-\rho_{rz'}}
\end{array}
$$

$$R_7 = e(T_3, v)^{\rho_r} e(T_3, g_2)^{\rho_x} e(h, g_2)^{-\rho_{xz} - \rho_{xz'}} e(h, v)^{-\rho_{rz} - \rho_{rz'}} e(h, w)^{\rho_z + \rho_{z'}}$$

$$
\begin{array}{cc}
R_8 = g^{\rho_y} f_1^{\rho_{x_1}} & R_9 = T_4^{\rho_{x_2}} g^{-\rho_{yx_2}} f^{-\rho_x}
\end{array}
$$

$$R_{10} = g^{\rho_{y'}} f_2^{\rho_{x_2}} f_3^{\rho_t}$$

$$R_{11} = G^{\rho_t} \qquad R_{12} = H_1^{\rho_t}(1+N)^{\rho_x} \qquad R_{13} = (H_2 H_3^{\mathcal{H}(\mathsf{hk}, C_0, C_1)})^{2\rho_t}$$

2. (calculate challenge) using a hash function denoted by `HASH` the value

$$c \leftarrow \mathtt{HASH}(m||T_1||\ldots||T_4||T_5||R_1||\ldots||R_{12}, R_{13})$$

is computed. The range of `HASH` is considered to be $\{0,1\}^k$.

3. (calculate response) Subsequently the following values are computed:

$s_z = \rho_z - cz$	in $\mathbb{Z}_p$	$s_{z'} = \rho_{z'} - cz'$	in $\mathbb{Z}_p$
$s_{xz} = \rho_{xz} - cxz'$	in $\mathbb{Z}_p$	$s_{xz'} = \rho_{xz'} - cxz'$	in $\mathbb{Z}_p$
$s_r = \rho_r - cr$	in $\mathbb{Z}_p$	$s_{rz'} = \rho_{rz'} - crz'$	in $\mathbb{Z}_p$
$s_{rz'} = \rho_{rz'} - crz'$	in $\mathbb{Z}_p$	$s_x = \rho_x - c(x - 2^{\ell'})$	in $\mathbb{Z}$
$s_{x_1} = \rho_{x_1} - c(x_1 - 2^{\ell''})$	in $\mathbb{Z}$	$s_{x_2} = \rho_{x_2} - c(x_2 - 1)$	in $\mathbb{Z}$
$s_y = \rho_y - c(y - 1)$	in $\mathbb{Z}$	$s_{y'} = \rho_{y'} - c(y' - 1)$	in $\mathbb{Z}$
$s_{yx_2} = \rho_{yx_2} - c(y \cdot x_2 - 2)$	in $\mathbb{Z}$	$s_t = \rho_t - c(t - 1)$	in $\mathbb{Z}$

The output of the signing algorithm is the tuple: $\langle T_1, T_2, T_3, T_4, T_5, C_0, C_1, C_2, c,$ $s_z, s_{z'}, s_{xz}, s_{xz'}, s_r, s_{rz}, s_{rz'}, s_x, s_{x_1}, s_{x_2}, s_y, s_{y'}, s_{yx_2}, s_t\rangle$.

`VERIFY`. Signature verification is achieved by the following tests:

$$s_x \stackrel{?}{\in} \pm\{0,1\}^{\epsilon\mu'+k+1} \wedge s_{x_1} \stackrel{?}{\in} \pm\{0,1\}^{\epsilon\mu''+k+1} \wedge C_0, C_1, C_2 \stackrel{?}{\in} \mathbb{Z}^*_{N^2} \wedge C_2 \stackrel{?}{\leq} N^2/2$$

$$\begin{aligned} c \stackrel{?}{=} \mathsf{HASH}\Big(& m \,||\, T_1 \,||\, T_2 \,||\, T_3 \,||\, T_4 \,||\, T_5 \,|| \\ & ||\, u^{s_z} T_1^c || (u')^{s_{z'}} T_2^c \\ & ||\, T_1^{-s_x + c2^{\ell'}} u^{s_{xz}} || T_2^{-s_x + c2^{\ell'}} (u')^{s_{xz'}} \\ & ||\, T_1^{s_r} u^{-s_{rz}} || T_2^{s_r} (u')^{-s_{rz'}} \\ & ||\, e(T_3, v)^{s_r} e(T_3, g_2)^{s_x - c2^{\ell'}} e(h, g_2)^{-s_{xz} - s_{xz'}} \\ & \quad e(h, v)^{-s_{rz} - s_{rz'}} e(h, w)^{s_z + s_{z'}} e(g_1, g_2)^c e(T_3, w)^{-c} \\ & ||\, T_4^c g^{s_y - c} f_1^{s_{x_1} - c2^{\ell''}} || T_4^{s_{x_2} - c} g^{-s_{yx_2} + 2c} f_1^{-s_x + c2^{\ell'}} || T_5^c g^{s_{y'} - c} f_2^{s_{x_2} - c} f_3^{s_t - c} \\ & || C_0^c G^{s_t - c} || C_1^c H_1^{s_t - c} (1+N)^{s_x - c2^{\ell'}} || C_2^c (H_2^2 H_3^{2\mathcal{H}(\mathsf{hk}, C_0, C_1)})^{s_t - c} \Big) \end{aligned}$$

`OPEN`. Given a signature as described above: first the signature is verified as well as the relation $(C_2)^2 = C_0^{2(a_2 + \theta\mathcal{H}(\mathsf{hk}, C_0, C_1))}$ is checked. if the test passes. If any check fails `OPEN` returns $\perp$. Otherwise, `OPEN` computes $\tilde{m} = C_1 C_0^{-a_1}$; due to the properties enforced by the proof of knowledge (cf. figure 2) it holds that $x = (\tilde{m} - 1)/N$. Then, the `OPEN` algorithm searches $St_{join-trans}$ for transcripts of the form $\langle j, x_j, \sigma_j, r_j\rangle$ with $x_j = x$ the identity j of the signer is recovered. If no such x_j is found, `OPEN` returns $\perp$.

5 Proof of Security

The proof of security is described here, it relies on the random oracle model (we prove the group signature rather than the interactive identity escrow scheme).

Theorem 1. *The signature of knowledge that specifies the* `SIGN` *algorithm satisfies: completeness, special soundness under the Strong-RSA assumption and statistical honest verifier zero-knowledge.*

Theorem 2. *Any misidentification attacker in the random oracle model against our group signature can be transformed to an adaptive chosen message attacker in the standard model against the BB signature assuming the Strong-RSA assumption.*

Theorem 3. *Any framing attacker in the random oracle model against our group signature can be transformed to a factoring algorithm in the standard model assuming the Strong-RSA assumption.*

Theorem 4. *Any anonymity adversary against our group signature in the random oracle model can be transformed to a CCA2 attacker against the public-key encryption that is employed in our scheme; this is conditional on the validity of (i) the Linear-DDH assumption, (ii) the assumption that the digital signature scheme employed (BB-signature) satisfies strong existential unforgeability. (iii) the DLOG assumption over the subgroup of $2N$-th residues inside $\mathbb{Z}^*_{N^2}$.*

The above three theorems culminate to the following theorem:

Theorem 5. *Our group signature is correct and secure in the random oracle model under the assumptions: SDH, Linear-DDH, Strong-RSA and DCR assumptions.*

References

1. G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A practical and provably secure coalition-resistant group signature scheme. In M. Bellare, editor, *Advances in Cryptology – CRYPTO ' 2000*, volume 1880 of *Lecture Notes in Computer Science*. International Association for Cryptologic Research, Springer, 2000.
2. G. Ateniese and B. de Medeiros. Efficient group signatures without trapdoors. In *ASIACRYPT: Advances in Cryptology – ASIACRYPT: International Conference on the Theory and Application of Cryptology*, Lecture Notes in Computer Science. International Association for Cryptologic Research, Springer, 2003.
3. G. Ateniese and G. Tsudik. Some open issues and new directions in group signatures. In M. Franklin, editor, *Financial cryptography: Third International Conference, FC '99, Anguilla, British West Indies, February 22–25, 1999: proceedings*, volume 1648 of *Lecture Notes in Computer Science*, pages 196–211. Springer-Verlag, 1999.

4. N. Barić and B. Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In W. Fumy, editor, *Advances in Cryptology – EUROCRYPT 1997*, volume 1233 of *Lecture Notes in Computer Science*, pages 480–494. International Association for Cryptologic Research, Springer-Verlag, Berlin Germany, 1997.
5. M. Bellare, D. Micciancio, and B. Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In E. Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, Warsaw, Poland, 2003. Springer.
6. M. Bellare, H. Shi, and C. Zhang. Foundations of group signatures: The case of dynamic groups. Cryptology ePrint Archive, Report 2004/077, 2004. `http://eprint.iacr.org/`.
7. D. Boneh and X. Boyen. Short signatures without random oracles. In C. Cachin and J. Camenisch, editors, *Advances in Cryptology – EUROCRYPT ' 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 56–73, Interlaken, Switzerland, 2004. Springer.
8. D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In M. Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 41–55. Springer, 2004.
9. J. Camenisch. Efficient and generalized group signatures. In W. Fumy, editor, *Advances in Cryptology - EUROCRYPT '97, International Conference on the Theory and Application of Cryptographic Techniques*, Lecture Notes in Computer Science, pages 465–479. International Association for Cryptologic Research, Springer, 1997.
10. J. Camenisch and J. Groth. Group signatures: Better efficiency and new theoretical aspects. In *Security in Communication Networks - SCN 2004*. Springer, 2003.
11. J. Camenisch and A. Lysyanskaya. An identity escrow scheme with appointed verifiers. In J. Kilian, editor, *Advances in Cryptology – CRYPTO ' 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 388–407. International Association for Cryptologic Research, Springer-Verlag, Berlin Germany, 2001.
12. J. Camenisch and A. Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In M. Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*. Springer, 2004.
13. J. Camenisch and M. Michels. A group signature scheme with improved efficiency. In K. Ohta and D. Pei, editors, *ASIACRYPT: Advances in Cryptology – ASIACRYPT: International Conference on the Theory and Application of Cryptology*, volume 1514 of *Lecture Notes in Computer Science*, pages 160–174. International Association for Cryptologic Research, Springer-Verlag, 1998.
14. J. Camenisch and M. Michels. Separability and efficiency for generic group signature schemes (extended abstract). In M. j. Wiener, editor, *19th International Advances in Cryptology Conference – CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 413–430. Springer, 1999.
15. J. Camenisch and V. Shoup. Practical verifiable encryption and decryption of discrete logarithms. In *CRYPTO 2003*. Springer-Verlag, 2003.
16. J. Camenisch and M. Stadler. Efficient group signature schemes for large groups. In B. S. K. Jr., editor, *Advances in Cryptology – CRYPTO ' 1997*, volume 1294 of *Lecture Notes in Computer Science*, pages 410–424. International Association for Cryptologic Research, Springer, 1997.
17. D. Chaum and E. van Heyst. Group signatures. In D. W. Davies, editor, *Advances in Cryptology, EUROCRYPT 1991 (Lecture Notes in Computer Science 547)*, pages 257–265. Springer-Verlag, April 1991. Brighton, U.K.

18. L. Chen and T. P. Pedersen. New group signature schemes (extended abstract). In A. D. Santis, editor, *Advances in Cryptology—EUROCRYPT 94*, volume 950 of *Lecture Notes in Computer Science*, pages 171–181. Springer-Verlag, 1995, 9–12 May 1994.
19. Y. Dodis, A. Kiayias, A. Nicolosi, and V. Shoup. Anonymous identification in ad hoc groups. In C. Cachin and J. Camenisch, editors, *Advances in Cryptology – EUROCRYPT ' 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 609–626, Interlaken, Switzerland, 2004. Springer.
20. P.-A. Fouque and J. Stern. One round threshold discrete-log key generation without private channels. In K. Kim, editor, *Public Key Cryptography — 4th International Workshop on Practice and Theory in Public Key Cryptosystems*, volume 1992 of *Lecture Notes in Computer Science*, pages 300–316, Cheju Island, Korea, 2001. Springer.
21. J. Garay, P. D. MacKenzie, and K. Yang. Strengthening zero-knowledge protocols using signatures. In E. Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 177–194, Warsaw, Poland, 2003. Springer.
22. R. Gennaro and Y. Lindell. A framework for password-based authenticated key exchange. In E. Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, Warsaw, Poland, 2003. Springer.
23. A. Kiayias, Y. Tsiounis, and M. Yung. Traceable signatures. In C. Cachin and J. Camenisch, editors, *Advances in Cryptology – EUROCRYPT ' 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 571–589, Interlaken, Switzerland, 2004. Springer.
24. A. Kiayias and M. Yung. Extracting group signatures from traitor tracing schemes. In E. Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 630–648, Warsaw, Poland, 2003. Springer.
25. A. Kiayias and M. Yung. Group signatures: Provable security, efficient constructions and anonymity from trapdoor-holders. Cryptology ePrint Archive, Report 2004/076, 2004. `http://eprint.iacr.org/`.
26. J. Kilian and E. Petrank. Identity escrow. In H. Krawczyk, editor, *Advances in Cryptology – CRYPTO 1998*, volume 1462 of *Lecture Notes in Computer Science*, pages 169–185. International Association for Cryptologic Research, Springer, 1998.
27. P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in cryptology—EUROCRYPT 1999*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238, 1999.
28. G. Tsudik and S. Xu. Accumulating composites and improved group signing. In *ASIACRYPT: Advances in Cryptology – ASIACRYPT: International Conference on the Theory and Application of Cryptology*, Lecture Notes in Computer Science, pages 269–286. International Association for Cryptologic Research, Springer, 2003.