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. The popularity of group signatures is evident from many recent research results that investigated precise definitions and realizations of group signature schemes. This paper is motivated by attractive features of group signatures, particularly, the potential to serve as foundation for anonymous credential systems. With that in mind, we re-examine the whole notion of group signatures from a *systems* perspective. Somewhat surprisingly, we discover that there are two new and hereto un-addressed requirements:** leak-freedom **and** immediate-revocation**, that are crucial for a large class of enterprise-centric applications. We then propose 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.**

I. Introduction

The concept of group signatures was introduced by Chaum and van Heyst [CvH91] and has been claimed to have many applications [C98], [CL01]. In the past decade, many research efforts and results focused on seeking precise definitions and efficient constructions of group signature schemes, or their interactive dual, *identity escrow* [KP98]. Indeed, group signatures and derivatives has been one of the most active research topics in recent years [BMW03], [KY03]. Intuitively, a group signature can be seen as a normal digital signature with some extra properties:

> Any verifier can establish that a valid (verifiable) group signature was generated by a legitimate group member, while the actual signer can only be identified by a designated entity, called a *group manager*. In addition, group signatures are unlinkable and neither a group member nor even a group manager can mis-attribute a valid group signature.

Group signatures are a very appealing concept with many nice features. In particular, they can be used as foundation for anonymous credential systems in various application contexts. With that in mind, our goal is to re-examine the notion of group signatures from a *systems* perspective. Somewhat surprisingly, we identify two additional requirements for group signatures which are important for a large class of enterprise-oriented applications. However, they have not been addressed in previously proposed schemes. To this end, we construct a *practical* (rather than "efficient"[1]) group signature scheme that captures these new (as well as other previously identified) requirements.

A. Background

The original and chief motivation for group signatures is to facilitate the following functionality in a context of a group [CvH91]: (1) only a group member can sign on behalf of the group; (2) anyone can verify a group signature, (3) no one (except a group manager) can discover the signer's identity or link/un-link multiple signatures; (4) no one can mis-attribute a valid group signature; and (5) if necessary, a group signature can be "opened" by the group manager in order to identify the actual signer.

Recently, there have been many research efforts to construct practical group signature schemes. Early schemes (e.g., [CP94]) have the drawback of either (or both) group public key size or group signature size being *linearly* dependent on the number of group members. Consequently, the complexity of generating and verifying signatures is linear in the number of current members. Such schemes are clearly unsuitable for large groups. Nonetheless, the early schemes offer some advantages: (1) some of the schemes are proven secure using some standard cryptographic assumptions, and (2) they can easily support dynamic membership since excluding (or adding) a member can be achieved by the group manager manipulating the group public key.

In order to avoid aforementioned *linear* complexity, Camenisch and Stadler [CS97] constructed a scheme where both the group public key and a group signature are of constant size. (However, this was achieved at the cost of expensive signature operations and non-standard assumptions.) Follow-on results, e.g., [CM98] and [ACJT00], gradually improved on both efficiency and reliance on standard cryptographic assumptions. Despite these advances, membership revocation has turned out to be a difficult problem. Some attempts have been made to support revocation in group signature schemes: Bresson and Stern [BS01], Song [S01], and Ateniese, et al. [AST02] as well as Camenisch and Lysianskaya [CL02].

Unfortunately, even in these schemes, revocation incurs a linear dependency on either the number of current, or the

[1]The term "efficient" is often used in a purely theoretical sense in cryptographic literature.

total number of revoked, members. This is because of: (1) group manager re-issuing all certificates for each revocation interval; (2) group member proving, as part of signing, that its certificate is not revoked; or (3) verifier checking each group signature against the current list of revoked certificates. The state-of-the-art is a scheme obtained by integrating the dynamic accumulator construct of Camenisch and Lysianskaya [CL02] (which allows for an efficient proof that a group member is not revoked) and the "bare" group signature scheme of [ACJT00] (which allows efficient proof of knowledge of a secret key corresponding to a valid certificate). However, even in [CL02], revocation is explicit and requires all parties to be aware of most recent accumulator parameters which can be viewed as part of the group public key. Such an approach is neither feasible nor efficient for a large distributed system. Moreover, the revoked group member is still be able to compute a "correct" group signature corresponding to the previous group setting. Consequently, besides obtaining the latest group certificate, the verifier is obliged to determine the signature generation time. These two burdens are widely recognized as expensive and cumbersome.

In the past, group signature schemes were striving to satisfy a jumble of (perhaps redundant and/or overlapping) security requirements: *unforgeability*, *exculpability*, *traceability*, *coalition-resistance*, *no-framing*, *anonymity*, and *unlinkability* [CvH91], [CP94], [CS97], [C98], [CM98], [KP98], [AT99], [BS01], [S01], [AST02], [CL02]. To untangle and simplify these somewhat messy requirements, Bellare et al. [BMW03] recently investigated "minimal" security requirements for group signature schemes. This line of research is very important, as is the pursuit of similar requirements for *secure* public key encryption schemes [GM84], [NY90], [RS91], [DDN91] and *secure* key exchange protocols. In [BMW03], 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 of the above (seven) requirements.

B. New Requirements

In this section we argue that, in addition to *full-traceability* and *full-anonymity*, two other requirements are needed 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 a given group signature. Also informally, 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 most often cited uses 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's private key is lost or purposely revealed. Alternatively, Alice might be aware of her impending lay-off or termination at ABC. In any case, Alice (or whoever has her private key), in collusion with the crooked supplier XYZ, has the incentive to sign unneeded purchase orders for ABC.

This type of abuse is possible – no matter what existing group signature revocation methods 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 [A97], [BM99], [S01] 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.

C. 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 [CL01a], it is strictly weaker than the leak-freedom property we want to achieve. (See Section IV-A 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 [ACJT00], [CL02] 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* [CW87].

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.[2] 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. Such trade-offs (additional communication or interaction for stronger security guarantees) have been explored in different contexts, e.g., [GHY85], [K03].

D. Outline

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

II. 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. [BMW03] which showed that full-traceability and full-anonymity are sufficient for a secure group signature scheme.

Definition 2.1: A **group signature scheme** is a digital signature scheme comprised of the following five procedures:

- SETUP: A probabilistic algorithm that, on input of a security parameter k, outputs the group public key $pk_{\mathcal{GM}}$ (including all system parameters) and the group manager's secret key $sk_{\mathcal{GM}}$.
- JOIN: A protocol between the group manager GM and a user results in the user becoming a group member $\mathcal{U}$. Their common output contains the user's unique membership public key $pk_{\mathcal{U}}$, and perhaps some updated information that indicates the current state of the system. The user's output includes a membership secret key $sk_{\mathcal{U}}$.
- REVOKE: An algorithm which, on input of identity of a group member (and perhaps her public key $pk_{\mathcal{U}}$), outputs some updated information that indicates the current state of the system after revoking the membership of this group member.
- UPDATE: An algorithm that may be triggered by a JOIN or REVOKE. This algorithm may be run by the group members after obtaining certain information from the group manager GM.
- SIGN: A probabilistic algorithm that, on input of a group public key $pk_{\mathcal{GM}}$, a user's membership secret/public key-pair $(sk_{\mathcal{U}}, pk_{\mathcal{U}})$, and a message m, outputs a group signature δ of m.
- VERIFY: A public algorithm that, on input of a group public key $pk_{\mathcal{GM}}$, a group signature δ and a message m, outputs a binary value *TRUE/FALSE* indicating whether δ is a valid group signature (under $pk_{\mathcal{GM}}$) of m.
- OPEN: An algorithm executed by the group manager GM that takes as input of a message m, a group signature δ, the group public key pk_{GM} and the group manager's secret key $sk_{\mathcal{GM}}$. It first executes VERIFY on the first three inputs and, if the δ is valid, outputs some incontestable evidence (e.g., a membership public key $pk_{\mathcal{U}}$ and a proof) that allows anyone to identify the actual signer.

We now summarize the well-known properties of group signatures: correctness, full-traceability, full-anonymity, and no-misattribution. We refer the reader to [BMW03] 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. We remark that allowing the adversary to know the group manager's secret key for opening signatures is needed to show the strength of a group signature scheme, and not for accommodating corruption of the group manager.

More formally, we consider an adversary $\mathcal{A}$ that runs in two stages, a **choose** stage and a **forge** stage. On input the group public key and the secret of the group manager, the adversary starts its attack by adaptively corrupting a set $\mathbb{C} \subseteq \mathbb{U}$. At the end of the **choose** stage, the set $\mathbb{C}$ contains the identities of the corrupted members. In the **forge** stage, the adversary attempts to produce a forgery (m, δ). We say the adversary wins if δ is a valid group signature on m, but the OPEN algorithm returns either 1) $\perp$ or 2) a valid user identity $i \in \mathbb{U} - \mathbb{C}$. We require the probability that the adversary wins to be negligible.

[2]Despite the interaction, ours is not *identity escrow* but a true *group signature* scheme.

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.

To capture this, an indistinguishability requirement is imposed, where the adversary is given the secret keys of all group members. The adversary also has access to the *opening oracle* which, when queried with a message m and signature δ, returns the identity of the signer. More formally, the adversary acts in two stages: a **choose** stage and a **guess** stage. In the **choose** stage, $\mathcal{A}$ takes as input the group members' secret keys as well as the group public key. Then, $\mathcal{A}$ queries the *opening oracle* on group signatures of his choice. At the end of the **choose** stage, $\mathcal{A}$ outputs two valid group member identities $\mathcal{U}_i$ and $\mathcal{U}_j$, and a message m. The adversary also outputs some state information to be used in the **guess** stage. In the **guess** stage, $\mathcal{A}$ is given the state information, and a signature on m produced using the secret key of $\mathcal{U}_i$ or $\mathcal{U}_j$, chosen at random. The goal is to guess which of the two secret keys was used. We say that a group signature scheme is fully-anonymous if, for any polynomial-time adversary $\mathcal{A}$, the advantage of $\mathcal{A}$ successfully telling the identity of the actual signer is negligible. We remark that it is sufficient to allow the adversary to issue a request to sign its message m under one of the two chosen identities $\mathcal{U}_i$ and $\mathcal{U}_j$.

No-misattribution: This property has been implicitly considered in prior group signature schemes (e.g., [ACJT00]), where the group manager typically publishes a proof that it "correctly" attributed a signature to the actual signer. Note that this property is not implied by full-traceability since the GM may be dishonest *only* in the OPEN process. Whereas, full-traceability assumes that the OPEN process is always honestly done. We note that this property was ignored in [BMW03] perhaps since in their trust model, a group manager is assumed trusted to execute the OPEN process. In general, we believe that this assumption is unrealistic. To illustrate this, we continue with the previous example. Suppose Alice and Cindy are given identical rights to sign purchase orders. Then, the group manager (colluding with Alice or not) can claim that a signature was generated by Cindy although it was actually generated by Alice.

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.[3]

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.

REMARK. On the one hand, the definition of full-anonymity [BMW03] is very strong because this definition implies previously considered definitions where anonymity is ensured for two incorrupt and honest group members. On the other hand, this definition is perhaps overly restrictive because that a group signature system, in which all the group members' private keys are compromised, is essentially useless. Nonetheless, we still follow the full-anonymity in this paper.

III. 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 basic idea underlying our proposal is the introduction of an entity called a *mediation server* $\mathcal{MS}$.[4] 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.

The rest of this section is organized as follows. First, we further motivate the need for a non-trivial solution. Section III-B presents the model for our construction and Section III-C introduces some cryptographic preliminaries used as ingredients in the construction. Then, Section III-D presents the definition of a major building block – accountable designated verifier signatures, and Section III-E gives such a concrete scheme. Finally, Section III-F presents our group signature construction, and its security is analyzed in Section III-G.

A. Further Motivation

At this point, it is natural to wonder whether a practical solution that satisfies all aforementioned requirements can be obtained in a trivial fashion. What would be a trivial solution? One natural approach is to make the group manager an online 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

[3]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 III-G for a concrete example).

[4]Note that this entity was not part of the definitions in Section II, since it is unclear if it is necessary for constructing a group signature scheme with all desirable properties. Constructing a secure *leak-free* group signature scheme with the *immediate-revocation* feature but without a mediation server is an open problem.

solution is perfect. 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). It essentially "puts all eggs in one basket." 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 [CW87] 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.

B. 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 directives) and a set of signature receivers. Each participant is modeled as a probabilistic polynomial-time interactive Turing machine.

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. Ideally, we could assume that no adversary ever gains access to this database, unless $\mathcal{MS}$ explicitly reveals some records to the adversary. To protect against potential database compromise, transaction anonymity can still be preserved if $\mathcal{MS}$ 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.)

As a more practical alternative to dummy records, we suggest that $\mathcal{MS}$ 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 IV we elaborate further 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 II 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.

C. Cryptographic Preliminaries

DIGITAL SIGNATURE SCHEMES. A digital signature scheme $\mathcal{SIG}$ consists of three algorithms, namely $\mathcal{SIG} = (\mathtt{Gen}, \mathtt{Sign}, \mathtt{Ver})$. On input a security parameter, a user $\mathcal{U}$ runs the probabilistic `Gen` to obtain a pair of public and private keys (pk, sk). On input a private key sk and a message m, $\mathcal{U}$ runs `Sign` to produce a signature $\sigma = \mathtt{Sign}_{\mathcal{U}}(m)$. On input a public key pk, a message m, and a tag σ, everyone can run `Ver` to check whether σ is a valid signature or not. We require a signature scheme to be existentially unforgeable under adaptive chosen-message attack [GMR88]. We will utilize a secure digital signature scheme without pinning down

on any concrete construction, which provides us with the flexibility when we implement the system.

DISCRETE LOGARITHM BASED CRYPTOGRAPHIC SETTING. For specific building blocks, we need a standard setting of discrete logarithm cryptosystem. Specifically, let $\mathbb{G}_q$ be the q-order subgroup of $\mathbb{Z}_p^*$, where both p and q are prime and $p = lq + 1$ such that l is co-prime to q. We will omit all the moduli when they are clear from the context.

D. Accountable Designated Verifier Signature Schemes

This is a notion that can be viewed as an enhancement of the private contract signature scheme in [GJM99]. 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, which is executed by participant P_i on message m for participant 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

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

3) ADVS-Proof, which is executed by T on input P_i, m, P_j, and a tag $\delta = \texttt{ADVS-Sign}_{P_i}(m, P_j, T)$, produces a proof via a Σ-protocol for the predicate *SignedBy*(δ, P_i) which is *TRUE* if δ is produced by P_i, and *FALSE* otherwise. (For practical reason, T typically turns such a proof into a signature using the Fiat-Shamir heuristics.)

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*.

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 [GJM99], but undesirable in our application contexts (i.e., we want to prohibit Alice from convincing XYZ that she did produce δ). A more precise explanation of this crucial difference between a private contract signature and an accountable designated verifier signature is the following: 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 transferring the bit information SignedBy(δ, P_i).

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).

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 $\delta = \texttt{ADVS-Sign}_{P_i}(m, P_j, T)$.

E. A Secure Accountable Designated Verifier Signature Scheme

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 [GJM99], which is, in turn, based on [CDS94], [C95], [JSI96]. Suppose that P_l, $l \in \{i, j\}$, has a pair of public and private keys $\langle y_l = g^{x_l}, x_l \rangle$ and that the trusted third party T has a pair of public and private keys $\langle y_T = g^{x_T}, x_T \rangle$.

For the sake of simplicity, we will use a boolean function $b()$ to denote the corresponding Σ-protocol. Thereby, P_i can generate an accountable designated verifier signature on message m for P_j by presenting a proof of a statement such as

"X is a T-encryption of 'i' AND I can sign m as P_i
OR
X is a T-encryption of 'j' AND I can sign m as P_j"

where X is some value, and T-encryption denotes a message encrypted via ElGamal using y_T. P_i can do this because she can perform a T-encryption of "i" to generate X, and she can sign m by herself. Upon receiving this proof, P_j can verify its correctness, but he cannot convince any other party that P_i produced such a proof (or the corresponding signature obtained using the Fiat-Shamir heuristics) because P_j can simply fake it. We present a concrete implementation of ADVS scheme in Appendix A with a sketched proof.

The computational complexity for the signing party is 11 exponentiations (among them 6 exponentiations can be calculated using the implementation speedup technique [MvOV96]); the computational complexity for the verifier is 12 exponentiations (all of them can be calculated using the speedup technique).

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

We are finally ready to present a concrete construction. As mentioned earlier, we use a systems approach combined with the well-known *separation-of-duty* principle [CW87]. The administrative tasks are split among two entities: group manager (GM) and mediation server ($\mathcal{MS}$). The former sets group policy, makes all decisions regarding group membership (admission/revocation), and performs the OPEN process on disputed group signatures. Whereas, group membership decisions are enforced by $\mathcal{MS}$. In other words, $\mathcal{MS}$ assists GM with membership control, specifically, to satisfy the immediate-revocation and leak-freedom requirements. In our trust model, $\mathcal{MS}$ is assumed to behave honestly with respect to its assigned tasks. However, it is not trusted to preserve anonymity.

The introduction of an on-line entity ($\mathcal{MS}$) makes our scheme different from most prior work. In addition to satisfying the above requirements, it allows us to construct a practical scheme where an anonymous channel is not necessary.

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. We also note that our trust model implies that there are no issues with fair exchange of $\delta = \texttt{ADVS-Sign}_{\mathcal{U}_i}(m, \mathcal{MS}, \mathcal{GM})$ for $\sigma = \texttt{Sign}_{\mathcal{MS}}(m)$.

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

1) The initialization of the group manager GM includes the following:
   - It chooses a security parameter κ, based on which it chooses a discrete-logarithm based *crypto-context* as specified in Section III-C. 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 specifies a set $\mathbb{U}$ so that each user or group member will be assigned with a unique identity $\mathcal{U}_i \in \mathbb{U}$.
   - It specifies (according to κ and *crypto-context*) an accountable designated verifier signature scheme ADVS as in Section III-D. In order for GM to play the role of the TTP in the ADVS scheme, it chooses a pair of public and private keys $\langle Y_{\mathcal{GM}} = g^{X_{\mathcal{GM}}}, X_{\mathcal{GM}} \rangle$.
   - It initializes a database DBUser-GM of entry structure (user-id, user-public-key, status), where "status" is for recording such information as "when the group member joins or is revoked". This database is to keep record of all the users that have ever been group members (i.e., in spite of the fact that a group member may have been revoked).
2) The initialization of the mediation server $\mathcal{MS}$ consists of the following:
   - In order for $\mathcal{MS}$ to play a role in the ADVS scheme, it chooses a pair of public and private group membership keys $\langle Y_{\mathcal{MS}} = g^{X_{\mathcal{MS}}}, X_{\mathcal{MS}} \rangle$ according to κ and *crypto-context*.
   - It chooses a pair of keys for a normal digital signature scheme $\mathcal{SIG} = (\texttt{Gen}, \texttt{Sign}, \texttt{Ver})$ that is secure against adaptive chosen-message attack (an alternative may be that GM specifies $\mathcal{SIG}$). 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 [XY02].
   - It initializes a database DBMember-MS of entry structure (group-member-id, group-member-public-key).
   - It initializes a database DBSig-MS of entry structure (group-member-id, ADVS-signature, normal-signature), which will be explained below.

JOIN. Whenever the group manager GM decides to admit a new group member, it assigns a unique identity $\mathcal{U}_i$ to the user. Then, the following protocol is executed.

1) In order for $\mathcal{U}_i$ to play a role in the ADVS scheme, it chooses a pair of public and private keys $\langle Y_{\mathcal{U}_i} = g^{X_{\mathcal{U}_i}}, X_{\mathcal{U}_i} \rangle$ according to the system-wide parameter κ and *crypto-context*.
2) GM inserts an entry $(\mathcal{U}_i, Y_{\mathcal{U}_i}, *)$ into its database DBUser-GM where "*" stands for any information GM wants to record, and may simply forward $(\mathcal{U}_i, Y_{\mathcal{U}_i})$ to the mediation server $\mathcal{MS}$ over a communication channel that is assumed to be authenticated and has no delay in delivering messages.
3) After receiving $(\mathcal{U}_i, Y_{\mathcal{U}_i})$ from GM via the authenticated communication channel, $\mathcal{MS}$ inserts an entry $(\mathcal{U}_i, Y_{\mathcal{U}_i})$ into its database DBMember-MS.

REVOKE. Whenever the group manager GM decides to revoke the membership of a group member $\mathcal{U}_i$, the following protocol is executed.

1) GM records relevant information (e.g., when membership is revoked) in the "status" column corresponding to $\mathcal{U}_i$ in its database DBUser-GM.
2) GM informs $\mathcal{MS}$ that $\mathcal{U}_i$ should be revoked over the communication channel, which is assumed to be authenticated and have no delay in delivering messages. For this purpose, $\mathcal{MS}$ simply deletes the entry $(\mathcal{U}_i, Y_{\mathcal{U}_i})$ from its database DBMember-MS.

UPDATE. This is a dummy process in our scheme, but may be important in other schemes (e.g., a signature receiver may need to obtain the current system status from the group manager).

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 $\texttt{ADVS-Ver}(m, \mathcal{MS}, \mathcal{GM}; \delta) = \text{TRUE}$. $\mathcal{MS}$ 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.[5] 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 Ver 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 $\texttt{ADVS-Ver}(m, \mathcal{MS}, \mathcal{GM}; \delta) = \text{TRUE}$; which always holds because of our trust model. Then, GM executes ADVS-Proof to produce a proof for $\text{SignedBy}(\delta, \mathcal{U}_i)$. If $\text{SignedBy}(\delta, \mathcal{U}_i)$ is TRUE, $\mathcal{U}_i$ is the signer; otherwise $\mathcal{MS}$ takes the responsibility.

G. 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.

Correctness. This can be verified by inspection.

Full-traceability. Let $\mathbb{C} \subseteq \mathbb{U}$ denote the set of participants that $\mathcal{A}$ compromised. In particular, we allow $\mathcal{A}$ to compromise all group members and partially compromise the mediation server $\mathcal{MS}$. Here, *partially* means that $\mathcal{A}$ is allowed access to $X_{\mathcal{MS}}$ – $\mathcal{MS}$' private key for ADVS. However, $\mathcal{A}$ **is not** given access to $sk_{\mathcal{MS}}$, the private key for generating external group signatures. (Otherwise, we cannot expect any traceability.) We can also allow $\mathcal{A}$ to have access to $X_{\mathcal{GM}}$, while the rest of GM's secrets remain safe. We note that, even if $\mathcal{A}$ has access to $X_{\mathcal{MS}}$, we still have to assume that all the signature generation requests are verified by the un-corrupted portion of $\mathcal{MS}$. Such an assumption not only facilitates our proof; it is also quite realistic since, for example, different address spaces may have different protection mechanisms.

Suppose that $\mathcal{A}$ that is able to produce a signature γ such that (1) γ is valid with respect to $\mathcal{MS}$' public verification key $pk_{\mathcal{MS}}$, and (2) GM cannot trace γ to any party $\mathcal{P} \in \mathbb{C} \cup \{\mathcal{MS}\}$.[6] Then, we show that either the normal signature scheme $\mathcal{SIG}$ or the ADVS scheme is broken. This leads to a contradiction.

Full-anonymity. Let $\mathbb{C} \subseteq \mathbb{U} \cup \{\mathcal{MS}\}$ denote the set of entities that the adversary $\mathcal{A}$ has compromised. Note that $\mathcal{A}$ is allowed to compromise $\mathcal{MS}$'s private keys $X_{\mathcal{MS}}$ for the ADVS scheme as well as $sk_{\mathcal{MS}}$ for the normal signature scheme $\mathcal{SIG}$. Of course, $\mathcal{A}$ does not know GM's private key $X_{\mathcal{GM}}$. Then, $\mathcal{A}$ plays the game by choosing two honest group members $\mathcal{U}_i$ and $\mathcal{U}_j$ and attempting to identify the actual signer of a given group signature γ. Clearly, without access to the database DBSig-MS, $\mathcal{A}$ has no clue about the actual signer and, thus, full-anonymity is trivially obtained. (Eavesdropping on the communication channels does not help $\mathcal{A}$ at all.)

In order to show the strength of our scheme (as was done in [BMW03] with respect to previous group signature schemes), we allow $\mathcal{A}$ to access DBSig-MS. Clearly, if we do not impose any restriction on $\mathcal{A}$'s capability in accessing to the recorded signature requests in DBSig-MS, the adversary who subverts a $\mathcal{MS}$ can immediately determine whether δ was generated by $\mathcal{U}_i$ or $\mathcal{U}_j$.

To prevent this type of triviality we require $\mathcal{MS}$ to insert into DBSig-MS two entries (which reflects that if the secrecy of database DBSig-MS is not ensured, then $\mathcal{MS}$ inserts some fake entries for a group signature, as indicated in our model): $\texttt{ADVS-Sig}_{\mathcal{U}_i}(m, \mathcal{MS}, \mathcal{GM})$ and $\texttt{ADVS-Sig}_{\mathcal{U}_j}(m, \mathcal{MS}, \mathcal{GM})$, where one of them is genuine and accepted by the un-corrupt part of $\mathcal{MS}$, while the other is faked by $\mathcal{MS}$.

Suppose that $\mathcal{A}$ has a non-negligible advantage in correctly guessing $\mathcal{U}_i$ or $\mathcal{U}_j$ as the actual signer. Then, there exists an algorithm $\mathcal{D}$ that can distinguish Diffie-Hellman triples (i.e., break the DDH assumption) with almost the same advantage. To facilitate $\mathcal{D}$, we simply let $\mathcal{D}$ embed the challenge (g, g_1, r_1, r_2) into the generation of $\texttt{ADVS-Sig}_{\mathcal{U}_i}(m, \mathcal{MS}, \mathcal{GM})$ and $\texttt{ADVS-Sig}_{\mathcal{U}_i}(m, \mathcal{MS}, \mathcal{GM})$ by setting $Y_{\mathcal{GM}} = g_1$ and the ElGamal ciphertexts be $\langle r_2^d l, r_1^d \rangle$ where $d \in_R \mathbb{Z}_q$ and $l \in_R \{i, j\}$ corresponding to the genuine request. We stress that $\mathcal{D}$ can answer all the ADVS-Proof queries because it has control over the un-corrupt part of $\mathcal{MS}$, and that knowing $X_{\mathcal{MS}}$, $sk_{\mathcal{MS}}$, and $X_{\mathcal{U}_i}$ for all $\mathcal{U}_i \in \mathbb{U}$ does not help $\mathcal{A}$. More specifically, we let $\mathcal{D}$ toss a random coin to decide which is genuine with equal probability. If the challenge (g, g_1, r_1, r_2) is a DDH tuple, then $\mathcal{A}$ has non-negligible advantage in guessing l; otherwise, $\mathcal{A}$ has no advantage at

[5] 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.

[6] This captures the fact that the adversary $\mathcal{A}$ who knows $X_{\mathcal{MS}}$ can produce an ADVS that will be accepted by the un-corrupt part of the $\mathcal{MS}$.

all. Therefore, $\mathcal{D}$ can answer with non-negligible advantage whether (g, g_1, r_1, r_2) is a DDH tuple.

No-misattribution. The group manager GM cannot misattribute $\delta = \mathsf{ADVS\text{-}Sign}_{\mathcal{U}_j}(m, \mathcal{MS}, \mathcal{GM})$ to group member $\mathcal{U}_i$, where $i \neq j$. The corresponding ElGamal ciphertext is denoted by $\langle A = j \cdot Y^k_{\mathcal{MS}}, g^k \rangle$. If GM can present a proof for SignedBy$(\delta, \mathcal{U}_i)$, then the Forking Lemma shows that there are two accepting transcripts. This is impossible because the input is not in the language.

Leak-freedom. Recall that $\mathcal{MS}$ cannot convince anyone that a given signature γ is produced by a certain group member. This is implied by the non-transferability of the underlying ADVS scheme. We note that attacks mentioned in [XY02] must be prevented by using appropriate initialization.

On the other hand, a group member $\mathcal{U}_i$ cannot convince anyone, except $\mathcal{MS}$, that a given signature γ is requested by $\mathcal{U}_i$. This is because we assume that the database DBSig-MS is not available to such an adversary. (A *strong-secure* ADVS scheme would ease this assumption; see Section IV for further discussion.)

Immediate-revocation. Suppose the channel between $\mathcal{MS}$ and GM is authenticated and has no delay in delivering messages. If a malicious user $\mathcal{U}_i$ is able to obtain a group signature after having been revoked, then there is an algorithm $\mathcal{F}$ that breaks either the ADVS scheme or the normal signature scheme $\mathcal{SIG}$.

REMARK. Suppose that $\mathcal{A}$ has completely compromised $\mathcal{MS}$, but no other party. Then, the system becomes leak-free but does not retain full-anonymity. Therefore, leak-freedom does not necessarily imply full-anonymity; they are geared to address different types of attacks.

IV. Extension and Discussion

Enhancing anonymity against traffi c 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: $(\mathcal{U}_i, \delta = \mathsf{ADVS\text{-}SIG}_{\mathcal{U}_i}(m, \mathcal{MS}, \mathcal{GM}), \mathtt{Sign}_{\mathcal{MS}}(m))$. 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 $\mathtt{Sign}_{\mathcal{MS}}(m)$. Clearly, if a *strong-secure* 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 $(\text{Alice}, \delta = \mathsf{ADVS\text{-}SIG}_{\text{Alice}}(m, \mathcal{MS}, \mathcal{GM}), \mathtt{Sign}_{\mathcal{MS}}(m))$.

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 $\mathcal{MS}$' 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.

A. Related Work

This paper can be viewed as one among many efforts pursuing practical and secure *group signature* or *identity escrow* schemes [CvH91], [CP94], [CS97], [KP98], [ACJT00], [CL01a], as well as anonymous *credential systems* [C85], [C85a], [CE86], [LRSW99], [CL01], [CvH02]. Among them, the prior work most relevant to this paper is [CL01a], 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 [CL01a] and the leak-freedom property specified

in this paper. Specifically, the [CL01a] 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 I-B) that we want to avoid!

Besides achieving the strictly stronger leak-freedom, our scheme is more efficient than [CL01a] 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 [CL01a], whereas, we achieve immediate-revocation which has only been explored in the context of traditional PKI-s [BDTW01].

A credential system is a system where users can obtain credentials from organizations and demonstrate possession of these credentials [C85], [C85a], [CE86], [LRSW99], [CL01], [CvH02]. Chaum and Evertse [CE86] 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. [LRSW99] captures many of the desirable properties. Camenisch and Lysianskaya [CL01] presented a better solution with ingredients from a secure group signature scheme of [ACJT00]. The prototype implementation of [CL01] was done by Camenisch and van Herreweghen [CvH02]. 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 [GJM99], 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.[7] Moreover, leak-freedom is similar to receipt-freedom property that has been investigated in the context of voting schemes [BT94], [HS00]. 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.

V. 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.

[7] 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 V below.

References

[A97] R. Anderson. Invited Talk at ACM CCS'97.

[ACJT00] G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A Practical and Provably Secure Coalition-Resistant Group Signature Scheme. Crypto'00.

[AST02] G. Ateniese, D. Song, and G. Tsudik. Quasi-Efficient Revocation of Group Signatures. Financial Crypto'02.

[AT99] G. Ateniese and G. Tsudik. Some Open Issues and New Directions in Group Signatures. Financial Crypto'99.

[BMW03] M. Bellare, D. Micciancio, and B. Warinschi. Foundations of Group Signatures: Formal Definitions, Simplified Requirements, and a Construction Based on General Assumptions. Eurocrypt'03, to appear.

[BM99] M. Bellare and S. Miner. A Forward-Secure Digital Signature Scheme. Crypto'99.

[BR93] M. Bellare and P. Rogaway. Random Oracles Are Practical: A Paradigm for Designing Efficient Protocols. ACM CCS'93.

[BT94] J. Benaloh and D. Tuinstra. Receipt-free Secret-Ballot Election (Extended Abstract). ACM STOC'94.

[BDTW01] D. Boneh, X. Ding, G. Tsudik, and M. Wong. A Method for Fast Revocation of Public Key Certificates and Security Capabilities. Usenix Security'01.

[BS01] E. Bresson and J. Stern. Group Signatures with Efficient Revocation. PKC'01.

[C98] J. Camenisch. Group Signature Schemes and Payment Systems Based on the Discrete Logarithm Problem. PhD Thesis. ETH Zurich 1998.

[CL01] J. Camenisch and A. Lysyanskaya. An Efficient System for Non-transferable Anonymous Credentials with Optional Anonymity Revocation. Eurocrypt'01.

[CL01a] J. Camenisch and A. Lysyanskaya. An Identity Escrow Scheme with Appointed Verifiers. Crypto'01.

[CL02] J. Camenisch and A. Lysianskaya. Dynamic Accumulators and Application to Efficient Revocation of Anonymous Credentials. Crypto'02.

[CM98] J. Camenisch and M. Michels. A Group Signature Scheme Based on an RSA-Variant. BRICS TR-98-27. Preliminary version in the proceedings of Asiacrypt'98.

[CS97] J. Camenisch and M. Stadler. Efficient Group Signature Schemes for Large Groups (Extended Abstract). Crypto'97.

[CvH02] J. Camenisch and E. van Herreweghen. Design and Implementation of the *idemix* Anonymous Credential System. ACM CCS'02.

[C85] D. Chaum. Showing Credentials without Identification. Eurocrypt'85.

[C85a] D. Chaum. Security without Identification: Transactions Systems to Make Big Brother Obsolete. C. ACM 28(10): 1030-1044, 1985.

[CE86] D. Chaum and J. Evertse. A Secure and Privacy-Protecting Protocol for Transmitting Personal Information between Organizations. Crypto'86.

[CvH91] S. Chaum and E. van Heyst. Group Signatures. Eurocrypt'91.

[CP94] L. Chen and T. Pedersen. New Group Signature Schemes. Eurocrypt'94.

[C95] R. Cramer. Modular Design of Secure yet Practical Cryptographic Protocols. PhD Thesis, 1995.

[CDS94] R. Cramer, I. Damgard, and B. Schoenmakers. Proofs of Partial Knowledge and Simplifi ed Design of Witness Hiding Protocols. Crypto'94.

[CW87] D. Clark and D. Wilson. A Comparison of Commercial and Military Computer Security Policies. IEEE Symposium on Security and Privacy. 1987.

[DDN91] D. Dolev, C. Dwork, and M. Naor. Nonmalleable Cryptography. ACM STOC'91.

[E85] T. ElGamal. A Public-Key Cryptosystem and a Signature Scheme Based on the Discrete Logarithm. IEEE Transactions of Information Theory, 31(4), 1985, pp 469–472.

[FS86] A. Fiat and A. Shamir. How to Prove Yourself: Practical Solutions to Identifi cation and Signature Problems. Crypto'86.

[GHY85] Z. Galil, S. Haber, and M. Yung. Symmetric Public-Key Encryption. Crypto'85.

[GJM99] J. Garay, M. Jakobsson, and P. MacKenzie. Abuse-Free Optimistic Contract Signing. Crypto'99.

[GM84] S. Goldwasser and S. Micali. Probabilistic Encryption. Journal of Computer and System Sciences, vol. 28, no. 1, 1984, pp 270-299.

[GMR88] S. Goldwasser, S. Micali, R. Rivest. A Digital Signature Scheme Secure against Adaptive Chosen-message Attacks. SIAM J. Computing, 17(2), 1988.

[HS00] M. Hirt and K. Sako. Effi cient Receipt-free Voting Based on Homomorphic Encryption. Eurocrypt'00.

[JSI96] M. Jakobsson, K. Sako, and R. Impagliazzo. Designated Verifi er Proofs and Their Applications. Eurocrypt'96.

[K03] J. Katz. Effi cient and Non-Malleable Proofs of Plaintext Knowledge and Applications. Eurorypt'03. to appear.

[KP98] J. Kilian and E. Petrank. Identity Escrow. Crypto'98.

[KY03] A. Kiayias and M. Yung. Extracting Group-Signatures from Traitor Tracing Schemes. Eurocrypt'03, to appear.

[LRSW99] A. Lysyanskaya, R. Rivest, A. Sahai, and S. Wolf. Pseudonym Systems. SAC'99.

[MvOV96] A. Meneze, P. van Oorschot, and S. Vanstone. Handbook of Applied Cryptography. CRC Press. 1996.

[NY90] M. Naor and M. Yung. Public-key Cryptosystems Provably Secure against Chosen Ciphertext Attacks. ACM STOC'90.

[PS96] D. Pointcheval and J. Stern. Security Proofs for Signature Schemes. Eurocrypt'96.

[RS91] C. Rackoff and D. Simon. Non-interactive Zero-Knowledge Proof of Knowledge and Chosen Ciphertext Attack. Crypto'91.

[S89] C. P. Schnorr. Effi cient Identifi cation and Signatures for Smart Cards. Crypto'89.

[S01] D. Song. Practical Forward-Secure Group Signature Schemes. ACM CCS'01.

[TY98] Y. Tsiounis and M. Yung. On the Security of ElGamal Based Encryption. PKC'98.

[XY02] S. Xu and M. Yung. The Dark Side of Threshold Cryptography. Financial Crypto'02.

APPENDIX

APPENDIX A: A SECURE ACCOUNTABLE DESIGNATED VERIFIER SIGNATURE SCHEME

We construct a secure accountable designated verifier signature scheme below and present an informal proof.

1) ADVS-Sign: P_i generates an ElGamal encryption of $i \in \mathbb{G}_q$ using y_T; denote the ciphertext by $\langle A = i \cdot y_T^k, B = g^k \rangle$ where $k \in_R \mathbb{Z}_q$. Then it executes as follows:
   a) Choose $k_1', k_2' \in_R \mathbb{Z}_q$ and compute $A' = y_T^{k_1'}$, $B' = g^{k_1'}$, and $C' = g^{k_2'}$.
   b) Choose $d_1^*, c_2, d_2^* \in_R \mathbb{Z}_q$ and compute $A^* = y_T^{d_1^*}(A/j)^{-c_2}$, $B^* = g^{d_1^*} B^{-c_2}$, and $C^* = g^{d_2^*} y_j^{-c_2}$.
   c) Compute $c = H(m, i, j, T, y_i, y_j, y_T, A, B, A', B', C', A^*, B^*, C^*)$, where $H : \{0,1\}^* \rightarrow \mathbb{Z}_q$ behaves like a random oracle.
   d) Compute $c_1 = c - c_2 \bmod q$.
   e) Compute $d_1' = c_1 \cdot k + k_1' \bmod q$ and $d_2' = c_1 \cdot x_i + k_2' \bmod q$. The accountable designated verifier signature on message m is $\delta = (m, i, j, T, y_i, y_j, y_T, A, B, c_1, c_2, A', B', C', d_1', d_2', A^*, B^*, C^*, d_1^*, d_2^*)$.
2) ADVS-Ver: Given $\delta = (m, i, j, T, y_i, y_j, y_T, A, B, c_1, c_2, A', B', C', d_1', d_2', A^*, B^*, C^*, d_1^*, d_2^*)$, P_j can verify if: $c_1 + c_2 = H(m, i, j, T, y_i, y_j, y_T, A, B, A', B', C', A^*, B^*, C^*)$, $y_T^{d_1'}(A/i)^{-c_1} = A'$, $g^{d_1'} B^{-c_1} = B'$, $g^{d_2'} y_i^{-c_1} = C'$, $y_T^{d_1^*}(A/j)^{-c_2} = A^*$, $g^{d_1^*} B^{-c_2} = B^*$, and $g^{d_2^*} y_j^{-c_2} = C^*$.
3) ADVS-Proof: Given δ, T publishes a proof for $log_g y_T = log_B(A/i)$, which corresponds to the predicate SignedBy(δ, P_i). The details are straightforward and omitted.

Theorem 1.1: The above ADVS scheme is secure.

Proof: (sketch) We show that the scheme satisfies the Definition 3.3.

- Unforgeability of ADVS-Sign$_{P_i}(m, P_j, T)$. Suppose there is a probabilistic polynomial-time algorithm $\mathcal{A}$ that doesn't know x_i and x_j, and is nevertheless able to forge a $\delta =$ ADVS-Sign$_{P_i}(m, P_j, T)$ with non-negligible probability. Then there is a probabilistic polynomial-time algorithm $\mathcal{B}$ that is able to break the discrete logarithm assumption (which implies the DDH assumption). Basically, $\mathcal{B}$ embeds a challenge discrete logarithm instance as the pair of public and private keys of (y_i, x_i) or (y_j, x_j), with equal probability. The Forking Lemma [PS96] shows that if $\mathcal{A}$ is able to forge a δ, then $\mathcal{B}$ can obtain two accepting transcripts for the corresponding Σ-protocol, and thus either x_i or x_j is extracted.
- Non-transferability of ADVS-Sign$_{P_i}(m, P_j, T)$. The semantic security of ElGamal implies that one cannot distinguish the encryption of "i" from the encryption of "j". Moreover, P_j can fake a δ' such that ADVS-Ver$(m, P_i, P_j; \delta') =$ TRUE using the same algorithm in the above scheme.
- Unforgeability of the proof for SignedBy(δ, P_i). Suppose there is a probabilistic polynomial-time algorithm $\mathcal{F}$, other than P_i and T, that is able to forge a proof for SignedBy(δ, i), no matter whether the valid δ is produced by P_i or P_j (note that the unforgeability of ADVS-Sign$_{P_i}(m, P_j, T)$ implies that no other party can produce such a δ).
  Suppose δ is indeed produced by P_i. Then we can construct a polynomial-time algorithm $\mathcal{D}$ to break the DDH assumption. Given a challenge (g, g_1, r, r_1), $\mathcal{D}$ simply sets $y_T = g_1$, $B = r$ and $A = r_1 \cdot i$. Note that $\mathcal{D}$ can answer all ADVS-Proof queries by having control over P_j. If the challenge is a DDH instance, then $\mathcal{F}$ succeeds with non-negligible probability; otherwise,

$\mathcal{F}$ cannot. If $\mathcal{F}$ succeeds, then $\mathcal{D}$ bets that the challenge is a DDH instance; otherwise, outputs a random guess. Therefore, $\mathcal{D}$ succeeds with non-negligible advantage.
Suppose δ is produced by P_j. This means that $(A = j \cdot y_T^k, B = g^k)$ for some $k \in \mathbb{Z}_q$; otherwise, the Forking Lemma allows us either to extract x_i or to have two accepting transcripts for $log_g B = log_{y_T}(A/j)$, either of which is a contradiction. If $\mathcal{F}$ is able to produce such a proof that $log_g B = log_{y_T}(A/i)$, then the Forking Lemma shows that we can get two accepting transcripts, which is impossible since the input is not in the language.

■