attributes in A, u the number of undefined attributes in A. An *undefined attribute* is an abstract attribute that is only meaningful to the application subsystem by private agreements. Thus increasing u makes it more difficult for the individual DLA nodes to infer any hints from the audit trail fragments. It comes up with a similar effect by increasing c for more DLA nodes are required to recover a complete audit trail.

The logging confidentiality of GL is defined as

$$\wp_{log}(GL) = \frac{cu}{h}$$

The auditing confidentiality of the normalized auditing criteria $Q = SQ_1 \wedge \cdots \wedge SQ_t$ is defined as

$$\wp_{audit}(Q) = \frac{g+t-1}{s+t-1},$$

where s is the number of atomic auditing predicates, g the number of global auditing predicates, $(t-1)$ the number of conjunctive predicates in Q.

The query confidentiality on GL based on the auditing criteria Q is defined as

$$\wp_{query}(Q, GL) = \wp_{audit}(Q) \times \wp_{log}(GL)$$

The DLA confidentiality of the whole system is defined as the average of the query confidentiality for all queries.

$$\wp_{DLA}(GL) = \overline{\wp_{query}(Q, GL)}$$

8. CONCLUSION

Confidential logging and auditing is suitable for distributed logging and auditing of group acticities. Our scheme restrict DLA nodes from gaining complete information of the log trails, so that no single node can abuse their responsibility in the logging and auditing process. By proper mapping of logging functions to the e-coin cryptographical system, we can make the DLA's service roles unforgeable and undeniable.

REFERENCES

[1] *Office of the Auditor-General Computer auditing: Background.* 2001. http://www.agsa.co.za/About%20%20us/Corporate%20Information/AGSA%20Information%20Systems%20Auditing/memo/Transversal%20Systems.doc

[2] T. F. Lunt, "Automated Audit Trail Analysis and Intrusion Detection: A Survey." *Proceedings of the 11th National Computer Security Conference*, 1988.

[3] W. Lee, S Stolfo and K. Mok. *Mining Audit Data to Build Intrusion Detection Models.* 1998. http://www1.cs.columbia.edu/~sal/JAM/PROJECT/HTML-papers/KDD98-audit/Paper/kdd98.html

[4] M. Roger and J. Goubault-Larrecq, "Log Auditing through Model Checking." *In Press, I. C. S., editor, Proceedings of the 14th IEEE Computer Security Foundations Workshop (CSFW'01)*, pp. 220-236, 2001.

[5] S. Axelsson. *Intrusion Detection Systems: A Survey and Taxonomy.* Chalmers University of Technology: Goteborg, Sweden, 2000.

[6] L. Strous, "Audit of Information Systems: The Need for Co-operation." *SOFSEM '98: Theory and Practice of Informatics, 25th Conference on Current Trends in Theory and Practice of Informatics*, LNCS 1521, pp. 264-272, 1998.

[7] L. Jean Camp and J.D. Tygar, "Providing Auditing While Protecting Privacy." *The Information Society*, 10(1), pp. 59-72, 1998.

[8] N. Szabo, "Confidential Auditing." http://szabo.best.vwh.net/confidential.html

[9] M. Franklin and M. Yung, "Varieties of Secure Distributed Computing." *In Proceedings of Sequences II, Methods in Communications, Security and Computer Science*, pp. 392-417, 1996.

[10] A. C. Yao, "Protocols for secure computations." *In Proceedings of 2rd IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 160-164, 1982.

[11] O. Goldreich, , S. Micali, and A. Wigderson, "How to Play Any Mental Game or a Completeness Theorem for Protocols with Honest Majority.", *In Proceedings of the 19th ACM Symposium on Theory of Computing*, pp. 218 – 229, 1987.

[12] D. Chaum, C. Crepeau, and I. Damgard, "Multiparty Unconditionally Secure Protocols." *The 20th ACM Symposium on the Theory of Computing*, pp.11--19, 1988.

[13] M. Ben-Or, S. Goldwasser, and A. Wigderson, "Completeness Theorems for Non-cryptographic Fault-tolerant Distributed Computation." *In Proceedings of the 20th Annual ACM Symposium on Theory of Computing, STOC'88*, pp. 1-10, 1988.

[14] D. Beaver, J. Feigenbaum, J. Kilian and P. Rogaway, "Security with Low Communication Overhead." Crytpo'90, pp. 62-76, 1990.

[15] T. Rabin and M. Ben-Or, "Verifiable Secret Sharing and Multiparty Protocols with Honest Majority." *In Proceedings of the 21st Symp. on the Theory of Computing*, pp 73—85, 1989.

[16] D. Beaver and S. Goldwasser. Multiparty computation with faulty majority. In Advances in Cryptology: CRYPTO '89, pages 589-590, Berlin, Aug. 1990. Springer.

[17] R. Cramer, I. Damgård, S. Dziembowski, M. Hirt, and T. Rabin, "Efficient Multiparty Computations Secure Against an Adaptive Adversary." *Advances in Cryptology - EUROCRYPT '99*, LNCS 1592, pp. 311-326, 1999.

[18] M. Hirt, U. Maurer, and B. Przydatek, "Efficient Secure Multi-Party Computation." *Advances in Cryptology - ASIACRYPT '00*, LNCS 1976, pp. 143-161, 2000.

[19] R. Gennaro, M. Rabin, T. Rabin, "Simplified VSS and Fast-Track Multi-party Computations with Applications to Threshold Cryptography." *ACM PODC'98*, 1998.

[20] C. Clifton, M. Kantarcioglu and J. Vaidya, "Tools for Privacy Preserving Distributed Data Mining." http://www.acm.org/sigs/sigkdd/explorations/issue4-2/clifton.pdf

[21] S. C. Pohlig and M. E. Hellman, "An Improved Algorithm for Computing Logarithms over GF(p) and its Cryptographic Significance." *IEEE Trans. Inform. Theory IT-24*, pp. 106-110, 1978.

[22] W. Du, Z. Zhan, "A Practical Approach to Solve Secure Multi-Party Computation Problems." *In Proceedings of New Security Paradigms Workshop*, 2002

[23] R. Fagin, M. Naor and P. Winkler, "*Comparing Information without Leaking It.*" *Communication of the ACM*, 39, pp. 77-85, 1996

[24] M. T. Ozsu and P. Valduriez, *Principles of Distributed Database Systems*, Prentice Hall, 1999.

[25] B. Schneier and J. Kelsey, "Secure Audit Logs to Support Computer Forensics.", *ACM Transactions on Information and System Security*, 2(2), pp. 159-176, 1999.

[26] J. Benaloh and M. de Mare, "One Way Accumulators: A Decentralized Alternative to Digital Signatures." *Advances in Cryptology - Proceedings of Eurocrypt '93*, 1993.

[27] M. T. Goodrich, and R. Tamassia and J. Hasic, "An Efficient Dynamic and Distributed Cryptographic Accumulator." *In Proceedings of Information Security Conference (ISC 2002),* 2002

[28] B. Neumann and T. Ts'o, "Kerberos: An Authentication Service for Computer Networks," *IEEE Communications Magazine*, 32(9), pp 33-38, 1994.

[29] C. Kruegel, T. Toth and C. Kerer, "Decentralized Event Correlation for Intrusion Detection." *In International Conference on Information Security and Cryptology (ICISC)*, LNCS, 2001.

[30] T.C. Lam and Jyh-Charn Liu "On the Evidence based Peer-to-Peer Resource Management in Distributed Computing System" Technical Report 2003-7-2, Department of Computer Science, Texas A & M University, 2003.

Leak-free Group Signatures with Immediate Revocation

Xuhua Ding[*]
School of Information Systems
Singapore Management University
xhding@smu.edu.sg

Gene Tsudik
Department of Computer Science
University of California, 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 an interesting and appealing cryptographic construct with many promising potential applications. This work is motivated by attractive features of group signatures, particularly, their potential to serve as foundation for anonymous credential systems. We re-examine the entire notion of group signatures from a systems *perspective and identify two new security requirements:* leak-freedom *and* immediate-revocation*, which are crucial for a large class of applications. We then present a new group signature scheme that achieves all identified properties. Our scheme is based on the so-called* systems architecture approach. *It is more efficient than the state-of-the-art and facilitates easy implementation. Moreover, it reflects the well-known* separation-of-duty *principle. Another benefit of our scheme is the obviated reliance on underlying anonymous communication channels, which are necessary in previous schemes.*

1. Introduction

The concept of group signatures was introduced by Chaum and van Heyst [18] in 1991. The chief motivation was to facilitate anonymous group-oriented applications. Given a valid group signature, any verifier can determine the signer's group membership, while only a designated entity (called a group manager) can identify the actual signer. In recent years, many research efforts sought efficient constructs and precise definitions of group signatures. Early schemes (e.g., [19]) suffered from linear complexities of either (or both) group public key size or signature size with respect to the number of group members. These drawbacks were first addressed by Camenisch and Stadler [13]. Follow-on results [12, 1] gradually improve on efficiency. Despite these advances, membership revocation remained a difficult obstacle [7, 33, 2]. Significant progress was made by Camenisch and Lysianskaya [9] who utilized a novel dynamic accumulator to construct a group signature scheme with reasonably efficient revocation. Further improvements have been made recently by Tsudik and Xu [34].

In the past, group signature schemes aimed to satisfy a jumble of (often redundant and overlapping) security requirements: *unforgeability*, *exculpability*, *traceability*, *coalition-resistance*, *no-framing*, *anonymity*, and *unlinkability* [13, 8, 12, 29, 3, 9]. To untangle and simplify these messy requirements, Bellare et al. [4] investigated "minimal" security requirements for group signature schemes. This line of research is very important, as it parallels the pursuit of similar requirements for *secure* public key encryption schemes [26, 31, 32, 22] and *secure* key exchange protocols. The work in [4] showed that two security properties: *full-traceability* and *full-anonymity* are sufficient to subsume all aforementioned (seven) requirements.

In this paper, we argue that *full-traceability* and *full-anonymity* are insufficient for certain realistic group signature settings. This claim is based on the observation that group signatures are inherently application-oriented, and, thus, cannot simply be treated as a primitive in the bare model. We identify two new requirements: leak-freedom and immediate-revocation which are necessary for a large class of commercial applications. Informally, leak-freedom means that a signer cannot

[*] Work done at the University of California, Irvine.

[†] Work done, in part, at the University of California, Irvine.

1063-6927/04 $20.00 © 2004 **IEEE**

convince anyone[1] that it generated a given group signature. Whereas, immediate-revocation means that a group member's revocation results in immediate inability of generating group signatures, i.e., a revoked member cannot any further group signatures.

Why is leak-freedom important? One of the main purposes of group signatures is for an organization to conceal its internal structure. Suppose Alice is an employee of a company (say, ABC) authorized to sign purchase orders and one of the suppliers is another company XYZ. If Alice – without revealing her private key or any other long-term secret – can convince XYZ that she is the signer of a given purchase order, she could obtain illegal kickbacks from XYZ as "gratitude" for her supplier selection. This sort of *information leakage* illustrates potential abuses of group signatures, and justifies the introduction of the leak-freedom property to block such abuses.

Why is immediate-revocation important? We continue with the previous example: any purchase order signed by Alice (on behalf of her employer ABC) for a supplier XYZ imposes certain financial and/or legal responsibilities on ABC. However, if Alice is aware of her impending lay-off, she could, in collusion with XYZ, sign fraudulent purchase orders for ABC. (Note that this problem is less grave in traditional public key infrastructures where such "poisoned" signatures cannot be attributed to anyone other than the signer herself.) This sort of abuse is possible in all current group signature schemes, unless we make very strong assumptions about underlying communication channels [23]. To this end, immediate-revocation is necessary to block such abuses without having to introduce unrealistic assumptions.

1.1. Our Contributions

This paper makes two noteworthy contributions. First, as mentioned above, we identify two important new security properties: leak-freedom and immediate-revocation. Second, we construct a secure and practical group signature scheme that satisfies both new and existing security properties. The proposed is very efficient: only 11 exponentiations are needed to generate a group signature, while the cost of signature verification is equivalent to verifying a single plain (i.e., non-group) signature, such as RSA. These costs are appreciably lower than those of state-of-the-art group signature schemes [9], [34].

The new scheme also offers two important advantages over previous work:

- It removes the burden of having all signers and all verifiers constantly obtaining and maintaining up-to-date "state information" pertaining to current membership. In prior work, either all signers and all verifiers had to be aware of public revocation lists [7, 2], or all signers had to be aware of public accumulator ingredient lists [9, 34]. In the proposed scheme, the issue of dynamic group membership is fully transparent to both signers (group members) and verifiers.
- It relaxes the requirement for underlying anonymous communication channels, since a group member does not need to communicate directly with signature verifier(s). Whereas, anonymous communication channels are essential (although rarely mentioned explicitly) in previous group signature schemes.

Outline: the rest of this paper is organized as follows: the next section provides the definitions of group signatures and their desired security requirements. Section 3 presents the new group signature scheme. Section 4 discusses several extensions and Section 5 concludes the paper.

2. Definitions

Definition 2.1 *A* **group signature scheme** *includes the following components:*

`Setup:` *a probabilistic polynomial-time algorithm that, on input of a security parameter k, outputs the specification of a cryptographic context including the group manager's public key $pk_{\mathcal{GM}}$ and secret key $sk_{\mathcal{GM}}$.*

`Join:` *a protocol between the group manager $\mathcal{GM}$ and a user that results in the user becoming a group member $\mathcal{U}$. Their common output includes 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 that, on input of a group member's identity (and perhaps her public key $pk_{\mathcal{U}}$), outputs updated information that indicates the current state of the system after revoking the membership of a given group member.*

`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:` *an 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 $\mathcal{GM}$. It 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*

[1] Except the group manager who can anyway always identify a signer.

(e.g., a membership public key $pk_{\mathcal{U}}$ and a proof) that allows anyone to identify the actual signer.

We now specify the desired security properties of a group signature scheme. These properties are presented informally; a more formal specification appears in the full version of this paper [21]. We note that, although we use the notions of full-traceability and full-anonymity introduced in [4], the definitions of [4] have to be amended to accommodate dynamic groups, since [4] only considers static groups.

Definition 2.2 *A* **secure** *group signature scheme must have the following properties:* correctness*,* full-traceability*,* full-anonymity*,* no-misattribution*, as well as the newly introduced* leak-freedom *and* immediate-revocation*.*

Correctness*: any signature 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 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 colluding coalition.*

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

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

Leak-freedom*: it is computationally infeasible for a signer to convince anyone that it actually signed a given message, even if it possesses all other signers' secrets, except the group manager's secret for opening signatures.*

Immediate-revocation*: it is computationally infeasible for a group member revoked at time t to generate a valid group signature at any time $t' > t$.*

3. Our Construction

Unlike 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 main idea is the introduction of a new entity referred to as the *mediation server* ($\mathcal{MS}$). $\mathcal{MS}$ is an on-line partially trusted server that helps in signature generation and membership revocation.

Roughly speaking, the system functions as follows: each time a group member needs to generate a signature, it somehow "identifies" itself to the mediation server which then (if the member is not revoked) produces a normal signature. This normal signature is the actual group signature and may be delivered to intended verifier(s). Nevertheless, the mere introduction of the mediation server does not imply that we can trivially obtain a group signature scheme possessing all desired security properties.

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

3.1. Further Motivation

At this point, one might wonder whether a practical solution that satisfies all aforementioned requirements can be obtained in a trivial fashion. A trivial approach that satisfies all aforementioned requirements is to make the group manager an on-line entity and have it "filter" all group signature requests. Each group member has a private 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 signature key. The latter is then released to the verifier(s) and treated as a group signature. With this trivial solution all security properties (including leak-freedom and immediate-revocation) are satisfied and signature generation/verification costs are minimal.

However, the trivial approach requires constant ironclad security of the $\mathcal{GM}$. If this can be assured, then the trivial solution is perfect. However, having a **fully trusted on-line** entity is undesirable and sometimes simply unrealistic.[2] A on-line $\mathcal{GM}$ would be a single point of failure in terms 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 distributed cryptography, which usually takes a heavy toll in system complexity, including: management, computation and communication. To avoid unnecessary complexity and subtleties, we design a system under the guidance of the well-known separation of duty principle. (See the seminal work of Clark and Wilson [20] for necessary background.) This approach, as will be shown below, facilitates a practical solution with a similar flavor of distributed security, i.e., compromise of either $\mathcal{GM}$ or the newly introduced

[2] For much the same reasons that Certification Authorities (CAs) are not on-line.

mediation server $\mathcal{MS}$, but not both, does not imply complete compromise of the system.

3.2. Model

PARTICIPANTS: a set of group members $\mathbb{U}$ where $|\mathbb{U}| = n$, a group manager $\mathcal{GM}$ who admits group members, a mediation server $\mathcal{MS}$ who revokes group members (according to $\mathcal{GM}$'s instructions) 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 dynamic membership database DBMember-MS and $\mathcal{GM}$ maintains a similar dynamic membership database DBUser-GM. Moreover, we assume that $\mathcal{MS}$ maintains a database DBSig-MS that records all signature transactions. We further assume that this database is never compromised. In practice, various uncomplicated protection mechanisms can be utilized.[3]

COMMUNICATION CHANNELS: all communication channels are asynchronous and under adversary's control, except the channel between $\mathcal{GM}$ and $\mathcal{MS}$. In a typical system configuration, we do not assume any anonymous channels. However, we do assume that the channels between a group member $\mathcal{U} \in \mathbb{U}$ and $\mathcal{GM}$, and between $\mathcal{GM}$ and $\mathcal{MS}$ are authentic. (This can be implemented via standard techniques and is thus ignored in the rest of the paper).

TRUST: precise specification of the trust model is admittedly difficult mainly because of the introduction of the new party ($\mathcal{MS}$). Nevertheless, taking into account the *separation-of-duty* principle, we have the following:

1. $\mathcal{GM}$ 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 refuse services to any and all revoked group members; equivalently, to produce group signatures only for legitimate members. $\mathcal{MS}$ is assumed not to misbehave if such activity will be traced to it. Nonetheless, $\mathcal{MS}$ may attempt to: 1) frame an honest member into signing a message, 2) generate an unattributable group signature, and 3) compromise anonymity of an honest group member (e.g., via out-of-band means).

SECURITY DEFINITIONS: in order to accommodate the new entity ($\mathcal{MS}$), we need to slightly amend some parts of Definition 2.2 for our setting. The changes are minimal but necessary for the sake of clarity. (The rest of the definitions remains unchanged.)

Definition 3.1 Full-traceability*: the set of colluders can additionally include $\mathcal{MS}$.*
Full-anonymity*: the adversary is additionally allowed to have access to the secret key(s) of $\mathcal{MS}$.*
Leak-freedom*: it is infeasible for a signer to convince anyone (except $\mathcal{MS}$) that it generated a group signature; it is infeasible for $\mathcal{MS}$ to convince anyone (except $\mathcal{GM}$) that a certain group member generated a group signature.*

3.3. Accountable Designated Verifier Signatures

We introduce the notion of accountable designated verifier signatures (ADVS) that will serve as the building block in our group signature scheme. This notion is an enhancement of the private contract signatures introduced in [25]. 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 appointed by the signer. An **accountable** designated verifier signature scheme, on the other hand, emphasizes the trusted third party's capability to identify the actual signer of a valid signature.

Definition 3.2 *Let P_i and P_j be two distinct entities and T be a trusted third party. Suppose P_i is the signer and P_j is the verifier. An accountable designated verifier signature scheme,* ADVS*, is a tuple of polynomial-time algorithms (*ADVS-Sign*,* ADVS-Ver*,* ADVS-Proof*) defined as follows:*

ADVS-Sign: *an algorithm executed by P_i on message m to output an ADVS: $\delta = \texttt{ADVS-Sign}_{P_i}(m, P_j, T)$.*

ADVS-Ver: *an algorithm which allows P_j to verify the validity of an input tag δ on message m, such that:*
$$\texttt{ADVS-Ver}(m, P_i, P_j, T; \delta) = \begin{cases} \text{TRUE} & if\ \delta = \texttt{ADVS-Sign}_{P_i}(m, P_j, T) \\ \text{FALSE} & otherwise \end{cases}$$

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

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

Definition 3.3 *An ADVS scheme is* **secure** *if the following properties are satisfied:*

Unforgeability *of* $\texttt{ADVS-Sign}_{P_i}(m, P_j, T)$*: for any m, it is infeasible for any $P \notin \{P_i, P_j\}$ to produce δ, such that* $\texttt{ADVS-Ver}(m, P_i, P_j, T; \delta) =$ TRUE.

[3] For example, $\mathcal{MS}$ can use a *decoy* technique and insert $(n-1)$ well-formed dummy transaction records for each genuine one. Alternatively, the database can be kept encrypted with the encryption key protected using some advanced cryptographic techniques. These and other approaches and their respective merits are discussed in the full version of this paper [21].

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* $\texttt{ADVS-Ver}(m, P_i, P_j, T; \delta) = \text{TRUE}$.

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 any* $P \notin \{T, P_i\}$ *to produce a proof for* $\text{SignedBy}(\delta, P_i)$ *being* TRUE.

We need to further differentiate between the case that P_i is able to produce a proof for $\text{SignedBy}(\delta, P_i)$ and the case that P_i is unable to do so. Although the former is desirable in the context of private contract signatures, it is undesirable in our setting. This is because private contract signatures only intend to prevent P_j transferring the one bit information $\text{SignedBy}(\delta, P_i)$ by any means. Whereas, it would be very useful in our context for both P_i and P_j not to be able to leak the bit: $\text{SignedBy}(\delta, P_i)$. Thus, we have the following definition:

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

A *strongly-secure* ADVS scheme can allow us to easily construct a simple leak-free group signature system. Unfortunately, we do not know how to construct such a scheme, so we 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 [25]. Due to space limitation, the details appear in the full version of this paper [21].

3.4. Leak-free Group Signatures with Immediate Revocation

We are finally ready to present a concrete construction. The basic operation of our scheme is as follows. Given a message m, a group member $\mathcal{U}_i$ presents an ADVS $\delta = \texttt{ADVS-Sign}_{\mathcal{U}_i}(m, \mathcal{MS}, \mathcal{GM})$ to $\mathcal{MS}$ requesting (and obtaining) a plain signature $\sigma = \texttt{Sign}_{\mathcal{MS}}(m)$. The latter is viewed as a group signature, for which there is a single group-wide verification key. Note that, since $\mathcal{GM}$ plays the role of a trusted third party in the ADVS scheme, it can hold an actual signer accountable. We also note that our trust model implies that there are no issues with the fair exchange of $\delta = \texttt{ADVS-Sign}_{\mathcal{U}_i}(m, \mathcal{MS}, \mathcal{GM})$ for $\sigma = \texttt{Sign}_{\mathcal{MS}}(m)$, even if σ is returned to the user. Following the definition of group signatures, our mediated group signature scheme consists of the following procedures:

SETUP: this procedure initializes a group manager $\mathcal{GM}$ and a mediation server $\mathcal{MS}$.

1. $\mathcal{GM}$chooses a system wide security parameter κ. Based on it, $\mathcal{GM}$chooses a discrete-logarithm based crypto-context as in a normal Schnorr signature setting. The parameter κ and the crypto-context are followed system-wide by all group members, the $\mathcal{MS}$ and $\mathcal{GM}$ itself. $\mathcal{GM}$initializes a set $\mathbb{U}$ wherein each group member will be assigned a unique identity $\mathcal{U}_i \in \mathbb{U}$. $\mathcal{GM}$then instantiates an ADVS scheme ADVS as well as its own public/private key-pair: $\langle Y_{\mathcal{GM}} = g^{X_{\mathcal{GM}}}, X_{\mathcal{GM}} \rangle$. Finally, $\mathcal{GM}$initializes the database DBUser-GM.
2. The initialization of the mediation server $\mathcal{MS}$ consists of the following: choose a pair of ADVS public and private keys $\langle Y_{\mathcal{MS}} = g^{X_{\mathcal{MS}}}, X_{\mathcal{MS}} \rangle$ in the same crypto-context as in step 1. Choose a pair of keys for a normal digital signature scheme $\mathcal{SIG} = (\texttt{Gen}, \texttt{Sign}_{\mathcal{MS}}, \texttt{Ver}_{\mathcal{MS}})$ that is secure against adaptive chosen-message attack [27]. 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.[4] Initialize databases DBMember-MS and DBSig-MS.

JOIN: whenever $\mathcal{GM}$ decides to admit a new member, it first assigns a unique identity $\mathcal{U}_i$. $\mathcal{U}_i$ generates its ADVS public/private key-pair: $\langle Y_{\mathcal{U}_i} = g^{X_{\mathcal{U}_i}}, X_{\mathcal{U}_i} \rangle$. $Y_{\mathcal{U}_i}$ is then registered at the $\mathcal{GM}$ and $\mathcal{MS}$; both DBMember-GM and DBMember-MS are updated accordingly.

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

1. $\mathcal{U}_i$ sends $\mathcal{MS}$ an ADVS $\delta = \texttt{ADVS-Sign}_{\mathcal{U}_i}(m, \mathcal{MS}, \mathcal{GM})$ over an insecure public (and unauthenticated) channel.
2. Upon 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; otherwise, $\mathcal{MS}$ verifies δ by checking whether $\texttt{ADVS-Ver}(m, \mathcal{MS}, \mathcal{GM}; \delta) =$ TRUE. If it is, $\mathcal{MS}$ 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 is a group signature on message m.

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

[4] We assume that $\mathcal{MS}$ knows $sk_{\mathcal{MS}}$ in its entirety so as to prevent attacks due to incorrect system initialization. This can be ensured by utilizing techniques due to [35].

`REVOKE`: whenever $\mathcal{GM}$ decides to revoke membership of $\mathcal{U}_i$, it first updates DBUser-GM indicating that $\mathcal{U}_i$ is revoked and logs the necessary information. $\mathcal{GM}$ then informs $\mathcal{MS}$ that $\mathcal{U}_i$ is revoked. and $\mathcal{MS}$ proceeds to delete the entry $(\mathcal{U}_i, Y_{\mathcal{U}_i})$ from its database DBMember-MS. Consequently, all subsequent signature requests from $\mathcal{U}_i$ will be rejected by $\mathcal{MS}$.

`OPEN`: whenever $\mathcal{GM}$ decides to identify the actual signer of signature γ on message m, the following protocol is executed by $\mathcal{GM}$ and $\mathcal{MS}$:

1. $\mathcal{GM}$ sends γ to $\mathcal{MS}$ via an authenticated channel.
2. $\mathcal{MS}$ retrieves $(\mathcal{U}_i, \delta, \gamma)$ from its database and sends it to $\mathcal{GM}$ via the same authenticated channel.
3. $\mathcal{GM}$ checks whether $\texttt{ADVS-Ver}(m, \mathcal{MS}, \mathcal{GM}; \delta) =$ TRUE. If so, $\mathcal{GM}$ executes `ADVS-Proof` to produce a proof for SignedBy$(\delta, \mathcal{U}_i)$; otherwise, $\mathcal{GM}$ concludes that $\mathcal{MS}$ is cheating.

Remark: the above protocol does not specify how a group signature is sent to its intended verifier(s). There are two options. One way that allows us to completely get rid of the anonymous channels is to let $\mathcal{MS}$ send γ directly to the verifier(s). In this case, there may be a need for a random delay to address potential traffic analysis (see further discussion below). Note that this kind of delay already exists in the anonymous channels currently available. The other way is to let $\mathcal{MS}$ broadcast γ so that $\mathcal{U}_i$ can receive and re-send γ to the verifier(s) via an anonymous channel. We prefer the former, however, a detailed analysis is deferred to [21].

3.5. Security Analysis

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

A formal proof of this theorem appears in a full version of this paper [21]. Intuitively, the theorem holds because the final group signatures are generated using a standard signature algorithm, and `ADVS` makes it infeasible for the group members or $\mathcal{MS}$ to cryptographically prove the linkage between a signature request and the resulting group signature.

4. Extension and Discussion

Enhancing anonymity against traffic analysis. Our scheme does not assume that the channel between a group member $\mathcal{U}$ and the mediation server $\mathcal{MS}$ is authenticated or anonymous. This is because $\mathcal{MS}$ may have incentives to cheat an outsider, which, in turn, implies:

1. Even if the adversary eavesdrops 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.
2. $\mathcal{MS}$ can easily cheat an outsider by injecting a fake `ADVS` into the network or its database DBSig-MS.

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 has a good chance of compromising anonymity of some honest group members by means of traffic analysis. Fortunately, this can be avoided by using standard techniques, such as end-to-end encryption and traffic padding.

On strongly-secure `ADVS` vs. secure `ADVS`. In our construction we utilized an `ADVS` that is *secure*, but not *strongly-secure*. Consequently, we assume the secrecy of $\mathcal{MS}$'s storage, particularly DBSig-MS consisting of all signature transaction entries. This is necessary in order to avoid the following attack: If an attacker has access to an entry in DBSig-MS, then Alice can convince the attacker that she generated a group signature $\texttt{Sign}_{\mathcal{MS}}(m)$. Clearly, if a *strongly-secure* `ADVS` is utilized, we can achieve strictly stronger security that Alice is unable to convince XYZ that she generated a signature, even if XYZ has access to the corresponding entry in DBSig-MS. It seems non-trivial to ensure secrecy of DBSig-MS, while preserving other desirable properties. We continue investigating technical mechanisms that can replace this rather strong assumption regarding database secrecy [21].

Denial-of-Service (DoS) attacks. Recall than $\mathcal{MS}$ always performs a number of modular exponentiations before it is able to determine whether an incoming `ADVS` is valid. This opens the door for DoS attacks on $\mathcal{MS}$. To counter such attacks, we suggest a simple solution. The idea is to let each $\mathcal{U}_i$ and $\mathcal{MS}$ share a unique secret key w_i. Each signature request from $\mathcal{U}_i$ must also be accompanied by an authentication token computed over the request with the key w_i. When processing a request, $\mathcal{MS}$ verifies the authentication token before performing much more expensive validation of the `ADVS`. Note that this additional authentication is does not in any way influence security properties of our scheme. Of course, this is only a partial solution since an adversary can still mount a DoS attack on the $\mathcal{MS}$'s network interface.

4.1. Related Work

This paper can be viewed as one of many efforts pursuing practical and secure *group signature* or *identity escrow* schemes [18, 19, 13, 29, 1, 11], as well as anonymous *credential systems* [16, 17, 30, 10, 14]. Among them, the prior work most relevant to this paper is [11], which presented an identity escrow scheme (and a corresponding group signature scheme) with the appointed verifier property. The motivation was to obtain a scheme where a group member can only convince one or more appointed verifiers of its membership, while no other party can verify membership even if the signer cooperates fully.

Note that there is a difference between the appointed verifier property in [11] and the leak-freedom property specified in this paper. Specifically, the scheme in [11], by definition, allows a signer to convince the designated verifiers that it is authorized to conduct the relevant transactions. Cast this in our example scenario, Alice can always convince XYZ that she is authorized to sign purchase orders. This capability can result in the leakage (outlined in Section 1) we meant to avoid! Besides achieving the strictly stronger leak-freedom, our scheme is more efficient than [11] 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 [11], whereas, we achieve immediate-revocation which has only been explored in the context of traditional PKI-s [6].

A credential system is a system where users can obtain credentials from organizations and demonstrate possession of these credentials [16, 17, 30, 10, 14]. Chaum and Evertse [17] presented a general scheme using a semi-trusted TTP common to multiple organizations. However, their scheme is impractical. The credential system by Lysianskaya et al. [30] captures many of the desirable properties. Camenisch and Lysianskaya [10] presented a better solution with ingredients from a secure group signature scheme of [1]. The prototype implementation of [10] was done by Camenisch and van Herreweghen [14]. 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 its credentials with other parties) requires both signer and verifier to compute 200 exponentiations.

The notion of *abuse-freedom* or abuse-freeness [25] 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. Finally, we remark that leak-freedom is similar to the property called *receipt-freedom* or receipt-freeness in the context of voting schemes [5, 28], and its closely related variants called deniable encryption [15] and deniable proof [24].

5. Conclusion

We identified two properties, namely leak-freedom and immediate-revocation, that are crucial for a large class of group signature applications. We also constructed a scheme that achieves all of traditional and newly-introduced goals by following a *systems architectural approach*. Our scheme is practical and easy to implement, because it needs only 11 exponentiations for a group member to generate a group signature and one normal signature verification for its validation. Another contribution of this paper is the relaxation on the requirement for anonymous communication channels, which are essential in all previous schemes.

In addition to the already mentioned issues (including the protection of the server databases, and the robustness against denial-of-service attacks) that need further investigations, our work also incurs some problems that are interesting in an even more general context:

1. How to construct a practical *strongly-secure* ADVS scheme?
2. How to construct a leak-free group signature scheme with immediate revocation that does not rely 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.
3. How to achieve a stateless $\mathcal{MS}$? This is not trivial because, otherwise, the binding of an ADVS signature (or any other token with the desired properties) and a normal signature would allow $\mathcal{MS}$ to convince an outsider of the identity of the actual signer.

References

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

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

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

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

[5] J. Benaloh and D. Tuinstra. Receipt-free secret-ballot election (extended abstract). In *STOC*, 1994.

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

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

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