A Survey on Group Signatures and Revocation Methods

Joseph Malcolm & Ryan Pakbaz

Abstract— **A group signature is a primitive that allows a member of a group to sign a message on behalf of the entire group. Group signature schemes can be used in any application where the membership status of a user is important, but their identity is not, such as in keycard locks. Such schemes require active membership management in order to add and remove members from a given group, but current membership revocation systems either scale poorly with group size or place constraints on the revocation process that make them infeasible for practical use. In this paper, we explore established methods for implementing group signatures and a selection of revocation system for group signatures as well as current research which extends and enhances these methods.**

I. Introduction

In today's digital world, the importance of secure information exchange becomes more important and necessary every day. The concept of group signatures aims to offer a solution to this growing problem. Group signatures have a wide variety of applications ranging from verifying user access rights via keycards to providing a secure environment for anonymous payment schemes and voting protocols. Group signatures allow users to anonymously sign on the behalf of a group while still allowing user signatures to be traced back to individuals in the case of misuse.

A member of a group can sign a message on behalf of the entire group. Only valid members of the group can sign the message, and any non-member attempting a group signature will result in an invalid signature. This means that a group signature scheme needs some means of detecting whether or not a message was signed by a valid group member. [1]

The process of signing a message on behalf of the group is anonymous, and this is an important trait of group signatures. A member signs a message on behalf of the group, but the identity of the user who signed said message is not publicly announced. Any normal member of the group can only verify that the message was signed by a valid member of the group. This does not mean that there is no user identity associated with the message, but instead that the user identity is encrypted for the sake of identifying the validity of that user's membership status.

Such groups needs some method of adding new members as well as revoking the membership status of members in order to dynamically manage the membership of the group. These group administration tasks are handled by the group manager, which has an elevated privilege level over the other members of the group. The group manager is the only member of the group capable of revoking the anonymity of a signature in the case of a dispute or in a situation where a particular member has abused somehow abused their membership status. This is why the identity of the user is still stored in the signature, albeit in an encrypted fashion.

When a user's membership is revoked, there needs to be a mechanism to indicate that what was once a valid member is no longer valid. The problem here is that by nature, this will require some form of log-keeping in order to change the permissions of the group so that the revoked user is no longer able to sign messages. In some schemes, this is reflected in the signatures themselves storing a list of revoked users. In others, the group public key itself somehow stores a list of revoked users. There are also cases where this is reflected in updating a public key and pushing that update to every user in the group whenever a revocation occurs, which can result in a large amount of computations performed across the group as a whole per each revocation. As a result, the general problem with revocation methods lies in the fact that all revoked users need to be remembered somehow and signatures need to be checked against this set of revoked users, which results in a tradeoff between space complexity and time complexity in signing, verifying, and revoking.

In the following section, we examine a selection of four papers which offer novel solutions for increasing the efficiency and security of group signatures. Another common point of interest we will examine are the different methods the authors use for revoking group members which becomes a difficult problem to manage as group size grows to less manageable sizes.

II. A Practical and Provably Secure Coalition-Resistant Group Signature Scheme

In reference [2], we are introduced to a method developed to create group signatures that are coalition-resistant under the strong RSA assumption which implies that the RSA problem is difficult even when an attacker is able to chose a public exponent. Under this assumption, even the entire group working together cannot generate a valid signature that the group manager cannot link. This work also improves upon previous join functions by increasing efficiency by an order of magnitude, and it is also statistically zero-knowledge with respect to the group members secrets whereas normally, the join function requires the group member to expose the product of their secret a prime of special form, and a random prime.

Joseph Malcolm and Ryan Pakbaz are with the Department of Electrical and Computer Engineering, University of California, Santa Barbara, CA 93106. E-mail: {*joseph_malcolm*,*rpakbaz*}@umail.ucsb.edu

The authors also propose a simple modification that can allow their work to be applied towards an escrow scheme. They suggest a simple change to the "sign" and "verify" functions by creating a protocol between a verifier that is derived from the "sign" function.

To keep track of revoked members and to prevent and/or track unauthorized signings, the authors utilize a trivial approach where members signatures can be compared to records in a membership table. This table grows linearly with the number of members in the group and can become difficult to manage over a long period of time.

III. Revocation and Tracing Schemes for Stateless Receivers

Reference [3] implements a number of improvements and enhancements over prior work. This includes a framework for algorithms called "Subset-Cover" which provides a seamless integration between the revocation and tracing functions such that the revocation algorithm does not need to be changed. This mechanism consists of the following three parts; (1) The initiation scheme where users are assigned their secret key, (2) the broadcast algorithm where a message is broadcast which contains all revoked users, and (3) a decryption algorithm that all non-revoked users can use to decrypt messages.

The authors present two incarnations of the "Subset-Cover" method, each with its own trade-offs.

Method	Msg. Len.	Storage	Time
Complete Subtree	$rlog\frac{N}{r}$	$logN$	$O(loglogN)$
Subset Difference	$2r-1$	$\frac{1}{2}log^2N$	$O(logN)$

TABLE I
Performance comparison of Complete Subtree and Subset Difference for "Subset-Cover" Method

Depending on the number of users, N, and the message length, r, the appropriate method can be chosen to maximise resources and processing time.

In the complete subtree method as illustrated in fig. 1, all users are assigned a leaf in a binary tree and all revoked users (illustrated in black) are part of their own subtree. The subset difference method aims to reduce partition size of the tree by increasing the number of subsets of non-revoked members into up to $2r-1$ subsets which reduces message length by a factor of $logN$. By using trees to represent current and revoked members, additional sets of users can be easily revoked by marking their parent nodes.

IV. Efficient Revocation in Group Signatures

This particular revocation method is built on top of a Camenish-Stadler group signature scheme [4]. The Camenish-Stadler group signature scheme uses ElGamal in order to encrypt its signatures. When a user signs a message in a group, that user's identity is encrypted using ElGamal with respect to the group manager's public key (h, R). As a result of this, the group's manager can determine the identity of the user who had sent a given message

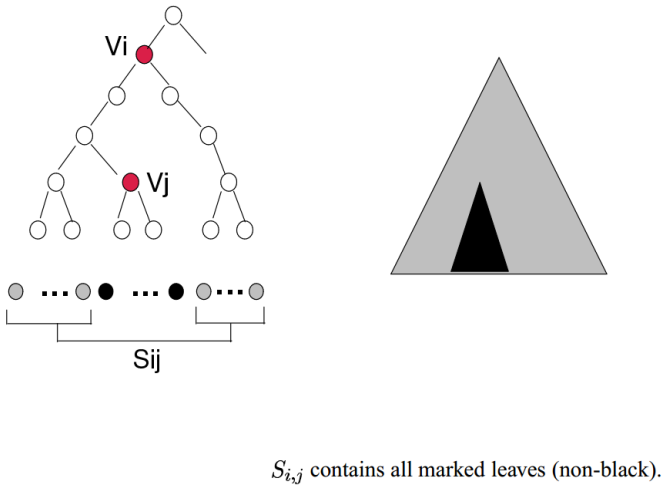

Fig. 1. An example of the complete subtree method in [3]

simply by decrypting the ciphertext sent by said user by using the group manager's private key.

They update this particular group signature scheme by modifying the mechanism used to verify whether the message was sent by a valid member of the group. In its original form, the Camenish/Stadler group signature scheme uses a zero-knowledge proof to prove the validity of a signature. The ciphertext signature provides two values (d, z), which are both generated using the value r, which is a private knowledge among all members of the group. The values (d, z) can be used for checking signature validity by allowing the recipient to calculate (d, z) using their known r value and then comparing the (d, z) value generated by this calculation against the values received through the signature. If the two sets of (d, z) match, then the signature has been proven to be valid, but if they do not match, then the signature is not a valid signature from a member of the group.

The zero-knowledge proof mechanism is modified to allow it to not only check the validity of the signature, but to also check against a list of revoked users in order to identify whether the signature was sent by an individual on the list of revoked users. This is done by including extra values in the ciphertext corresponding to users that have been revoked from the group. For each revoked user, there is a witness value t, which is calculated using the following formula.

$$t_i = (z/z_i)^r$$

In this formula, z is the identity of the user sending the message, and z_i is the identity of a revoked user. This formula will either return one if the identity of the user signing a message matches that of a revoked user or it is nonzero otherwise. Since there are a number of t values equal to the number of revoked users, that means that a user whose signature has no t entries equal to one is in theory an non-revoked user. The recipient of a signature

will be able to calculate the t values associated with the message and check them against those included in the signature. Then, if the t values from the calculated and sent sets match and none of the t entries are equal to one, then that means the message was not sent from a revoked user. If, however, this is not true, then the signature did not come from a valid non-revoked user.

This method of revocation has the distinct advantage of not requiring any changes in the group public key when a user is revoked. This is advantageous because not only are there no public key changes that need to be propagated to every user in the group, but also none of the signatures made in prior to the revocation are affected in any way. This means that past signatures are not compromised or otherwise invalidated by the process or revocation. Unfortunately, this comes at the cost of giving every public signature a space complexity of O(r), where r refers to the number of users revoked from the group [5]. In general, this implementation is only truly useful in groups that do not expect to have many revocations since the cost of signing scales poorly with the number of users revoked.

V. Dynamic Accumulators and Application to Efficient Revocation of Anonymous Credentials

This particular revocation method relies on the use of a dynamic accumulator. For the purposes of this application, they defined a dynamic accumulator as a function or family of functions that provides efficient generation, efficient evaluation, quasi-commutativity, witnesses, security, efficient deletion, and efficient provability. The following properties are particularly important for use in revocation.

- Efficient evaluation, so that the result of adding an element to the accumulator is efficient enough for practical use.
- Quasi-commutativity, so that a set of values added to the accumulator will produce the same result, regardless of what order said values are added.
- Security, so that values added to an accumulator cannot feasibly be recovered by an adversary without explicit knowledge of a private value.
- Efficient deletion, so that entries can be removed as well as just added to the accumulator, thus allowing for dynamic management.
- Efficient provability, so that the presence (or lack thereof) of a secret value can be determined.

The specific accumulator used in [6] is based on a Baric and Pfitzmann accumulator, but with some modifications. First, the domain of values that can be accumulated by the accumulator must consist solely of prime numbers. This is necessary since the revocation method relies on being able to exactly identify each element that has been added into the accumulator, and using only primes means that every value that is accumulated can easily be checked or removed with an inverse operation. Second, as alluded to previously, elements can be not only added to the accumulator but deleted from it as well. Third, the acts of deleting users and updating witnesses have a reduced time complexity. Finally, since there needs to be a way of checking whether a message signer is or is not a revoked user while maintaining anonymity, there is also a zero-knowledge proof added in order to obtain membership knowledge.

The accumulator in this case is based on RSA. Each element that is added to the accumulator is added through modular exponentiation, and each element that is removed is removed through the inverse of the modular exponentiation. For example, adding and removing elements in the accumulator is performed using the following equations.

$$v_{new} = (v_{old})^{x} mod(n)$$

In the above equation for adding to the accumulator, v_old refers to the accumulator prior to adding the value x, v_new refers to the accumulator after x has been added, and x is the value to be added. For the purposes of revocation, the value x is a prime number used to identify a particular member of the group, so adding that ID to the accumulator means that particular user has had their group membership revoked. The modulus value n is the product of p and q, which are both prime numbers. The equation for removing from the accumulator shown below uses the same basic variable as those used in the addition operation, but this particular equation makes use of p and q, and is why the selection for p and q is important.

$$v_{new} = (v_{old})^{x^{-1} mod((p-1)*(q-1))} mod(n)$$

Dynamic accumulators as tools for revocation have some very nice properties, such as the fact that signing and verifying messages has a time complexity of O(1). This is because the accumulator itself is only every modified or extensively manipulated during the actual revocation process. Unfortunately, this does come at the cost of needing to constantly track all revocations made throughout the group. When a revocation occurs, every user must update their membership by adding the revoked user to their accumulator. This cost is at its worst when a new user is added to the group, since that user will have to accumulate every single revocation prior to their initiation into the group. As a result, revocation operations have at worst an O(r) time complexity, where r refers to the number of users whose memberships have been revoked [5]. In general, this is a greedy revocation method, since it makes normal operation quick to execute, but makes the revocation itself expensive due to the logging that all group members must perform.

VI. Conclusion

All current revocation implementations scale poorly in a two common cases, either as group sizes grows or as the number of revoked users grows. For example, [4]s implementation scales poorly as the number of revoked users

grows because every signature must append the list of revoked users onto it in the form of additional witnesses. The dynamic accumulator approach in [6] prevents the signatures themselves from growing in size, but does so at the cost of requiring every user to update their membership after every revocation, showing that space complexity on the signature can be decreased, but in this case with the tradeoff of having an increased time complexity associated with the act of revocation. Each of the papers reviewed in this survey offer promising developments, however there still seem to be no solutions that are feasible for use in large groups with large numbers of revoked users.

References

[1] David Chaum and Eugne Heyst, "Group signatures," in *Advances in Cryptology EUROCRYPT 91*, DonaldW. Davies, Ed., vol. 547 of *Lecture Notes in Computer Science*, pp. 257–265. Springer Berlin Heidelberg, 1991.

[2] Giuseppe Ateniese, Jan Camenisch, Marc Joye, and Gene Tsudik, "A practical and provably secure coalition-resistant group signature scheme," in *Advances in Cryptology CRYPTO 2000*, Mihir Bellare, Ed., vol. 1880 of *Lecture Notes in Computer Science*, pp. 255–270. Springer Berlin Heidelberg, 2000.

[3] Dalit Naor, Moni Naor, and Jeff Lotspiech, "Revocation and tracing schemes for stateless receivers," in *Advances in Cryptology CRYPTO 2001*, Joe Kilian, Ed., vol. 2139 of *Lecture Notes in Computer Science*, pp. 41–62. Springer Berlin Heidelberg, 2001.

[4] Emmanuel Bresson and Jacques Stern, "Efficient revocation in group signatures," in *Public Key Cryptography*. Springer, 2001, pp. 190–206.

[5] Benot Libert, Thomas Peters, and Moti Yung, "Group signatures with almost-for-free revocation," in *Advances in Cryptology CRYPTO 2012*, Reihaneh Safavi-Naini and Ran Canetti, Eds., vol. 7417 of *Lecture Notes in Computer Science*, pp. 571–589. Springer Berlin Heidelberg, 2012.

[6] Jan Camenisch and Anna Lysyanskaya, "Dynamic accumulators and application to efficient revocation of anonymous credentials," in *Advances in Cryptology CRYPTO 2002*, Moti Yung, Ed., vol. 2442 of *Lecture Notes in Computer Science*, pp. 61–76. Springer Berlin Heidelberg, 2002.

VII. Appendix

A. Basic Functions of Group Signatures

Most group signature schemes, at their most basic, have the following functions in common. The papers discussed in this survey attempt to alter these functions in such a way that they retain this defined functionality while also improving upon them.

`SETUP: An algorithm for generating the initial group public key Y.`

`JOIN: A protocol between the group manager and a user that results in the user becoming a new group member`

`SIGN: A protocol between a group member and a user whereby a group signature on a user supplied message is computed by the group member`

`VERIFY: An algorithm for establishing the validity of a group signature given a group public key and a signed message`

`OPEN: An algorithm that given a signed message and a group secret key determines the identity of the signer`

B. Zero-Knowledge Proofs

Zero-knowledge proofs, which are sometimes also referred to as signatures of knowledge, are a signature mechanism that proves whether or not the sender of a message knows a secret value without revealing said secret value [4]. These are commonly used in group signature schemes due to the fact that they can verify the validity of a signed message without revealing the identity of the sender, thus allowing for the anonymous message signing that characterizes group signatures. They can also be useful as a tool used for revocation purposes, as will be demonstrated in [4] and [6].