7-2009

# Leak-Free Mediated Group Signatures

Xuhua DING
*Singapore Management University*, xhding@smu.edu.sg

Gene TSUDIK
*University of California - Irvine*

Shouhuai XU
*University of Texas at San Antonio*

## Citation

DING, Xuhua; TSUDIK, Gene; and XU, Shouhuai. Leak-Free Mediated Group Signatures. (2009). *Journal of Computer Security*. 17, (4), 489-514.
**Available at:** https://ink.library.smu.edu.sg/sis_research/792

# Leak-free Mediated Group Signatures[*]

Xuhua Ding

School of Information Systems

Singapore Management University

xhding@smu.edu.sg

Gene Tsudik

Department of Information and Computer Science

University of California at Irvine

gts@ics.uci.edu

Shouhuai Xu

Department of Computer Science

University of Texas at San Antonio

shxu@cs.utsa.edu

**Abstract**

Group signatures are a useful cryptographic construct for privacy-preserving non-repudiable authentication, and there have been many group signature schemes. In this paper, we introduce a variant of group signatures that offers two new security properties called leak-freedom and immediate-revocation. Intuitively, the former ensures that an insider (i.e., an authorized but malicious signer) be unable to convince an outsider (e.g., signature receiver) that she indeed signed a certain message; whereas the latter ensures that the authorization for a user to issue group signatures can be immediately revoked

[*]An earlier version appeared in the Proceedings of IEEE ICDCS'04 [32].

whenever the need arises (temporarily or permanently). These properties are not offered in existing group signature schemes, nor captured by their security definitions. However, these properties might be crucial to a large class of enterprise-centric applications because they are desirable from the perspective of the enterprises who adopt group signatures or are the group signatures liability-holders (i.e., will be hold accountable for the consequences of group signatures). In addition to introducing these new security properties, we present a scheme that possesses both traditional and these newly introduced properties. Our scheme is constructed using an architectural approach where a mediation server is exploited to trade on-line communications for the extra security properties, which explains why the resulting scheme is called "leak-free mediated group signatures."

## 1 Introduction

The concept of group signatures was introduced by Chaum and van Heyst [27]. Intuitively, a group signature can be seen as a normal digital signature with some extra properties:

> Any verifier can establish that a valid (i.e., verifiable) group signature was generated by a legitimate group member (i.e., one of a set of possible signers), while the actual signer can only be identified by a designated entity, called 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. More specifically, group signatures facilitate the following functionalities simultaneously: (1) only a group member can sign on behalf of the group; (2) anyone can verify a group signature, (3) no one, except a designated entity called *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.

Group signatures have many applications. In particular, they can be used as foundation for anonymous credential systems in various application contexts (cf. [16, 18]).

In the past decade, many research efforts and results focused on seeking precise definitions and efficient constructions of group signature schemes [5, 43, 13, 10, 20], or their interactive dual, *identity escrows* [45]. In particular, 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*, which were developed in a series of investigations [27, 28, 22, 16, 21, 45, 4, 15, 54, 3, 17]. To untangle and simplify these somewhat messy requirements, Bellare et al. [5] 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 [38, 49, 51, 33] and *secure* key exchange protocols. Specifically, Bellare et al. showed that two security properties, *full-traceability* and *full-anonymity*, are sufficient to subsume all of the above seven requirements (but not an important property known as *no-misattribution*).

## 1.1 Motivation: New Security Requirements

In this section we show that, in addition to the aforementioned *full-traceability*, *full-anonymity* and *no-misattribution*, there are other important properties that are crucial to a large class of group signature applications. Two properties we identify are: leak-freedom and immediate-revocation. We briefly explain these two concepts below and highlight their importance in from an application perspective. A more formal treatment including definitions and analysis is given in Section 2. Intuitively, 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.

**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.

Informally, 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, the liability for such a "poisoned" group signature can not be relegated to Alice; thus, the company has to bear all attendant cost and responsibilities. We remark that forward-security [1, 6, 54] 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.

## 1.2 Our Contributions

The concept of group signatures is not only a cryptographic notion, but has a wide spectrum of practical applications. In this paper, instead of studying it as a cryptographic primitive from a theoretic perspective, we consider the security issues of applying group signatures within a large organization.

The main contribution of this paper is twofold: First, we identify the aforementioned two important security properties: leak-freedom and immediate-revocation which are necessary for a large class of group signature applications. Second, we propose a scheme that possesses both traditional and the newly introduced security properties. Specifically, our scheme exploits a semi-trusted but online entity called *mediation server* to achieve immediate-revocation, and a cryptographic trick to fulfill leak-freedom. Therefore, we call the resulting scheme "leak-free mediated group signatures." Third, our scheme outperforms other group signature schemes on the following aspects.

1. **Signature Issuance/Verification**: Our scheme only needs 11 modular exponentiations to generate a group signature. More importantly, the resulting group signature is in the form of plain non-group digital signatures, such as RSA signatures. Therefore, its signature length is the shortest. Its verification is equivalent to verifying a single plain signature. This is appreciably more efficient than the state-of-the-art [2, 17, 10]. This is also relevant when comparing the proposed scheme with an imaginary two-party mediated group signature scheme (i.e., the signing capability of an authorized user is distributed to the user and the mediation server) — even if it is made leak-free.

2. **User Revocation**: In other group signature schemes [17, 56, 10] that support revocations, a revocation requires the group manager to update the cryptographic setting e.g., the group public key. The new setting should be broadcast to all group members and the verifiers. In our scheme, no changes are

made in the cryptographic setting, meaning that the process of revocation is transparent to all other group members and the verifiers. Our instant revocation feature releases the verifiers the burden of revocation checking and minimize the incurred computation cost.

3. **Member JOIN**: In all other group signature schemes, JOIN is a costly process. It takes a heavy toll on the group manager to issue membership certificates. In our construction, the JOIN process is as simple as the normal certification of a private signing key.

4. **Communication Channel**: Our scheme allows us to relax the requirement for the underlying anonymous communication channel, which is essential in all previous schemes.

Caveat on trading on-line interaction for new security properties: Compared with prior *non-interactive* group signature schemes, our scheme requires maintenance of an online server and one signer-server communication per signature. The architecture of our scheme resembles those used in [11] and [47]. All these architectures feature an online semi-trusted server assisting users to perform cryptographic operations. The notable drawbacks of this approach are the cost of interactions and the risk of server failure. Nonetheless, the payback justifies the cost not only because the new security properties — such as fine-grained control in [11] and immediate-revocation as well as leak-freedom in the present paper — are important, but also because the new security properties are very difficult to fulfill otherwise (if not impossible — we leave this as an important open problem). Therefore, it seems reasonable to trade on-line interactions for the aforementioned two security properties. (It is interesting to note that trade-offs of this kind have been exploited in other contexts and applications [36, 42]). Moreover, our scheme would be easy to implement because the mediation server can be integrated with an organization's email server or web service server.

## 1.3 Related Works

This paper can be viewed as one among many efforts pursuing practical and secure *group signature* or *identity escrow* schemes [27, 28, 22, 45, 2, 19], as well as anonymous *credential systems* [24, 25, 26, 46, 18, 23].

**Related works in group signatures**. In recent years, major research theme has been to construct practical group signature schemes. Early schemes (e.g., [28]) 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 [22] 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., [21] and [2], 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 [15], Song [54], and Ateniese, et al. [3] as well as Camenisch and Lysyanskaya [17].

Unfortunately, even in these schemes, revocation incurs a linear dependency on either the number of current, or the 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 Lysyanskaya [17] (which allows for an efficient proof that a group member is not revoked) and the "bare" group signature scheme of [2] (which allows efficient proof of knowledge of a secret key corresponding to a valid certificate). A more concise and integrated scheme is presented in [56]. However, even in [17, 56], 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. Other works on group signatures include, Kiayias and Yung's scheme [44] with an efficient and secure user joining protocol and Boyen and Water's scheme [14] which is proven secure in the standard model with the signature length being logarithmic in the number of signers.

Among all these prior works, the one that is most relevant to this paper is [19], 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 [19] and the leak-freedom property specified in this paper. Specifically, the [19] 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 1.1) that we want to avoid! Besides achieving the strictly stronger leak-freedom, our scheme is more efficient than [19] 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 [19], whereas, we achieve immediate-revocation which has only been explored in the context of traditional PKI-s [12].

**Related works in anonymous credentials**. A credential system is a system where users can obtain credentials from organizations and demonstrate possession of these credentials. Chaum and Evertse [26] presented a general scheme using a semi-trusted TTP common to multiple organizations. However, their scheme is impractical. The credential system by Lysyanskaya, et al. [46] captures many of the desirable properties. Camenisch and Lysyanskaya [18] presented a better solution with ingredients from a secure group signature scheme of [2]. The prototype implementation of [18] was done by Camenisch and van Herreweghen [23]. 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.

**Other loosely related works**. The notion called *abuse-freedom* investigated in the context of contract signing [37], is weaker than leak-freedom because the former only intends 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. Moreover, leak-freedom is similar to receipt-freedom property that has been investigated in the context of voting schemes [8, 40]. 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.

The leak-freedom property of group signature schemes introduced in this paper may be reminiscent of the "deniability" of ring signatures [52, 9]. A ring signature scheme can be seen as a group signature scheme without a setting-up process. However, exactly because of this ring signatures do not appear to be appropriate for the applications we target in this paper.

### 1.4 Outline

The rest of this paper is organized as follows: the next section provides the model and definition of leak-free mediated group signatures, Section 3 presents a leak-free mediated group signature scheme. Next, Section 4 discusses some extensions and Section 5 concludes the paper.

## 2 Leak-free Mediated Group Signatures: Model and Definition

In this section we present the functionality and security specifications of the mediated group signature scheme.

### 2.1 Model

Participants: A set of group members $\mathbb{U}$, a group manager $\mathcal{GM}$ who admits group members, a mediation server $\mathcal{MS}$, and a set of signature receivers. Each participant is modeled as a probabilistic polynomial-time interactive Turing machine. The role of $\mathcal{MS}$ is to facilitate group members' signature generation and to revoke group members on $\mathcal{GM}$'s directives; this abides by the well-known *separation-of-duty* principle [29].

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. $\mathcal{MS}$ encrypts every entry in the database with $\mathcal{GM}$'s public encryption key. Although this incurs slight additional complexity, the OPEN process that is only occasionally invoked remains efficient. In Section 4 we elaborate further on the issue of database secrecy. Besides this database, both $\mathcal{MS}$ and $\mathcal{GM}$ maintain a dynamic membership database that allows both *insert* and *delete* operations but cannot be tampered with by unauthorized parties. This is not new since similar assumptions are made in all prior group signature schemes.

Communication Channels: Our model consists of four kinds of communication channels: 1) the channels between group users and the $\mathcal{GM}$, 2) the channels between the $\mathcal{GM}$ and the $\mathcal{MS}$, 3) the channels between group users and the $\mathcal{MS}$, and 4) the channels between the $\mathcal{MS}$ and signature recipients. All of these channels are assumed to be public (i.e., not private), but the first two kinds of channels are assumed to be authenticated.

TRUST: Providing a precise specification of the trust model turns out to be difficult mainly because of the introduction of the new party: $\mathcal{MS}$. Nevertheless, In 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, $\mathcal{GM}$ may want to frame an honest group member.

2. $\mathcal{MS}$ is trusted to enforce $\mathcal{GM}$'s policy, e.g., to stop services for the revoked group members as requested by $\mathcal{GM}$ and to produce group signatures only for legitimate members. In the suggested system configuration where $\mathcal{MS}$ delivers group signatures, $\mathcal{MS}$ will introduce appropriate delay for blocking trivial traffic analysis attack. Nonetheless, $\mathcal{MS}$ may want to: 1) forge an honest member's group signatures, 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).

## 2.2 Definition

**Definition 1** *A* **leak-free mediated group signature scheme** *consists of the following six procedures:*

- `SETUP`*: A probabilistic algorithm that, on input of a security parameter $\rho$, outputs the group public key $pk_{\mathcal{G}}$ (including all system parameters), the group manager's public/secret key pair: $pk_{\mathcal{GM}}/sk_{\mathcal{GM}}$, the mediation server's public/secret key pair $pk_{\mathcal{MS}}/sk_{\mathcal{MS}}$.*

- `JOIN`*: A protocol between the group manager $\mathcal{GM}$ 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 executed by $\mathcal{GM}$ and $\mathcal{MS}$, which, on input of the 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.*

- `SIGN`*: A probabilistic algorithm jointly executed by user $\mathcal{U}$ and $\mathcal{MS}$, which, on input of a group public key $pk_{\mathcal{G}}$, $\mathcal{U}$'s membership secret/public key-pair $(sk_{\mathcal{U}}, pk_{\mathcal{U}})$, $\mathcal{MS}$'s public/secret key-pair $(pk_{\mathcal{MS}}, sk_{\mathcal{MS}})$ and a message $m$, outputs a group signature $\delta$ on $m$.*

- `VERIFY`*: A public algorithm that, on input of a group public key $pk_{\mathcal{G}}$, a group signature $\delta$ and a message $m$, outputs a binary value* TRUE/FALSE *indicating whether $\delta$ is a valid group signature of $m$.*

- `OPEN`*: An algorithm jointly executed by $\mathcal{GM}$ and $\mathcal{MS}$, which takes as input of a message $m$, a group signature $\delta$, the group public key $pk_{\mathcal{G}}$ and the group manager's secret key $sk_{\mathcal{GM}}$. It first executes* `VERIFY` *on the first three inputs and, if the $\delta$ is valid, outputs some incontestable evidence (e.g., a membership public key $pk_{\mathcal{U}}$ and a proof) that either allows anyone to identify the actual signer or determine that $\mathcal{MS}$ cheated.*

## 2.3 Security

We now summarize the security properties of mediated group signatures: correctness, full-traceability, full-anonymity, no-misattribution, leak-freedom and immediate-revocation. Among them, the leading four properties are defined based on their counterparts of traditional group signatures [5], with slight modification. leak-freedom and immediate-revocation are defined for the first time. Although it is straightforward to extend the two definitions to traditional group signatures, we conjecture that it is difficult to achieve them with existing techniques.

**Definition 2 (Security Properties of Mediated Group Signatures)**

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

- Full-traceability*: No collusion among group members and $\mathcal{MS}$ (even consisting of the entire group, and even being in possession of the group manager's secret key) can create valid signatures that cannot be* *opened, or signatures that cannot be traced back to some member in collusion. We remark that allowing the adversary to know the group manager's secret key for opening signatures is to show the strength of a group signature scheme, and not for accommodating corruption of the group manager.*

- 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 and $\mathcal{MS}$.*

- No-misattribution*: It is infeasible for the $\mathcal{GM}$ to frame an honest group member (by misbehaving during the* `OPEN` *process). This property has been implicitly considered in prior group signature schemes (e.g., [2]), 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 $\mathcal{GM}$ may be dishonest* only *in the* `OPEN` *process. In contrast,* full-traceability *assumes that the* `OPEN` *process is always honestly done. We note that this property was ignored in [5] perhaps since in their trust model, a group manager is trusted to honestly execute the* `OPEN` *process. We observe that this assumption in general is impractical. To illustrate this, we continue with the previous example. Suppose that Alice and Cindy are given identical rights to sign purchase orders. Then, the group manager (colluding with Alice or not) can maliciously claim that a signature was generated by Cindy although it was actually generated by Alice.*

- Leak-freedom*: It is infeasible for a signer to prove the ownership of a given group signature without giving away her private key, even if she is in possession of all other members' secrets (but not the secret key of the group manager for opening signatures). Note that the same information could also be available to an adversary targeting* full-anonymity*. While* full-anonymity *does not guarantee a signer's incapability to convince anyone that she generated a given signature,* leak-freedom *does not necessarily imply* full-anonymity *either (see the remark in Section 3.6 for a concrete example).*

- Immediate-revocation*: It is infeasible for a 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 I. It is clear that leak-freedom becomes irrelevant when an authorized signer is willing to give away her private key to someone else. Also, we do not consider some possible "semantics-based" attacks such as the message that is being signed is embedded with randomness (e.g., hash of a string) provided by one to whom the signer plans to prove the ownership of group signature. This kind of attack can be defeated by only allowing "randomness-free" messages with a prescribed format. It is worthwhile to note that leak-freedom is a security property against potentially dishonest signers, not against the $\mathcal{GM}$ because the $\mathcal{GM}$ must be able to identify the actual signer in a no-misattribution fashion.

Remark II. The immediate-revocation property releases verifiers from the burden of CRL or time-stamp checking, since at the moment a group signature was issued, the signer's secret key was verified valid. This is important not only when an authorized signer need be permanently revoked, but also when an authorized signer need be temporarily revoked.

## 3 Leak-free Mediated Group Signatures: Construction

In contrast to most prior works, our construction is designed from a *system*, rather than purely cryptographic, perspective, though cryptography still plays a major role. 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. Then, Section 3.2 introduces some cryptographic preliminaries used as ingredients in the construction. Section 3.3 presents the definition of a major building block – accountable designated verifier signatures, and Section 3.4 gives such a concrete scheme. Finally, Section 3.5 presents our group signature construction, and its security is analyzed in Section 3.6.

## 3.1 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. One natural approach is to make the group manager an on-line entity and have it to "filter" all group signature requests. Each group member has an anonymous channel to the group manager $\mathcal{GM}$ and, for each message to be signed, it submits a message signed under its normal long-term signature key. $\mathcal{GM}$ then "translates" each incoming signature into a signature under its own well-known group public 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 $\mathcal{GM}$ can be assured, then the trivial solution is perfect. However, having a **fully-trusted on-line** entity is typically undesirable and sometimes simply unrealistic. Moreover, such an entity would be the single point of failure in the sense of both security (i.e., compromise of $\mathcal{GM}$ means compromise of the whole system) and anonymity (i.e., a dishonest $\mathcal{GM}$ 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 it might require 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 [29] for necessary background.) This approach, as will be shown below, facilitates a similar flavor of distributed security. Namely, compromise of either $\mathcal{GM}$ or the newly introduced mediation server $\mathcal{MS}$, but not both, does not necessarily imply complete compromise of the system since opening a group signature requires cooperations between $\mathcal{GM}$ and $\mathcal{MS}$.

## 3.2 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 $\mathtt{Gen}$ to obtain a pair of public and private keys $(pk, sk)$. On input a private key $sk$ and a message $m$, $\mathcal{U}$ runs $\mathtt{Sign}$ to produce a signature $\sigma = \mathtt{Sign}_{\mathcal{U}}(m)$. On input a public key $pk$, a message $m$, and a tag $\sigma$, everyone can run $\mathtt{Ver}$ to check whether $\sigma$ is a valid signature or not. We require a signature scheme to be existentially unforgeable under adaptive chosen-message attack [39]. 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 primes 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.

Proof of Knowledge of Discrete Logarithm. Suppose in a discrete logarithm setting, that Alice knows that the discrete logarithm of $y$ with respect to base $g$ is $x$, i.e. $y = g^x$. Alice can prove to Bob of her knowledge of $x$ without revealing any information on $x$. Such an interactive proof can be easily converted into the so-called *signature of proof of knowledge* under the random oracle model [7]. Alice's signature proof is a tuple $(c, s)$, satisfying $c = \mathcal{H}(m \| g^s y^{-c})$, where $\mathcal{H}()$ denotes a cryptographically secure hash function (drawn from a family of hash functions). For security reason, we require that the security parameter $\kappa \leq \log q$. This way, $g^s$ for a random $s$ takes a specific value is no more than $1/(q-1) \leq 2/2^{\kappa}$.

Proof of ElGamal Ciphertext Content. Suppose that Bob has a pair of ElGamal [34] public and private keys $\langle y = g^x, x\rangle$ where $g$ is of order $q$ and generates $\mathbb{G}_q$, namely $\mathbb{G}_q = \langle g \rangle$. To generate an ElGamal encryption of a plaintext $m \in \mathbb{G}_q$ for Bob, Alice chooses $k \in_R \mathbb{Z}_q^*$ and sends Bob $\langle y^k m, g^k\rangle$. Bob decrypts an incoming message $\langle A, B\rangle$ by computing $A/B^x$. Security of ElGamal encryption can be based on the intractability of Decision Diffie-Hellman (DDH) problem [55].

Given an ElGamal ciphertext $\langle A, B\rangle = \langle y^k m, g^k\rangle$, both Alice and Bob can prove that $m$ is the corresponding plaintext using standard techniques. Specifically, Alice proves $log_y(A/m) = log_g B$ and Bob proves $log_g y = log_B(A/m)$ using a $\Sigma$-protocol [30] such as Schnorr's [53]. Alice's proof of ElGamal ciphertext content with respect to $(A, B)$ is a tuple $(c, s)$, satisfying $c = \mathcal{H}(y^s (A/m)^{-c} \| g^s B^{-c})$. Similarly, Bob's proof is a tuple $(c, s)$ satisfying $c = \mathcal{H}(B^s (A/m)^{-c} \| g^s y^{-c})$. With such $\Sigma$-protocols, conjunctions and disjunctions (i.e., ANDs and ORs) of certain statements/predicates [31] can be proved as well. Moreover, one can use the Fiat-Shamir heuristics [35] to transform an interactive $\Sigma$-protocol into a signature scheme, which is nevertheless secure in the random oracle model.

## 3.3 Definition of Accountable Designated Verifier Signatures

This is a notion that can be viewed as an enhancement of the private contract signature scheme in [37]. Informally, a private contract signature is a designated verifier signature that can be converted into a universally-verifiable signature by either the signer or a trusted third party (TTP) appointed by the signer. The TTP's identity and power to convert can be verified without interaction by 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** *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 only $P_j$ to verify the validity of an input tag $\delta$ so that*

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

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

Remark We stress that the above definition does not capture whether $P_i$ should be able to produce a proof for $\text{SignedBy}(\delta, P_i)$. This capability is necessary in the context of [37], but undesirable in our application contexts. Imagine that we want to prevent Alice from convincing XYZ that she did produce $\delta$. 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 $\text{SignedBy}(\delta, P_i)$ by whatever means, whereas the latter intends to prevent both $P_i$ *and* $P_j$ from transferring the bit information $\text{SignedBy}(\delta, P_i)$.

**Definition 4** *An accountable designated verifier signature scheme is* secure *if all 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* $\delta$ *such that* $\texttt{ADVS-Ver}(m, P_i, P_j, T; \delta) = \text{TRUE}$.
2. Unforgeability *of the proof for* $\text{SignedBy}(\delta, P_i)$*: 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* $\text{SignedBy}(\delta, P_i) = \text{TRUE}$.
3. Indistinguishability *of* $\texttt{ADVS-Sign}_{P_i}(m, P_j, T)$ *and* $\texttt{ADVS-Sign}_{P_j}(m, P_i, T)$*: Given* $\delta_i = \texttt{ADVS-Sign}_{P_i}(m, P_j, T)$ *and* $\delta_j = \texttt{ADVS-Sign}_{P_j}(m, P_i, T)$*, without the knowledge of* $T$*'s private key, no entity has non-negligible advantage in distinguishing the origins of* $\delta_i$ *and* $\delta_j$ *than random guess.*

Remark The definition of Indistinguishability implies that for any message $m$, both $P_i$ and $P_j$ can compute an ADVS signature with respect to each other. Moreover, no entity except $T$ is able to distinguish the ownership of these two signatures.

**Definition 5** *An accountable designated verifier signature scheme is* strong-secure *if, in addition to being* secure*, it ensures that a signing party* $P_i$ *cannot produce a proof for* $\text{SignedBy}(\delta, P_i) = \text{TRUE}$ *with non-negligible probability, where* $\delta = \texttt{ADVS-Sign}_{P_i}(m, P_j, T)$.

Note that the notion of strong-secure captures the aforementioned requirement of preventing the signer from proving the ownership of her generated signatures. This feature is not concerned in most signature schemes. However, it reflects the leak-freedom requirement in group signature settings.

## 3.4 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 with adequate efficiency. We leave it as an interesting open problem and discuss it in Section 4. 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 [37], which is, in turn, based on [31, 30, 41]. 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 $\langle y_T = g^{x_T}, x_T \rangle$, where $\mathbb{G}_q = \langle g \rangle$.

Thereby, $P_i$ can generate an accountable designated verifier signature on message $m$ for $P_j$ by presenting a proof of a statement [37] such as

$$\text{“}X \text{ is a } T\text{-encryption of ‘}g^i\text{’ AND I can sign } m \text{ as } P_i$$

$$\text{OR}$$

$$X \text{ is a } T\text{-encryption of ‘}g^j\text{’ AND I can sign } m \text{ as } P_j\text{”}$$

where $X$ is some value(s) 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 “$g^i$” 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 produce a similar proof for the same statement.

In order to facilitate practitioners, we give a concrete implementation of a secure `ADVS` scheme.

1. `ADVS-Sign`: $P_i$ generates an ElGamal encryption of $g^i \in \mathbb{G}_q$ using $y_T$; denote the ciphertext by $\langle A = g^i \cdot y_T^k, B = g^k \rangle$ where $k \in_R \mathbb{Z}_q$. Then it executes as follows:

   (a) Choose $r_1, r_2 \in_R \mathbb{Z}_q$ and compute $u = y_T^{r_1}$, $v = g^{r_1}$, and $w = g^{r_2}$.

   (b) Choose $d'_1, d'_2, c' \in_R \mathbb{Z}_q$ and compute $u' = y_T^{d'_1}(A/g^j)^{-c'}$, $v' = g^{d'_1}B^{-c'}$, and $w' = g^{d'_2}y_j^{-c'}$.

   (c) If $i \leq j$, compute $\theta = H(m\|A\|B\|u\|v\|w\|u'\|v'\|w')$,

   otherwise $\theta = H(m\|A\|B\|u'\|v'\|w'\|u\|v\|w)$, where $H : \{0,1\}^* \rightarrow \mathbb{Z}_q$ behaves like a random oracle, and $\|$ denotes the concatenation of binary strings.

   (d) Compute $c = \theta - c' \bmod q$.

   (e) Compute $d_1 = c \cdot k + r_1 \bmod q$ and $d_2 = c \cdot x_i + r_2 \bmod q$.

(f) If $i \leq j$, $P_i$'s accountable designated verifier signature on message $m$ with respect to $P_j$ is $\delta = (m, A, B, c, d_1, d_2, c', d'_1, d'_2)$; Otherwise, $\delta = (m, A, B, c', d'_1, d'_2, c, d_1, d_2)$.

2. `ADVS-Ver`: Given $\delta = (m, A, B, c, d_1, d_2, c', d'_1, d'_2)$, the verifier checks if:

$$c + c' = H(m \| A \| B \| y_T^{d_1}(A/g^i)^{-c} \| g^{d_1} B^{-c} \| g^{d_2} y_i^{-c} \| y_T^{d'_1}(A/g^j)^{-c'} \| g^{d'_1} B^{-c'} \| g^{d'_2} y_j^{-c'})$$

.

3. `ADVS-Proof`: Given a valid $\delta$, $T$ decrypts $(A, B)$ into $g^l$ such that either $l \in \{i, j\}$. $T$ then publishes a proof for $log_g y_T = log_B(A/g^l)$, which corresponds to the predicate SignedBy($\delta, P_l$). In plain words, $T$, having the decryption key for ElGamal ciphertext $(A, B)$, proves the clause that "X is a T-encryption of $g^l$". The proof of knowledge of ElGamal ciphertext content ensures $\delta$ is indeed generated by $P_l$. The details are straightforward and omitted.

The computational complexity for the signing party is 11 exponentiations (among them 6 exponentiations can be calculated using the implementation speedup technique [48]); the computational complexity for the verifier is 12 exponentiations (all of them can be calculated using the speedup technique).

We note that the `ADVS` signature scheme can be simulated without knowing any of $x_i$ and $x_j$ because we can first choose $c$ and $c'$, and then define $\theta$ to be $c + c' \mod q$. In order to prove security of `ADVS`, we need a slight variant of the Forking Lemma [50], which basically states that if there is an adversary that can forge a signature with a non-negligible probability, then there exists a polynomial-time algorithm that can produce two signatures with a non-negligible probability. Proof of the lemma is essentially the same as in [50], in what follows we highlights the difference caused by the difference between `ADVS` and the scheme in [50].

**Lemma 1** *If there is an adversary that can forge an* `ADVS` *signature with a non-negligible probability, then there exists a polynomial-time algorithm that produces two* `ADVS` *signatures with respect to* $\theta_1 \neq \theta_2$ *(and thus* $c_1 = \theta_1 - c' \mod q$, $c_2 = \theta_2 - c' \mod q$, *and* $c_1 \neq c_2$*).*

**Proof 1** *(sketch) Suppose the attacker asks* `a` *distinct queries to the random oracle, and* `b` *queries for obtaining* `ADVS` *signatures. The queries can be answered by the simulator mentioned above, which does not know the secret key* $x_i$ *of* $P_i$ *and* $x_j$ *of* $P_j$. *The simulation cannot get through when we encounter a "collision" in the* *definition of the random oracle. Note that the probability for a "commitment" (i.e., $m\|A\|B\|u\|v\|w\|u'\|v'\|w'$ in this case) happens to match the simulated output of the signing oracle is no more than the probability that $w$ happens to match the simulated output of the signing oracle, which is negligible in security parameter $\kappa$. Note that some $m\|A\|B\|u\|v\|w\|u'\|v'\|w'$, which the simulator outputs, appears in the list of queries made to the random oracle by the attacker is less than $\mathsf{2ab}/2^{\kappa}$. Moreover, the probability that some $m\|A\|B\|u\|v\|w\|u'\|v'\|w'$ is produced by the simulator twice is bounded by $\mathsf{b}^2/2^{\kappa}$. Combining them together and borrowing the remainder of the proof from [50], it can be shown that the lemma holds.*

**Theorem 1** *The above* `ADVS` *scheme is secure under the random oracle model and the Decision Diffie-Hellman assumption.*

**Proof 2** *(sketch) We show that the scheme satisfies the Definition 4.*

- Unforgeability *of* $\texttt{ADVS-Sign}_{P_i}(m, P_j, T)$. *Suppose there is a probabilistic polynomial-time algorithm $\mathcal{A}$ that does not know $x_i$ and $x_j$, and is nevertheless able to forge a $\delta_{ij} = \texttt{ADVS-Sign}_{P_i}(m, P_j, T)$ 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 discrete logarithm challenge instance as the pair of public and private keys of $(y_i, x_i)$ or $(y_j, x_j)$, with equal probability. Lemma shows that if $\mathcal{A}$ is able to forge a $\delta$, then $\mathcal{B}$ can obtain two accepting transcripts for the corresponding $\Sigma$-protocol with respect to two different $\theta$'s, which corresponds to two different $c$'s or two different $c'$. This means that either $x_i$ or $x_j$ can be extracted.*

- Unforgeability *of the proof for SignedBy$(\delta, P_i)$. Given a valid $\delta$ produced by either $P_i$ or $P_j$, suppose that there exists a probabilistic polynomial-time algorithm $\mathcal{F}$, that is able to forge a proof for SignedBy$(\delta, P_i)$ without knowing $P_i$ and $T$'s secret key.*

  *Case I: $\delta$ is indeed produced by $P_i$. Then we can construct a polynomial-time algorithm $\mathcal{D}$ to break the DDH assumption. Given a DDH challenge $(g, g_1, r, r_1)$, $\mathcal{D}$ simply sets $y_T = g_1$, $B = r$ and $A = r_1 \cdot g^i$. 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 over random guess.*

*Case II: $\delta$ is produced by $P_j$. This means that $(A = g^j \cdot y_T^k, B = g^k)$ for some $k \in \mathbb{Z}_q$; otherwise, Lemma allows us either to extract $x_i$ or to have two acceptable transcripts for $log_g B = log_{y_T}(A/g^j)$, both of which are contradictions. If $\mathcal{F}$ is able to produce such a proof that $log_g B = log_{y_T}(A/g^i)$, then Lemma shows that we can get two acceptable transcripts, which is impossible since the input is not in the language.*

- Indistinguishability *of* `ADVS-Sign`$_{P_i}(m, P_j, T)$ *and* `ADVS-Sign`$_{P_j}(m, P_i, T)$. *Without loss of generality, let $i \leq j$. Let* `ADVS-Sign`$_{P_i}(m, P_j, T)$ *be $(m, A, B, c, d_1, d_2, c', d_1', d_2')$ denoted by $\delta$, and* `ADVS-Sign`$_{P_j}(m, P_i, T)$ *be $(m, \bar{A}, \bar{B}, \bar{c}, \bar{d}_1, \bar{d}_2, \bar{c}', \bar{d}_1', \bar{d}_2')$, denoted by $\bar{\delta}$. It is straightforward to observe that the two distributions of $\delta$ and $\bar{\delta}$ are the same. First, due to the semantic security of ElGamal encryption, the distribution of $(A, B)$ and $(\bar{A}, \bar{B})$ are indistinguishable. Secondly, $c', d_1', d_2', \bar{c}', \bar{d}_1', \bar{d}_2'$ are random numbers independently selected from $\mathbb{Z}_q$. Since the hash function $H()$ is modeled as a random oracle, its outputs are random numbers. Thus, $c$ and $\bar{c}$ have a uniform distribution over $\mathbb{Z}_q$. Both $(d_1, d_2)$ and $(\bar{d}_1, \bar{d}_2)$ are also random as they are the sum of random numbers. Thus, the distributions of $\delta$ and $\hat{\delta}$ are identical.*

## 3.5 Leak-free Mediated Group Signature Scheme: Putting the Pieces Together

We are finally ready to present a concrete construction of group signatures satisfying all aforementioned requirements. As mentioned earlier, we use a system approach combined with the well-known *separation-of-duty* principle [29]. $\mathcal{GM}$ sets group policies, makes all decisions regarding group membership (admission/revocation), and performs the `OPEN` process on disputed group signatures, while $\mathcal{MS}$ enforces the policies and decisions.

The basic operation of our scheme is as follows. To sign a message $m$, a group member $\mathcal{U}_i$ presents an accountable designated verifier signature $\delta =$ `ADVS-Sign`$_{\mathcal{U}_i}(m, \mathcal{MS}, \mathcal{GM})$ to the mediation server $\mathcal{MS}$ thereby requesting a plain signature $\sigma =$ `Sign`$_{\mathcal{MS}}(m)$. The latter is treated as a group signature on the message $m$. The process of group signature verification is the same as `Ver`$_{\mathcal{MS}}(\sigma, m)$, except that the verifier has to check whether the certificate for $\mathcal{MS}$'s public key certifies $\mathcal{MS}$ as the mediation server for the involved group. Note that, since $\mathcal{GM}$ 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 $\sigma = \texttt{Sign}_{\mathcal{MS}}(m)$. The details of our construction are as follows.

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

1. The initialization of group manager $\mathcal{GM}$ includes the following:

   - It chooses a security parameter $\kappa$, based on which it chooses a discrete-logarithm based *crypto-context* including $\mathbb{Z}_p^*$ and its $q$-order subgroup $\mathbb{G}_q$, as specified in Section 3.2. The parameter $\kappa$ and the *crypto-context* are thus followed system-wide by group members, $\mathcal{MS}$, and $\mathcal{GM}$ itself.
   - It specifies a user set $\mathbb{U}$ so that each user or group member will be assigned with a unique identity $\mathcal{U} \in \mathbb{U}$.
   - It specifies (according to $\kappa$ and *crypto-context*) an accountable designated verifier signature scheme `ADVS` as in Section 3.3. In order for $\mathcal{GM}$ 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 information such as the time "user-id" joins or is revoked. This database is to keep record of all the current group members and ex-members who have been revoked.

2. The initialization of 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 $\kappa$ 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 $\mathcal{GM}$ 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 [57].
   - 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 $\langle$group-member-id, `ADVS`-signature, normal-signature$\rangle$, which will be explained below.

`JOIN`. Whenever the group manager $\mathcal{GM}$ 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 to play a role in the `ADVS` scheme, $\mathcal{U}_i$ 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 $\kappa$ and *crypto-context*.
2. $\mathcal{GM}$ inserts an entry $(\mathcal{U}_i, Y_{\mathcal{U}_i}, *)$ into its database DBUser-GM where "$*$" stands for any information $\mathcal{GM}$ 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 $\mathcal{GM}$ 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 $\mathcal{GM}$ decides to revoke the membership of a group member $\mathcal{U}_i$, the following protocol is executed.

1. $\mathcal{GM}$ records relevant information (e.g., when membership is revoked) in the "status" column corresponding to $\mathcal{U}_i$ in its database DBUser-GM.
2. $\mathcal{GM}$ informs $\mathcal{MS}$ that $\mathcal{U}_i$ should be revoked over the communication channel. For the revocation purpose, $\mathcal{MS}$ simply deletes the entry $(\mathcal{U}_i, Y_{\mathcal{U}_i})$ from its database DBMember-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 receiving $\delta$, $\mathcal{MS}$ 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. Otherwise, $\mathcal{MS}$ verifies $\delta$ 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 $\gamma$ will be treated as a group signature on message $m$. How to send the group signature $\gamma$ 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 $\gamma$ to the receiver.[1] Another option, which is not so elegant, is to let $\mathcal{MS}$ broadcast $\gamma$ so that $\mathcal{U}_i$ can get and resend $\gamma$ to the receiver via an anonymous channel.

3. When the security of DBSig-MS is not assumed, $\mathcal{MS}$ encrypts $(\mathcal{U}_i, \delta, \gamma)$ using ElGamal encryption algorithm under $\mathcal{GM}$'s public key $Y_{\mathcal{GM}}$.

For the clarity purpose, throughout the paper we still call $U_i$ the *originator* or the *signer* of group signature $\gamma$, despite the fact that $\gamma$ is actually computed by $\mathcal{MS}$.

`VERIFY`. Given $pk_{\mathcal{MS}}$ and a tag $\gamma$, anyone can verify whether $\gamma$ is a valid group signature by running Ver on inputs: $pk_{\mathcal{MS}}$, $m$, and $\gamma$.

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

1. $\mathcal{GM}$ sends $\gamma$ to $\mathcal{MS}$ via an authenticated communication channel.

2. On receiving $\gamma$, $\mathcal{MS}$ retrieves from its databases the encrypted $(\mathcal{U}_i, \delta, \gamma)$ and sends it to $\mathcal{GM}$ via the same channel.

3. After decryption of the entry, $\mathcal{GM}$ checks whether `ADVS-Ver`$(m, \mathcal{MS}, \mathcal{GM}; \delta)$ = TRUE. If not, $\mathcal{GM}$ aborts the OPEN procedure and determines that $\mathcal{MS}$ is faulty. Otherwise, $\mathcal{GM}$ 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.

## 3.6 Security Analysis

The correctness of our scheme can be verified by inspection. In the following, we first show that our construction satisfies three basic security requirements for group signatures, i.e., full-traceability, full-anonymity

[1] In this case, there may need a random delay to defeat traffic analysis, but such a delay exists in current anonymous channels.

and non-misattribution. Then, we show that our construction satisfies leak-freedom and supports immediate-revocation.

**Theorem 2** *The proposed group signature scheme is secure, in the sense of satisfying* full-traceability, full-anonymity *and* non-misattribution.

**Proof 3** *(sketch)*

(1) Full-traceability. *The proof of full-traceability is quite trivial. Let $\mathbb{C} \subseteq \mathbb{U}$ denote the set of participants that an adversary $\mathcal{A}$ compromised. In particular, we allow $\mathcal{A}$ to compromise all group members and the mediation server $\mathcal{MS}$.*

*Suppose that $\mathcal{A}$ is able to produce a signature $\gamma$ such that it is valid with respect to $\mathcal{MS}$' public verification key $pk_{\mathcal{MS}}$. When $\mathcal{GM}$ attempts to open $\gamma$, there are two possible cases:*

***Case I:*** *$\mathcal{GM}$ fails in Step 2 of the* `OPEN` *algorithm, i.e. no corresponding $(\mathcal{U}_i, \delta, \gamma)$ tuple is found. Note that our group signature scheme does not provide the adversary any advantage in attacking $\mathcal{SIG}$ since the computation of $\mathcal{SIG}$ is cryptographically independent of ADVS signatures. In other words, $\mathcal{SIG}$ is still secure against existential forgery. Thus, $\mathcal{GM}$ asserts that $\mathcal{MS}$ is the originator of $\gamma$.*

***Case II:*** *$\mathcal{GM}$ obtains the corresponding $(\mathcal{U}_i, \delta, \gamma)$ from $\mathcal{MS}$. Assume that the database is secure against unauthorized modifications, the* `OPEN` *process can identify the entity responsible for $\gamma$. $\mathcal{GM}$ first verifies $\delta$. If it is verified false, $\mathcal{GM}$ asserts that $\mathcal{MS}$ takes the responsibility since she violates the protocol. If $\delta$ is valid, $\mathcal{GM}$ executes* `ADVS-Proof` *to produce a proof for SignedBy($\delta$,$\mathcal{U}_i$). If SignedBy($\delta$,$\mathcal{U}_i$) is TRUE, $U_i$ is the originator. Otherwise, the signature $\delta$ is issued by $\mathcal{MS}$.*

*Therefore, combining **Case I** and **Case II**, $\mathcal{GM}$ is always able to trace the originator or a part of the originators of a correct group signature.*

(2) Full-anonymity. *The adversary $\mathcal{A}$ is challenged to identify the actual signer of a given group signature $\gamma$, which is generated by either $\mathcal{U}_i$ and $\mathcal{U}_j$. Let $\mathbb{C} \subseteq \mathbb{U} \cup \{\mathcal{MS}\}$ denote the set of entities that the adversary $\mathcal{A}$ has compromised after $\mathcal{A}$ is given its challenge $\gamma$. 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 $\mathcal{GM}$'s private key $X_{\mathcal{GM}}$.*

*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 [5] with respect to previous group signature schemes), we do not make assumptions on the security of DBSig-MS. We allow $\mathcal{A}$ to break into DBSig-MS. Nonetheless, penetration of DBSig-MS does not offer $\mathcal{A}$ any advantage in determining the identity of the signer. Recall that in Step 3 of the* `SIGN` *algorithm, all database entries are encrypted using ElGamal encryption under $\mathcal{GM}$'s public key. The semantic security of ElGamal encryption ensures that $\mathcal{A}$ does not even learn one bit information of its plaintext $(\mathcal{U}_l, \delta, \gamma)$, where $l \in \{i, j\}$. Therefore, the adversary is incapable of determining the actual signer even if she compromises all secret keys except $\mathcal{GM}$'s.*

(3) No-misattribution. *Given a group signature $\gamma$, let $(\mathcal{U}_i, \delta, \gamma)$ be the corresponding request entry in DBSig-MS. We prove below that $\mathcal{GM}$ cannot misattribute $\delta = \texttt{ADVS-Sign}_{\mathcal{U}_i}(m, \mathcal{MS}, \mathcal{GM})$ to group member $\mathcal{U}_j$, where $i \neq j$.*

*Without loss of generality, let $\delta = (m, A, B, c, d_1, d_2, c', d'_1, d'_2)$, where $A = g^i \cdot Y^k_{\mathcal{GM}}, B = g^k$. Suppose that $\mathcal{GM}$ can present a proof for SignedBy($\delta$,$\mathcal{U}_j$) = TRUE. The validity of the proof implies that $\mathcal{GM}$ has the knowledge of $\log_g Y_{\mathcal{GM}}$ and $\log_g Y_{\mathcal{GM}} = \log_B(A/g^j)$. Therefore, $A = g^j B^x$. Obviously, it contradicts to $A = g^i B^x$, which proves the non-misattribution property.*

Now we show that our group signature construction also satisfies two new requirements introduced earlier in this paper. It is summarized in the following theorem.

**Theorem 3** *The proposed group signature scheme fulfills* leak-freedom *and* immediate-revocation*.*

**Proof 4** *(sketch) It is quite trivial to observe that group members can be immediately revoked. Once $\mathcal{GM}$ updates $\mathcal{MS}$ the latest revoked member list, $\mathcal{MS}$ stops services for them instantly. Without the help from $\mathcal{MS}$, the revoked members are unable to produce any group signatures.*

*Given a group signature $\gamma$ issued by $\mathcal{MS}$ upon $\mathcal{U}_i$'s request $\delta$, our next goal is to prove that neither $\mathcal{MS}$ nor $\mathcal{U}_i$ is able to convince a third party $\mathcal{V}$, that $\mathcal{U}_i$ is the originator of $\gamma$.*

*We first show $\mathcal{MS}$'s inability in leaking information of $\mathcal{U}_i$. In order to convince $\mathcal{V}$, $\mathcal{MS}$ must show as the evidence that $(\mathcal{U}_i, \delta, \gamma)$ is stored in DBSig-MS . One may argue that $\mathcal{MS}$ may keep all randomness used in previous encryptions of DBSig-MS entries. Therefore it is easy for $\mathcal{MS}$ to prove, and reveal, that $(\mathcal{U}_i, \delta, \gamma)$ is* *indeed the plaintext of a DBSig-MS entry. Nonetheless, even if $\mathcal{MS}$ manages to achieve this, she is unable to convince $\mathcal{V}$ that $\delta$ is the original request generated by $\mathcal{U}_i$. This is in fact due to the* indistinguishability *of ADVS signatures. $\mathcal{V}$ is not certain whether $\delta$ is issued by $\mathcal{U}_i$ or by $\mathcal{MS}$.*

*Then, we show that it is also infeasible for $\mathcal{U}_i$ to leak the information of $\gamma$ even if she is the originator. As argued above, $\mathcal{U}_i$ has to prove that her signature $\delta$ is part of the plaintext of some entry in DBSig-MS. Consider the extreme case that $\mathcal{U}_i$ successfully breaks into DBSig-MS and luckily obtains the entry for $(\mathcal{U}_i, \delta, \gamma)$. Even if she knows that the entry is the ElGamal encryption of $(\mathcal{U}_i, \delta, \gamma)$, she is unable to prove it to anyone. Note that the proof of knowledge of an ElGamal encryption can only be done by the encrypter, which is $\mathcal{MS}$, and the one who has the decryption key, which is $\mathcal{GM}$. Thus $\mathcal{U}_i$ cannot construct a proof.*

*In summary, neither $\mathcal{U}_i$ nor $\mathcal{MS}$ is able to leak information of the identity of the originator of $\gamma$. It is obvious that other group member are unable to leak either which is evident from full-anonymity.*

## 4 Extension and Discussion

**Full-anonymity vs. leak-freedom**. The difference between full-anonymity and leak-freedom is quite subtle, yet important. They are geared to addressed two different security aspects of group signatures. Full-anonymity emphasizes on the resistance against external attacks, including key compromise, whereas leak-freedom constrains the behavior of insiders. An adversary might be able to compromise a group member or $\mathcal{MS}$'s secret key somehow. Nonetheless, such a security failure does not imply that the adversary obtains the victim's history state, including random numbers used in computation prior to attacks. Leak-freedom handles stronger attacks where the insiders could be adversary, who can easily record all previous states. Full-anonymity does not imply leak-freedom. Many existing group signatures satisfy full-anonymity, while their group members are able to leak the ownership of signatures. On the other hand, leak-freedom does not imply full-anonymity either. For example, a message authentication code is leak-free, but is not a group signature scheme and does not offer full-anonymity.

**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 potential incentives to cheat an outsider, which implies:

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

But what if it is known that a mediation server $\mathcal{MS}$, while not being trusted to preserve anonymity, does not always insert fake traffic into the network? Then an eavesdropping adversary still has a good chance to compromise the anonymity of some honest group members by simply conducting a traffic-analysis attack. Fortunately, this can be easily resolved by letting the mediation server choose a public key cryptosystem whereby the communication between any group member and $\mathcal{MS}$ is protected.

**On strong-secure `ADVS` vs. secure `ADVS`**. In our construction we utilized an `ADVS` that is *secure*, but not *strong-secure*. As a consequence, we assume that the storage of the mediation server $\mathcal{MS}$, particularly secrecy of the entries of $(\mathcal{U}_i, \delta = \texttt{ADVS-SIG}_{\mathcal{U}_i}(m, \mathcal{MS}, \mathcal{GM}), \texttt{Sign}_{\mathcal{MS}}(m))$ is ensured. This is necessary to avoid 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* `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 $\mathcal{MS}$.

**Robustness against denial-of-service attack**. Since the mediation server $\mathcal{MS}$ always needs to calculate modular exponentiations before it can discard an incoming illicit signature request, there is potentially the risk of denial-of-service attack for exhausting the resources of the $\mathcal{MS}$. We propose a simple solution that can substantially ease this concern. The idea is to let a group member $\mathcal{U}_i$ shares a unique symmetric key $w_i$ with the $\mathcal{MS}$. Each signature request from $\mathcal{U}_i$ must also be accompanied with a message authentication code (MAC) with respect to $w_i$, whose validity is then verified before $\mathcal{MS}$ verifies the validity of the accountable designated verifier signature. We remark that the introduction of message authentication code does not jeopardize the properties of our scheme, because $w_i$ is common to both $\mathcal{U}_i$ and $\mathcal{MS}$. (Clearly, no group signature scheme can be based on such common secrets; it is only for the purpose of robustness against denial-of-service attack.)

## 5 Conclusion

We identified two new properties, namely leak-freedom and immediate-revocation, that are necessary for a large class of group signature applications. We also constructed a practical scheme that achieves all traditional and newly-introduced goals by following a *system architectural approach*, which 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 investigation:

- 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 a time-stamping service, it is nevertheless conceivable that other alternatively constructions could fit well into different specific application scenarios.
- How to minimize $\mathcal{MS}$'s load? One possible venue is to construct a weaker form of mediation where the interaction with $\mathcal{MS}$ is not per-signature based. Another is to build a stateless $\mathcal{MS}$. This is not trivial since a trapdoor is needed to allow the group manager to open group signatures. A promising approach is to encrypt the state and embeds the ciphertext into group signatures.

**Acknowledgement**. We are grateful to the reviewers for their constructive comments that helped improve the paper.

Shouhuai Xu was supported in part by ARO, NSF, and AFOSR.

## References

[1] R. Anderson. Two remarks on public key cryptography, Invited talk on ACM CCS 1997.

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

[3] G. Ateniese, D. Song, and G. Tsudik. Quasi-efficient revocation of group signatures. In *Proceedings of Financial Cryptography'2002*.

[4] G. Ateniese and G. Tsudik. Some open issues and new directions in group signatures. In *Proceedings of Financial Cryptography'1999*.

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

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

[7] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of ACM Computer and Communications Security Conference (CCS)'93*, pages 62–73.

[8] J. Benaloh and D. Tuinstra. Receipt-free secret-ballot election (extended abstract). In *Proceedings of Annual Symposium on Theory of Computing (STOC)'94*.

[9] A. Bender, J. Katz, and R. Morselli. Ring signatures: Stronger definitions, and constructions without random oracles. In *Proceedings of Theory of Cryptography (TCC)*, 2006.

[10] D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In *Proceedings of Advances in Cryptology – CRYPTO '04*.

[11] D. Boneh, X. Ding, and G. Tsudik. Fine-grained control of security capabilities. *ACM Transactions on Internet Technology*, 4(1), February 2004.

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

[13] D. Boneh and H. Shacham. Group signatures with verifier-local revocation. In *Proceedings of ACM Computer and Communications Security Conference (CCS)'94*.

[14] X. Boyen and B. Waters. Compact group signatures without random oracles. In *Proceedings of Advances in Cryptology – EUROCRYPT '2006*.

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

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

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

[18] J. Camenisch and A. Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *Proceedings of Advances in Cryptology – EUROCRYPT '2001*, pages 93–118.

[19] J. Camenisch and A. Lysyanskaya. An identity escrow scheme with appointed verifiers. In *Proceedings of Advances in Cryptology – CRYPTO '2001*, pages 388–407.

[20] J. Camenisch and A. Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *Proceedings of Advances in Cryptology – CRYPTO '04*.

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

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

[23] J. Camenisch and E. van Herreweghen. Design and implementation of the *idemix* anonymous credential system. In *Proceedings of ACM Computer and Communications Security Conference (CCS)'02*.

[24] D. Chaum. Showing credentials without identification. In *Proceedings of Advances in Cryptology – EUROCRYPT '85*.

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

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

[27] D. Chaum and E. van Heyst. Group signatures. In *Proceedings of Advances in Cryptology – EUROCRYPT '91*, pages 257–265.

[28] L. Chen and T. Pedersen. New group signature schemes. In *Proceedings of Advances in Cryptology – EUROCRYPT '94*, pages 171–181.

[29] 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*.

[30] R. Cramer. *Modular Design of Secure yet Practical Cryptographic Protocols*. PhD thesis, University of Amsterdam, 1997.

[31] R. Cramer, I. Damgard, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Proceedings of Advances in Cryptology – CRYPTO '94*, pages 174–187.

[32] X. Ding, G. Tsudik, and S. Xu. Leak-free group signatures. In *Proceedings of IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2004.

[33] D. Dolev, C. Dwork, and M. Naor. Nonmalleable cryptography. In *Proceedings of Annual Symposium on Theory of Computing (STOC)'91*.

[34] T. ElGamal. A public-key cryptosystem and a signature scheme based on the discrete logarithm. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.

[35] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Proceedings of Advances in Cryptology – CRYPTO '86*, pages 186–194.

[36] Z. Galil, S. Haber, and M. Yung. Symmetric public-key encryption. In *Proceedings of Advances in Cryptology – CRYPTO '85*, pages 128–139.

[37] J. Garay, M. Jakobsson, and P. MacKenzie. Abuse-free optimistic contract signing. In *Proceedings of Advances in Cryptology – CRYPTO '99*, pages 449–466.

[38] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(1):270–299, 1984.

[39] S. Goldwasser, S. Micali, and R. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2), 1998.

[40] M. Hirt and K. Sako. Efficient receipt-free voting based on homomorphic encryption. In *Proceedings of Advances in Cryptology – EUROCRYPT '2000*, pages 539–556.

[41] M. Jakobsson, K. Sako, and R. Impagliazzo. Designated verifier proofs and their applications. In *Proceedings of Advances in Cryptology – EUROCRYPT '96*, pages 143–154.

[42] J. Katz. Efficient and non-malleable proofs of plaintext knowledge and applications. In *Proceedings of Advances in Cryptology – EUROCRYPT '2003*.

[43] A. Kiayias and M. Yung. Extracting group-signatures from traitor tracing schemes. In *Proceedings of Advances in Cryptology – EUROCRYPT '2003*.

[44] A. Kiayias and M. Yung. Group signatures with efficient concurrent join. In *Proceedings of Advances in Cryptology – EUROCRYPT '2005*.

[45] J. Kilian and E. Petrank. Identity escrow. In *Proceedings of Advances in Cryptology – CRYPTO '98*, pages 169–185.

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

[47] P. MacKenzie and M. K. Reiter. Networked cryptographic devices resilient to capture. In *Proceedings of the 2001 IEEE Symposium on Security and Privacy*, pages 12–25, May 2001.

[48] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of applied cryptography.* CRC Press, 1997. ISBN 0-8493-8523-7.

[49] M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *Proceedings of Annual Symposium on Theory of Computing (STOC)'90.*

[50] D. Pointcheval and J. Stern. Security proofs for signature schemes. In *Proceedings of Advances in Cryptology – EUROCRYPT '96*, pages 387–398.

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

[52] R. Rivest, A. Shamir, and Y. Tauman. How to leak a secret. In *Proceedings of Advances in Cryptology – ASIACRYPT '01.*

[53] C. P. Schnorr. Efficient identification and signatures for smart cards. In *Proceedings of Advances in Cryptology – CRYPTO '89*, pages 239–252.

[54] D. Song. Practical forward-secure group signature schemes. In *Proceedings of ACM Computer and Communications Security Conference (CCS)'01.*

[55] Y. Tsiounis and M. Yung. On the security of elgamal based encryption. In *Proceedings of International Workshop on Practice and Theory in Public Key Cryptography (PKC) '98.*

[56] G. Tsudik and S. Xu. Accumulating composites and improved group signing. In *Proceedings of Advances in Cryptology – ASIACRYPT '2003.*

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