Leak-free Group Signatures with Immediate Revocation

Xuhua Ding, Gene Tsudik, and Shouhuai Xu
Department of Information and Computer Science
University of California, Irvine
Email: {*xhding,gts,shxu*}*@ics.uci.edu*

Abstract

Group signatures are an interesting and appealing cryptographic construct with many promising potential applications. This paper is motivated by attractive features of group signatures, particularly, the potential to serve as foundation for anonymous credential systems. We re-examine the whole notion of group signatures from a systems *perspective and propose two new requirements:* leak-freedom *and* immediate-revocation*, which are crucial for a large class of enterprise-centric applications. We then present a new group signature scheme that achieves all identified properties. Our scheme is based on the so-called* systems architecture approach*. It is appreciably more efficient than the state-of-the-art, easy to implement and reflects the well-known* separation-of-duty *principle. Another benefit of our scheme is the obviated reliance on underlying anonymous communication channels, which has been a requirement in all previous group signature schemes.*

1. Introduction

The concept of group signatures was first introduced by Chaum and van Heyst [18] in order to facilitate group applications. Its basic property is that: given a valid (verifiable) group signature, any verifier can determine the signer's group membership, while only a designated entity (group manager) qis able to identify the actual signer. In recent years, many research efforts have been made to seek efficient constructions and precise definitions of group signatures. Early schems, (e.g., [19]) suffer from linear complexity of either (or both) group public key size or signature size with respect to the number of group members. This is overcomed by Camenisch and Stadler in [14] based on non-standard cryptographic assumptions. Follow-on results, e.g., [13] and [1], gradually improved on both efficiency and reliance on standard assumptions. Despite these advances, membership revocation remains as a hard problem. Some researchers have attempted to address it: Bresson and Stern [8], Song [25], and Ateniese, et al. [2] as well as Camenisch and Lysianskaya [10]. However, all their approaches incur a linear dependency on the number of current/revoked members.

Along with the progress of building more efficient and practical group signature schemes, its definition becomes a jumble of (perhaps redundant and/or overlapping) security requirements: *unforgeability*, *exculpability*, *traceability*, *coalition-resistance*, *no-framing*, *anonymity*, and *unlinkability* [18, 19, 14, 9, 13, 23, 3, 8, 25, 2, 10]. To untangle and simplify them, Bellare et al. [5] recently investigated "minimal" security requirements for group signature schemes. This line of research is very important, as it is the pursuit of similar requirements for *secure* public key encryption schemes [4] and *secure* key exchange protocols. In [5], Bellare, et al. took an approach similar to the formalization of *secure* encryption schemes, and showed that two security properties: *full-traceability* and *full-anonymity* are sufficient to subsume all aforementioned seven requirements.

We argue that *full-traceability* and *full-anonymity* are not sufficient for many realistic group signature settings. Our rationale is based on the observation that the group signature concept is inherently application-oriented, and thus cannot simply be treated as a primitive in the bare model.

We identify two additional requirements: leak-freedom and immediate-revocation. We believe that both are necessary for a large class of enterprise oriented applications. Informally, leak-freedom means that no signer can convince anyone (except the group manager who can identify a signer anyway) that she indeed generated the given group signature; immediate-revocation means that, once a group member is revoked, her capability of generating group signatures is disabled immediately, (ideally) without imposing extra burden on verifiers.

Why is leak-freedom **important?** Consider the following example: One of the main goals of group signatures is for an organization (commercial, government or military) to hide its internal structure. Suppose that Alice is an employee of a company (say, ABC) who is designated to sign purchase

orders and one of the suppliers is another company (say, XYZ). If, via her signature, Alice can convince XYZ that she is the signer, she could obtain kick-backs from XYZ as "gratitude" for her supplier selection. This *information leakage* illustrates potential abuse of group signatures.

We say that Alice successfully leaks a group signature if (without revealing her private key and/or any other long-term secrets) she can convince a verifier that she is the signer of a given group signature. Therefore, leak-freedom is an important property for a large class of enterprise-centric applications.

Why is immediate-revocation **important?** We continue along with the previous example: Clearly, any purchase order signed by Alice on behalf of ABC for a supplier XYZ – using any reasonable group signature scheme maintained by ABC – imposes certain financial and/or legal responsibilities on ABC. However, suppose that Alice might be aware of her impending lay-off, she can, in collusion with XYZ, deceptively report that her key is stolen while secretly sign unneeded purchase orders for ABC. This type of abuse is possible – no matter what existing group signature revocation method is used – unless we assume mandatory group signature time-stamping service or we impose a strict time limits on "depositing" all outstanding group signatures. Neither assumption is realistic. Therefore in practice the revoked member could issue group signatures using previous group setting and leave the time checking burden to the verifier. From this perspective, we consider the existing revocation schemes do not provide immediate-revocation. Also, more subtle attacks are conceivable given that underlying communication is asynchronous and that there is no easy way to differentiate between accidental and willing compromise of a private key. Therefore, the liability for such a "poisoned" group signature can not relegated to Alice; thus, the company has to bear all attendant costs and responsibilities. We remark that forward-security [6, 25] does not help here at all, since Alice could simply misbehave by keeping copies of private keys corresponding to all previous time periods. We also note that this problem is less grave in traditional public key infrastructures (PKI-s) where a "poisoned" signature cannot be attributed to anyone other than the public key certificate owner.

1.1. Our Contributions

The main contribution of this paper is twofold: First, we identify two important aforementioned properties: leak-freedom and immediate-revocation which are necessary for a large class of group signature applications. Although a similar property called *appointed verifier* has been previously explored in the context of group signatures or identity escrow [12], it is strictly weaker than the leak-freedom property we want to achieve. (See Section 4.1 for the discussions on related work including this issue.)

Second, given the combination of old and new security requirements, the obvious next step is to construct a practical scheme that achieves all of these goals. To this end, we design a new scheme which is practical and easy to implement. Specifically, our scheme needs only 11 exponentiations to generate a group signature and group signature verification is equivalent to verifying a single plain (i.e., non-group) signature, such as RSA. This is appreciably more efficient than the state-of-the-art [1, 10] which, as all other previously proposed group signature schemes, provides neither leak-freedom nor immediate-revocation. Our result has two advantages over the approaches deployed in all of the previous group signature schemes: (1) It completely releases a verifier from the burdensome obligation of "getting" the fresh state information of the system, even though which is well-defined; and (2) a revoked group member is unable to compute any "correct" group signature corresponding to previous setting. The failure of achieving this in existing schemes, under certain circumstances, would incur the requirement of "immediate deposit" of group signatures (as we have highlighted before) – another significant burden on an honest signature verifier or receiver. Another contribution of our approach is a careful examination of the corresponding trust model. It allows us to relax the requirement for the underlying anonymous communication channel, which is essential in all previous schemes.

Finally, our scheme makes use of, and reflects, the well-known security principle called *separation-of-duty* [20].

Caveat: Compared with prior **non-interactive** group signature schemes, our scheme requires light-weight interaction, which explains why it is able to satisfy all of the requirements. While interaction can be viewed as a notable drawback, we claim that it is a reasonable (even small) price to pay for additional security properties.

1.2. Outline

The rest of this paper is organized as follows: the next section provides the definition of group signatures and their security requirements. Section 3 presents the new group signature scheme. Next, Section 4 discusses some extensions and Section 5 concludes the paper.

2. Definitions and Properties of Group Signatures

In this section we present the functionality and security specifications of a group signature scheme. For the presentation of well-known properties, we follow the recent work of Bellare, et al. [5] which showed that full-traceability and

full-anonymity are sufficient for a secure group signature scheme.

A *group signature* scheme is composed of the following procedures: *1) Setup* which intializes the cryptographic setting for the whole group, e.g. parameters and group public key. *2) Join* which admits new group member; *3) Revoke* which nullifies users' group membership; *4) Sign* which, executed by group members, outputs group signatures on given messages; *5) Verify* which returns TRUE/FALSE indicating the validity of a given group signature; *6) Open*: which, executed by the group manager, identifies the actual signer of a given group signature.

We now recall the well-known properties of group signatures: correctness, full-traceability, full-anonymity, and no-misattribution. We refer the reader to [5] for a more formal treatment of the first three.

Correctness: All signatures produced by any group member using `SIGN` must be accepted by `VERIFY`.

Full-traceability: No subset of colluding group members (even consisting of the entire group, and even being in possession of the group manager's secret key for opening signatures) can create valid signatures that cannot be opened, or signatures that cannot be traced back to some member of the coalition.

Full-anonymity: It is computationally infeasible for an adversary (who is not in possession of the group manager's secret key for opening signatures) to recover the identity of the signer from a group signature, even if the adversary has access to the secret keys of **all** group members.

No-misattribution: It is computationally infeasible for a group manager to provenly attribute a group signature to a member who is not the actual signer.

Now, we provide more precise definitions of leak-freedom and immediate-revocation.

Leak-freedom: It is infeasible for a signer to convince anyone that she actually signed a message, even if the said signer is in possession of all other signers' secrets, except the secret of the group manager for opening signatures.[1]

Immediate-revocation: It is infeasible for a valid group member revoked at time t to generate a valid signature at any time $t' > t$. This addresses all potential disputes that might result from the underlying asynchronous communication channel.

1 Note that the same information could also be available to an adversary targeting full-anonymity. While full-anonymity does not guarantee anything about a signer's inability to convince anyone that she generated a given signature, leak-freedom does not necessarily imply full-anonymity either (see the remark in Section 3.5 for a concrete example).

3. Our Construction

In contrast to most prior work, our construction is designed from a *systems*, rather than purely cryptographic, perspective. (Nonetheless, cryptography still plays a major role in our construction.) The underlying idea is the introduction of an entity called a *mediation server* ($\mathcal{MS}$). $\mathcal{MS}$ is an online, partially trusted server, which helps in signature generation and membership revocation.

Roughly speaking, the system functions as follows: each time a group member needs to generate a signature, she has to somehow "identify" herself to the mediation server which then (if the member is not revoked) produces a group signature that can be verified using the group public key. As described below, the mere introduction of the mediation server does not imply that we can trivially obtain a group signature scheme possessing all the desired properties.

3.1. Further Motivation

A trivial approach that satisfies all aforementioned requirements is to make the group manager an on-line entity and have it "filter" all group signature requests. Each group member has an anonymous channel to the group manager GM and, for each message to be signed, it submits a message signed under its normal long-term signature key. GM then "translates" each incoming signature into a signature under its own well-known group signature key. The latter is then released to the requesting member and treated as a group signature. This approach is trivial, yet seemingly very effective. All security properties (including leak-freedom and immediate-revocation) are trivially satisfied and signature generation/verification costs are minimal.

There are, however, several issues with the above approach. If constant security of GM can be assured, then the trivial solution is feasible. However, having a **fully-trusted on-line** entity is typically undesirable and sometimes simply unrealistic. Moreover, such an entity would be a single point of failure in the sense of both security (i.e., compromise of GM means compromise of the whole system) and anonymity (i.e., a dishonest GM can arbitrarily "open" a group signature without being held accountable). One standard way to avoid a single point of failure is to utilize a distributed cryptosystem, which usually takes a heavy toll in system complexity, including management, computation and communication. The situation here is seemingly more complicated because we might need some advanced (and, therefore, less efficient) tools. To avoid such a single point of failure while ensuring that the resulting scheme is practical, we design a system under the guidance of the well-known separation of duty principle. (See the seminal work of Clark and Wilson [20] for necessary background.) This approach, as will be shown below, facilitates a similar flavor

of distributed security, i.e., compromise of either GM or the newly introduced mediation server $\mathcal{MS}$, but not both, does not necessarily imply complete compromise of the system.

3.2. Model

PARTICIPANTS: a set of group members $\mathbb{U}$, a group manager GM who admits group members, a mediation server $\mathcal{MS}$ who revokes group members (according to GM's instructions) and a set of signature receivers. We assume that $\mathcal{MS}$ maintains a secure dynamic database which is used to record signature transactions: once a record is stored, it cannot be deleted. To protect transaction anonymity against potential database compromise, $\mathcal{MS}$ utilizes a *decoy* technique. This can be done by having $\mathcal{MS}$ insert $(n-1)$ well-formed dummy transaction records for each genuine one. (Here n is the current number of group members.)

A more practical alternative to dummy records is for $\mathcal{MS}$ to encrypt the database with a public key of GM. Although this incurs slight additional complexity, the `OPEN` process that is only occasionally invoked remains efficient. In Section 4 we further elaborate on the issue of database secrecy. Besides this database, we assume that both $\mathcal{MS}$ and GM maintain a dynamic membership database that allows both *insert* and *delete* operations but cannot be tampered with. This is not new; similar assumptions are made in all prior group signature schemes.

COMMUNICATION CHANNELS: the communication channel between a group member $\mathcal{U} \in \mathbb{U}$ and GM is not anonymous, but authentic (which can be implemented via standard methods and is thus ignored in the rest of this paper). The GM to $\mathcal{MS}$ channel is likewise not anonymous, but authentic. In a typical system configuration, the communication channel between a group member $\mathcal{U}$ and $\mathcal{MS}$ is not anonymous. Finally, the channel between $\mathcal{MS}$ and signature receivers is not anonymous.

TRUST: precise specification of the trust model turns out to be difficult mainly because of the introduction of the new party: $\mathcal{MS}$. Nevertheless, In the light of the well-known *separation-of-duty* principle, we have:

1. The group manager is trusted not to introduce any illegal (or phantom) group members. However, GM may want to frame an honest group member.
2. $\mathcal{MS}$ is trusted to enforce GM's policy, e.g., to stop services for the revoked group members as requested by GM and to produce group signatures only for legitimate members. $\mathcal{MS}$ is also assumed not to misbehave if such an activity will be held accountable; for example, in the suggested system configuration where $\mathcal{MS}$ delivers group signatures, $\mathcal{MS}$ will incur appropriate delay for blocking trivial traffic analysis attack because a misbehavior here is easily held accountable. Nonetheless, $\mathcal{MS}$ may want to: 1) frame an honest member into signing a message, 2) generate a group signature without being caught, and 3) compromise anonymity of an honest group member (e.g., via an out-of-band channel).

SECURITY DEFINITIONS: due to the introduction of the mediation server $\mathcal{MS}$, we need to slightly re-tool some of the security definitions in Section 2 for our specific setting: full-traceability, full-anonymity, and leak-freedom. The changes are minimal but necessary for the sake of clarity. (The rest of the definitions remains unchanged.)

Definition 3.1
- Full-traceability*: the only change is that the set of colluders is allowed to include $\mathcal{MS}$.*
- Full-anonymity*: similarly, the only change is that,* **in addition**, *we allow the adversary to have access to the secret key(s) of $\mathcal{MS}$.*
- Leak-freedom*: it is infeasible for a signer to convince anyone (except $\mathcal{MS}$) she resulted in a group signature; it is infeasible for $\mathcal{MS}$ to convince any other party that certain group member resulted in a group signature.*

REMARK 1. In the above definition, we implicitly assumed that, if an adversary compromised a $\mathcal{MS}$, then the adversary is allowed to have partial control over $\mathcal{MS}$ (i.e., the adversary knows $\mathcal{MS}$'s key for one functionality but not for the other). This assumption is not uncommon at all.

2. The fact that GM is able to identify the actual signer of any given group signature is not treated as a leakage, because GM must have this capability anyway.

3.3. Accountable Designated Verifier Signature Schemes

We introduce the notion of accountable designated verifier signatures as the building block of our new group signature scheme. This notion is an enhancement of the private contract signature scheme in [21]. Informally, a private contract signature is a designated verifier signature that can be converted into universally-verifiable signature by either the signing party or a trusted third party appointed by the signing party, whose identity and power to convert can be verified (without interaction) by the party who is the designated verifier. An accountable designated verifier signature scheme, on the other hand, emphasizes on the trusted third party's capability of identifying an actual signer of a valid signature.

Definition 3.2 *Suppose that P_i and P_j are two participants where $i \neq j$, and that T is a trusted third party. An accountable designated verifier signature scheme,* `ADVS`*, is a tuple of polynomial-time algorithms (*`ADVS-Sign`, `ADVS-Ver`, `ADVS-Proof`*) defined as follows.*

1. ADVS-Sign, *executed by* P_i *on message* m *for* P_j *with respect to* T*, outputs an accountable designated verifier signature* $\delta = \texttt{ADVS-Sign}_{P_i}(m, P_j, T)$.
2. ADVS-Ver *allows* P_j *to verify the validity of an input tag* δ *so that*

$$\texttt{ADVS-Ver}(m, P_i, P_j, T; \delta) = \begin{cases} \text{TRUE} & if\ \delta = \texttt{ADVS-Sign}_{P_i}(m, P_j, T) \\ \text{FALSE} & otherwise \end{cases}$$

3. ADVS-Proof*, which is executed by* T *on input* P_i*,* m*,* P_j*, and a tag* δ*, produces a proof for the predicate* SignedBy(δ, P_i) *which is* TRUE *if* δ *is produced by* P_i*, and* FALSE *otherwise.*

We require that if $\delta = \texttt{ADVS-Sign}_{P_i}(m, P_j, T)$*, then we always have* $\texttt{ADVS-Ver}(m, P_i, P_j, T; \delta) =$ TRUE *and* T *always outputs a proof for* SignedBy(δ, P_i) *being* TRUE*.*

Definition 3.3 *An accountable designated verifier signature scheme is* secure *if the following are satisfied:*

1. Unforgeability *of* $\texttt{ADVS-Sign}_{P_i}(m, P_j, T)$*: For any* m*, it is infeasible for anyone not belonging to* $\{P_i, P_j\}$ *to produce* δ *such that* $\texttt{ADVS-Ver}(m, P_i, P_j, T; \delta) =$ TRUE*.*
2. Non-transferability *of* $\texttt{ADVS-Sign}_{P_i}(m, P_j, T)$*: For* P_j *there is a polynomial-time forgery algorithm which, for any* m*,* P_i*, and* T*, outputs* δ *such that* $\texttt{ADVS-Ver}(m, P_i, P_j, T; \delta) =$ TRUE*.*
3. Unforgeability *of the proof for* SignedBy(δ, P_i)*: For any* $\delta = \texttt{ADVS-Sign}_{P_i}(m, P_j, T)$*, it is infeasible for anyone not belonging to* $\{T, P_i\}$ *to produce a proof for* SignedBy(δ, P_i)*.*

Remark. We stress that the above definition does not capture whether P_i should be able to produce a proof for SignedBy(δ, P_i). This capability is necessary in the context of private contract signatures, but undesirable in our application contexts, because the former *only* intends to prevent P_j from being able to transfer the bit information SignedBy(δ, P_i) by whatever means, whereas the latter intends to prevent both P_i *and* P_j from leaking information. Thus, we have the following definition.

Definition 3.4 *An accountable designated verifier signature scheme is* strong-secure *if it, in addition to being* secure*, ensures that a signing party* P_i *cannot produce a proof for* SignedBy(δ, P_i) *with non-negligible probability, where* $\delta = \texttt{ADVS-Sign}_{P_i}(m, P_j, T)$*.*

Ideally we need a *strong-secure* ADVS scheme because such a scheme allows us to construct a simpler leak-free group signature system. Unfortunately, we do not know how to construct such a scheme and leave it as an interesting open problem. In order to facilitate a group signature scheme that is leak-free with immediate-revocation, we utilize a *secure* ADVS scheme that is based on the ideas in [21]. The details are available in our full version paper.

3.4. Leak-free Group Signature Scheme with Immediate-Revocation

We are finally ready to present a concrete construction. The basic operation of our scheme is as follows. For message m, a group member $\mathcal{U}_i$ presents an accountable designated verifier signature $\delta = \texttt{ADVS-Sign}_{\mathcal{U}_i}(m, \mathcal{MS}, \mathcal{GM})$ to the mediation server $\mathcal{MS}$ thereby requesting a plain signature $\sigma = \texttt{Sign}_{\mathcal{MS}}(m)$. The latter is viewed as a group signature, since there is a single group-wide verification key. Note that, since GM plays the role of a trusted third party in the ADVS scheme, it can hold the actual signer accountable. Following the definition of group signatures, our mediated group signature scheme is composed of the procedures below:

SETUP. This consists of initializing a group manager GM and a mediation server $\mathcal{MS}$.

1. The initialization of the system setting includes the following: It chooses a system wide security parameter κ, based on which it chooses a discrete-logarithm based *crypto-context* as specified in normal Schnorr setting.The parameter κ and the *crypto-context* are thus followed system-wide by the group members, the mediation server $\mathcal{MS}$, and the group manager GM itself. It also specifies an accountable designated verifier signature scheme ADVS as well as GM's public/private key pair $\langle Y_{\mathcal{GM}} = g^{X_{\mathcal{GM}}}, X_{\mathcal{GM}} \rangle$. It then sets up a database DBUser-GM to keep record of all users status.
2. The initialization of the mediation server $\mathcal{MS}$ consists of the following: It chooses a pair of public and private group membership keys $\langle Y_{\mathcal{MS}} = g^{X_{\mathcal{MS}}}, X_{\mathcal{MS}} \rangle$.

   It chooses a pair of keys for a normal digital signature scheme $\mathcal{SIG} = (\texttt{Gen}, \texttt{Sign}_{\mathcal{MS}}, \texttt{Ver}_{\mathcal{MS}})$ that is secure against adaptive chosen-message attack. Denote by $\langle pk_{\mathcal{MS}}, sk_{\mathcal{MS}} \rangle$ the pair of group signature verification and generation keys, where $pk_{\mathcal{MS}}$ is publicly known. We remark that any secure signature scheme can be used as $\mathcal{SIG}$. We assume that $\mathcal{MS}$ knows $sk_{\mathcal{MS}}$ in its entirety; this is to prevent attacks from happening because of an inappropriate system initialization, and can be ensured by utilizing techniques due to [26].
3. It initializes a database DBMember-MS, and a database DBSig-MS of entry structure (group-member-id, ADVS-signature, normal-signature). The latter one is to keep a log of all issued group signatures.

JOIN. Whenever the group manager GM decides to admit a new group member, it assigns a unique identity $\mathcal{U}_i$ to the user. $\mathcal{U}_i$ generates her public/private key pair with respect to the ADVS scheme: $\langle Y_{\mathcal{U}_i} = g^{X_{\mathcal{U}_i}}, X_{\mathcal{U}_i} \rangle$. $Y_{\mathcal{U}_i}$ is then registered in GM and $\mathcal{MS}$, and the related databases are updated accordingly.

REVOKE. Whenever the group manager GM decides to revoke the membership of a group member $\mathcal{U}_i$, GM updates the database DBUser-GM indicating $\mathcal{U}_i$ is revoked and logs necessary information. GM then informs $\mathcal{MS}$ that $\mathcal{U}_i$ has been revoked. $\mathcal{MS}$ simply deletes the entry $(\mathcal{U}_i, Y_{\mathcal{U}_i})$ from its database DBMember-MS. All subsequent signature requests from $\mathcal{U}_i$ will be rejected by $\mathcal{MS}$.

SIGN. Whenever a group member $\mathcal{U}_i$ wants to generate a group signature on a message m, the following protocol is executed.

1. $\mathcal{U}_i$ sends to $\mathcal{MS}$ an accountable designated verifier signature $\delta = \texttt{ADVS-Sign}_{\mathcal{U}_i}(m, \mathcal{MS}, \mathcal{GM})$ over a public and unauthenticated channel.
2. On receipt, $\mathcal{MS}$ retrieves $\mathcal{U}_i$'s public key $Y_{\mathcal{U}_i}$ from its database DBMember-MS. If no entry is found, $\mathcal{MS}$ simply ignores the request. Next, $\mathcal{MS}$ verifies δ by checking whether $\texttt{ADVS-Ver}(m, \mathcal{MS}, \mathcal{GM}; \delta) =$ TRUE. $\mathcal{MS}$ then produces a normal signature $\gamma = \texttt{Sign}_{\mathcal{MS}}(m)$ and inserts a new record: $(\mathcal{U}_i, \delta, \gamma)$ into its database DBSig-MS. The signature γ will be treated as a group signature on message m. How should the group signature γ be sent to the potential verifier(s) depends on the local policy. One option that allows us to completely get rid of all anonymous channels is to let $\mathcal{MS}$ send γ to the receiver.[2] Another option, which is not so elegant, is to let $\mathcal{MS}$ broadcast γ so that $\mathcal{U}_i$ can get and resend γ to the receiver via an anonymous channel.

VERIFY. Given $pk_{\mathcal{MS}}$, the public group signature verification key of $\mathcal{MS}$, and a tag γ, anyone can verify that γ is a valid (group) signature by running $\texttt{Ver}_{\mathcal{MS}}$ on inputs: $pk_{\mathcal{MS}}$, m, and γ.

OPEN. Whenever GM decides to identify the actual signer of signature γ on message m (i.e., the group member that requested γ from $\mathcal{MS}$), the following protocol is executed by the group manager GM and the mediation server $\mathcal{MS}$:

1. GM sends γ to $\mathcal{MS}$ via an authenticated communication channel.
2. Given γ, $\mathcal{MS}$ retrieves from its databases $(\mathcal{U}_i, \delta, \gamma)$, which it sends to GM via the same authenticated channel.
3. GM checks whether $\texttt{ADVS-Ver}(m, \mathcal{MS}, \mathcal{GM}; \delta) =$ TRUE; which always holds because of our trust model. Then, GM executes ADVS-Proof to produce a proof for SignedBy$(\delta, \mathcal{U}_i)$. If SignedBy$(\delta, \mathcal{U}_i)$ is TRUE, $\mathcal{U}_i$ is the signer; otherwise $\mathcal{MS}$ takes the responsibility.

3.5. Security Analysis

Security of our construction is stated in the following theorem:

Theorem 3.1 *Our scheme satisfies the requirements specified in Definition 3.1:* correctness, full-traceability, full-anonymity, no-misattribution, leak-freedom, *and* immediate-revocation.

Due to space limitations, we only present below a summary of informal security arguments. A full formal proof will appear in a longer technical report version of this paper.

4. Extension and Discussion

Enhancing anonymity against traffic analysis. Our scheme does not assume that the channel between a group member $\mathcal{U}$ and the mediation server $\mathcal{MS}$ is authenticated, nor is it assumed that there is an anonymous channel. This gain comes from the general assumption that the $\mathcal{MS}$ has some potential incentives to cheat an outsider, which, in turn, implies:

- Even if an adversary can eavesdrop on all channels, there could still be an out-of-band channel between a group member and $\mathcal{MS}$. Thus, the adversary could still be fooled.
- $\mathcal{MS}$ can easily cheat an outsider by injecting fake ADVSs into the network or fake entries into its database.

However, an adversary might know that $\mathcal{MS}$, while not being trusted to preserve anonymity, does not always inject fake traffic into the network? Then, an adversary still has a good chance of compromising anonymity of some honest group members by simply conducting a traffic analysis attack. Fortunately, this can be easily resolved by using link encryption and traffic padding.

On strong-secure ADVS vs. secure ADVS. In our construction we utilized an ADVS that is *secure*, but not *strong-secure*. Consequently, we assume the secrecy of storage at $\mathcal{MS}$, particularly of the database entries: $(\mathcal{U}_i, \delta = \texttt{ADVS-SIG}_{\mathcal{U}_i}(m, \mathcal{MS}, \mathcal{GM}), \texttt{Sign}_{\mathcal{MS}}(m))$. This is necessary in avoiding the following attack: If an attacker has access to such an entry in the database of $\mathcal{MS}$, then Alice can easily convince any party such as XYZ that she resulted in $\texttt{Sign}_{\mathcal{MS}}(m)$. Clearly, if a *strong-secure*

2 In this case, there may be a need for a random delay to defeat traffic analysis, however, such a delay already exists in current anonymous channels.

ADVS is utilized, then we can achieve strictly stronger security that (for instance) Alice is still unable to convince XYZ that she resulted in a signature, even if she has access to the corresponding entry in the database of $\mathcal{MS}$. We remark, however, that $\mathcal{MS}$ is always unable to convince XYZ that Alice resulted in an entry $(\text{Alice}, \delta = \texttt{ADVS-SIG}_{\text{Alice}}(m, \mathcal{MS}, \mathcal{GM}), \texttt{Sign}_{\mathcal{MS}}(m))$.

Robustness against denial-of-service (DoS) attacks. Recall than an $\mathcal{MS}$ always performs some non-negligible computation (which includes modular exponentiations) before it can determine whether an incoming signature is valid. This opens the door for DoS attacks aiming to render $\mathcal{MS}$ incapable of providing service. To counter such attacks, we propose a simple and intuitive solution: require each $\mathcal{U}_i$ and $\mathcal{MS}$ to share a unique secret key w_i. Each signature request from $\mathcal{U}_i$ must also be accompanied by an authentication token (e.g., a message authentication code or MAC) computed over the request with the key w_i. When processing a request, $\mathcal{MS}$ first verifies the authentication token before performing a much more expensive validation of the ADVS signature. (Verifying a symmetric MAC or HMAC is several orders of magnitude cheaper than verifying a public key signature.) Note that the introduction of the authentication token does not jeopardize the properties of our scheme, since w_i is known to both $\mathcal{U}_i$ and $\mathcal{MS}$. (Clearly, no group signature scheme can be based on common secrets; it is only used to protect against DoS attacks.)

The adversary may still mount a DoS attack on an $\mathcal{MS}$'s network interface. If $\mathcal{MS}$' becomes unreachable, members can no longer generate group signatures. One simple countermeasure is to duplicate the signature key among a set of $\mathcal{MS}$-s. Nevertheless, such a strategy would incur some other issues that need to be dealt with. Furthermore, We observe that the group manager could *possibly* detect compromise of the normal signing key, provided that the adversary cannot maintain the consistence between a fake, yet valid, normal signature and its corresponding entry in the database. Due to space limitation, we will analyze in detail those relevant issues in the extended version of this paper.

4.1. Related Work

This paper can be viewed as one among many efforts pursuing practical and secure *group signature* or *identity escrow* schemes [18, 19, 14, 23, 1, 12], as well as anonymous *credential systems* [16, 17, 24, 11, 15]. Among them, the prior work most relevant to this paper is [12], which presented an identity escrow scheme (and a corresponding group signature scheme) with the appointed verifier property. Their motivation was to obtain a scheme where a group member can only convince one or more appointed verifiers of her membership, while no other party can verify membership even if the signer cooperates fully. (As long as she does not give away her long-term secrets).

Clearly, there is a difference between the appointed verifier property in [12] and the leak-freedom property specified in this paper. Specifically, the [12] scheme, by definition, allows a signer to convince designated verifiers that she is authorized to conduct relevant transactions. Cast in the previous example, Alice can always convince XYZ that she is authorized to sign purchase orders. However, this exact capability can result in the leakage (outlined in Section 1) that we want to avoid!

Besides achieving the strictly stronger leak-freedom, our scheme is more efficient than [12] which requires both a signer and a verifier to compute more than $17k$ exponentiations, where k is a security parameter (say, $k = 80$). Moreover, membership revocation is not supported in [12], whereas, we achieve immediate-revocation which has only been explored in the context of traditional PKI-s [7].

A credential system is a system where users can obtain credentials from organizations and demonstrate possession of these credentials [16, 17, 24, 11, 15]. Chaum and Evertse [17] presented a general scheme using a semi-trusted TTP common to multiple organizations. However, their scheme is impractical. The credential system by Lysianskaya, et al. [24] captures many of the desirable properties. Camenisch and Lysianskaya [11] presented a better solution with ingredients from a secure group signature scheme of [1]. The prototype implementation of [11] was done by Camenisch and van Herreweghen [15]. This scheme requires both signers and verifiers to compute 22 modular exponentiations. Their advanced scheme which provides all-or-nothing non-transferability (to discourage a signer from sharing her credentials with other parties) requires both signer and verifier to compute 200 exponentiations.

The notion called *abuse-freedom* that has been previously investigated in the context of contract signing [21], is weaker than leak-freedom because the former intends only to prevent the *designated verifier* from being able to transfer the information about the actual signer, whereas the latter intends to prevent a *signer* as well as the *designated verifier* from being able to transfer the same information.[3] Moreover, leak-freedom is similar to receipt-freedom property that has been investigated in the context of voting schemes [22]. The main difference is that the former disallows a signer to convince a signature receiver for whom a signature is targeted, whereas the latter has no such targeted signature receiver.

3 Although we achieve leak-freedom using a *system architecture* approach instead of a *pure* cryptographic approach, the latter is left an interesting open question; see Section 5 below.

5. Conclusion

We identified two crucial aforementioned properties: leak-freedom and immediate-revocation which are necessary for a large class of group signature applications. We also constructed a practical scheme that achieves all of traditional and newly-introduced goals by following a *system architectural approach*, which is realistic since the resultant scheme is practical and easy to implement. Specifically, our scheme needs only 11 exponentiations for a group member to generate a group signature and one normal signature verification, such as RSA, for its validation. Another contribution of our approach is a careful examination of the corresponding trust model that we relax the requirement for the underlying anonymous communication channel, which is essential in all previous schemes.

There are several interesting open problems for future investigations:

- How to construct a practical *strong-secure* accountable designated verifier signature scheme?
- How to construct a leak-free group signature scheme with immediate-revocation without relying on a mediation server? Although we believe that the existence of a mediation server is more realistic than the existence of (for instance) a time-stamping service, it is nevertheless conceivable that other alternatively constructions could fit well into different specific application scenarios.
- How to achieve a stateless $\mathcal{MS}$? This is not trivial because the binding of an ADVS and a normal signature will allow $\mathcal{MS}$ to convince an outsider of the identity of the actual signer.

References

[1] G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A practical and provably secure coalition-resistant group signature scheme. In *CRYPTO'2000*, pages 255–270.

[2] G. Ateniese, D. Song, and G. Tsudik. Quasi-efficient revocation of group signatures. In *Financial Crypto'02)*.

[3] G. Ateniese and G. Tsudik. Some open issues and new directions in group signatures. In *Financial Crypto'99)*.

[4] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among notions of security for public-key encryption schemes.

[5] M. Bellare, D. Micciancio, and B. Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In *EuroCrypto'2003*.

[6] M. Bellare and S. Miner. A forward-secure digital signature scheme. In *CRYPTO'99*, pages 431–448.

[7] D. Boneh, X. Ding, G. Tsudik, and C. Wong. A method for fast revocation of public key certificates and security capabilities. In *10th USENIX Security Symposium*, 2001.

[8] E. Bresson and J. Stern. Group signatures with efficient revocation. In *International Workshop on Practice and Theory in Public Key Cryptography (PKC) '2001*.

[9] J. Camenisch. *Group Signature Schemes and Payment Systems Based on the Discrete Logarithm Problem*. PhD thesis, ETH Zurich, 1998.

[10] J. Camenisch and A. Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In *CRYPTO'2002*, pages 61–76.

[11] J. Camenisch and A. Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *EuroCrypto'2001*, pages 93–118.

[12] J. Camenisch and A. Lysyanskaya. An identity escrow scheme with appointed verifiers. In *CRYPTO'2001*, pages 388–407.

[13] J. Camenisch and M. Michels. A group signature scheme with improved efficiency (extended abstract). In *Advances in Cryptology – ASIACRYPT '98*.

[14] J. Camenisch and M. Stadler. Efficient group signature schemes for large groups (extended abstract). In *CRYPTO'97*, pages 410–424.

[15] J. Camenisch and E. van Herreweghen. Design and implementation of the *idemix* anonymous credential system. In *ACM CCS*, 2002.

[16] D. Chaum. Security without identification: transaction systems to make big brother obsolete. *Communications of the ACM*, 28(10):1030–1044, Oct. 1985.

[17] D. Chaum and J. Evertse. A secure and privacy-protecting protocol for transmitting personal information between organizations. In *CRYPTO'86*, pages 118–167.

[18] D. Chaum and E. van Heyst. Group signatures. In *EuroCrypto'91*, pages 257–265.

[19] L. Chen and T. Pedersen. New group signature schemes. In *EuroCrypto'94*.

[20] D. Clark and D. Wilson. A comparison of commercial and military computer security policies. In *Proceedings of the IEEE Symposium on Research in Security and Privacy '87*.

[21] J. Garay, M. Jakobsson, and P. MacKenzie. Abuse-free optimistic contract signing. In *CRYPTO'99*, pages 449–466.

[22] M. Hirt and K. Sako. Efficient receipt-free voting based on homomorphic encryption. In *EuroCrypto'2000*, pages 539–556.

[23] J. Kilian and E. Petrank. Identity escrow. In *CRYPTO'98*, pages 169–185.

[24] A. Lysyanskaya, R. Rivest, A. Sahai, and S. Wolf. Pseudonym systems. In *6th Annual Workshop on Selected Areas in Cryptography (SAC '99)*.

[25] D. Song. Practical forward-secure group signature schemes. In *ACM CCS*, 2001.

[26] S. Xu and M. Yung. The dark side of threshold cryptography. In *Financial Crypto (FC '02)*.