A New Paradigm for Group Cryptosystems Using Quick Keys

Amitabh Saxena and Ben Soh
Department of Computer Science and Computer Engineering
La Trobe University
Bundoora, VIC, Australia 3083

Abstract- **In this paper we introduce a new approach to group key agreement. Our approach is based on the idea of an Associative One Way Function (AOWF). We illustrate how such functions can be used to perform highly dynamic and fully contributory multiparty key agreement in group-oriented cryptosystems. We also show how such schemes could be used to create efficient group digital signature schemes. Since at present, we have no working examples of AOWFs, the protocols proposed here only have theoretical value. A similar scheme was also discussed in [1] and our work is an extension to it.**

1. INTRODUCTION

The growth of the Internet has made it clear that in the future, computing environments will be highly collaborative and distributed in nature and will make extensive use of the Internet multicasting infrastructure. In this regard, two separate issues have emerged. Firstly we require a robust and efficient multicast infrastructure that scales well to the growing need. Secondly, security requirements dictate that multicast messages be protected from outsiders. In a highly dynamic environment where group membership is fairly short-lived, these two requirements often contradict each other (for example to achieve efficiency multicast messages may be sent via insecure channels, possible to unauthorized recipients). The research in group-oriented cryptosystems is therefore mainly focused on approaches to achieve efficient and scalable key agreement in Dynamic Peer Groups (DPGs) [2, 3].

All group key agreement protocols can be broadly classified into three classes depending on how the shared key is obtained [4, 5]: (a) Contributory, where all group members jointly contribute to generate a shared key, (b) Centralized, where a central controller maintains pair-wise secure channels with each group member and is responsible for generating the shared key and (c) Distributed, where there is no fixed controller but any one of the host is selected as a controller when needed. We believe that in a highly distributed environment, such as the Internet, the use of a key controller is a disadvantage. Current contributory approaches, however, have their own set of drawbacks [4, 5, 6].

In this paper, we discuss the possibility of achieving fully contributory key agreement without these drawbacks. The rest of the paper is organized as follows: In section 2, we discuss the issue of 'active' vs. 'passive' interactions in multiparty protocols. In section 3, we discuss the possibility of key agreement protocol using the idea of Associative One Way Functions (AOWFs). In sections 4-7, we describe multiparty protocols based on AOWFs that are passive and fully contributory.

2. ACTIVE VS PASSIVE PARTICIPATION

All multiparty key agreement protocols are interactive. In the context of group communication, passive participation implies asynchronous interactions while active participation implies synchronous (or ordered) interactions between group members. As a general rule, if any group interaction requires two or more ordered one-to-one messages between individual members then it is an active interaction. It is easy to see that active participation tends to increase communication overhead since it gets more complicated to synchronize group messages in a highly dynamic environment.

In a contributory public key system, all users who can possibly form a group have public-private key pairs. Each view of the group contains only a subset of the members and is associated with a different group key generated using a combination of the public and private keys of the active members. As an example of this, the original two-party Diffie-Hellman key exchange can be extended to generate a shared group key. This protocol is known as Group Diffie-Hellman (or simply GDH) and is described in [2, 6].

3. THE CONCEPT OF QUICK KEYS

A major drawback of GDH is the number of ordered interactions increase linearly with the number of members, making it unsuitable for large dynamic groups. We thus say that GDH requires active participation of all members in each key computation.

The model could be considerably improved if the group key could be computed with only passive participation from every member. In other words, members would be able to compute independently and quickly the shared group key without any one-to-one interactions. We will use the general term *Quick Keys* to refer to any such group-keying scheme (because the key computations can be performed fairly quickly, compared to GDH). We discuss this in detail in the next section.

Our construction of Quick Keys uses the idea of an associative function that is difficult to invert. For our purposes, a quasi-one way function discussed in [7] that is associative will also work. Rabi and Sherman also proposed the use of Associative One Way Functions (AOWFs) in multiparty key agreement in [1]. We extend their idea to include non-commutative AOWFs to construct Quick Keys.

0-7803-7788-5/03/$17.00 ©2003 IEEE.

4. PROPERTIES OF QUICK KEYS

Before we can describe Quick Keys, we need to discuss the security requirements for a multiparty key agreement scheme. We assume that group members have to share a common secret (the *group key*) in order to participate in the group session. Therefore the problem of securing multicast messages in a large dynamic group with frequent membership changes essentially reduces to the problem of enabling only the 'right' recipients to compute this group key efficiently and independently, since every such change requires members to recompute the group key.

4.1 Multiparty Key Agreement: Ideal Case

We consider the case where membership changes are frequent (about 1 per minute) and the size of the group is large (up to a few thousand members). Also unlike ordinary multicast groups, there is no fixed sender; every authorized recipient can also be a sender. This could be compared to a radio channel at a specific frequency where anyone with a transmitter and receiver can send and receive messages. To protect messages in such an environment, we would require a *broadcast* encryption scheme that would enable only a specific subset of the recipients to 'understand' the transmitted message. One possible way to achieve this is to encrypt every transmission with a key and ensure that only the authorized recipients know the corresponding decryption key. In this paper, we focus on this approach. We further assume that group messages are encrypted using a symmetric key system. Thus all authorized senders and recipients have to share a secret 'group' key at times. We observe that every membership change requires a new group key to be agreed upon. From a cryptographic point of view, the following properties are essential for this group key [5]:

1) Key Freshness: A group key should never be reused.

2) Key independence: Successive group keys should be independent. Given any set of successive group keys, it should not be possible to compute the next or the previous group key.

3) Key Secrecy: For authorized members, computing the group key should be easy. For non-members, this task should be computationally hard.

Additionally the following properties are desirable:

4) The group key should be fully contributory. This means that the group should not rely on a central controller for any re-keying operations.

5) The number and size of messages required for re-keying the group should be small. The re-key messages should be sent over one broadcast or multicast channel rather than individual one-to-one channels. Moreover these re-key messages have to be in plain text.

6) Ideally only one 'short' non-secret re-key message (possibly of the same size as the group key) broadcast to all users of the system should enable only the authorized members to compute the group key independently and quickly while computation of the same key by unauthorized recipients should be infeasible.

Many schemes (including the GDH discussed above) satisfy properties 1-4. Some schemes also address problem 5 and have partial solutions where there is compromise between the size of messages, their number or the complexity of the network [3, 4, 5]. Condition 6 is the most difficult to fulfil since the re-key message is non-secret and is multicast to unauthorized recipients. We argue that condition 6 cannot be satisfied unless there is some prior contribution from the authorized recipients to this re-key message. Those members must either passively or actively contribute to the non-secret message exchanged. In the following sections, we develop a model for multiparty key agreement by proposing two protocols and show how this key exchange can be done using only passive contribution from members provided certain conditions are satisfied. We say that a 'Quick Key' system should satisfy conditions 1-6 above.

4.2 Quick Keys Using Associative One-Way Functions.

We will now describe an approach to construct Quick Key schemes using a special type of one-way function. For this section, we assume the existence of a binary function, $\oplus$ over a set $\boldsymbol{S}$ of finite elements with the following properties:

1) For any $\boldsymbol{a}, \boldsymbol{b} \in \boldsymbol{S}$, we have, $\boldsymbol{a} \oplus \boldsymbol{b} \in \boldsymbol{S}$ (closure).

2) For any $\boldsymbol{a}, \boldsymbol{b} \in \boldsymbol{S}$, $\oplus$ is polynomial-time computable. (i.e. computing $\oplus$ is 'easy').

3) $(\boldsymbol{a} \oplus \boldsymbol{b}) \oplus \boldsymbol{c} = \boldsymbol{a} \oplus (\boldsymbol{b} \oplus \boldsymbol{c})$ for all $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c} \in \boldsymbol{S}$ (i.e. $\oplus$ is associative).

4) If $\boldsymbol{S}$ is large, given any $\boldsymbol{c} \in \boldsymbol{S}$, it is computationally intractable to find pairs $\boldsymbol{a}, \boldsymbol{b} \in \boldsymbol{S}$ such that $\boldsymbol{c} = \boldsymbol{a} \oplus \boldsymbol{b}$.

5) If $\boldsymbol{S}$ is large, inverting $\oplus$ is computationally intractable even if one of its arguments is given. In other words, given $(\boldsymbol{a} \oplus \boldsymbol{b})$ and $\boldsymbol{a}$, it is not be possible to compute $\boldsymbol{b}$ for most $\boldsymbol{a}, \boldsymbol{b} \in \boldsymbol{S}$.

6) In addition to 5, if $\oplus$ does not commute (or if $\boldsymbol{a} \oplus \boldsymbol{b} \neq \boldsymbol{b} \oplus \boldsymbol{a}$) then given $(\boldsymbol{a} \oplus \boldsymbol{b})$, $(\boldsymbol{b} \oplus \boldsymbol{a})$ and $\boldsymbol{a}$, computation of $\boldsymbol{b}$ is still intractable for most $\boldsymbol{a}, \boldsymbol{b} \in \boldsymbol{S}$.

Clearly, condition 6 is more demanding than condition 5. Due to conditions 1- 4, we say that $\oplus$ is a 'weak' AOWF. A function satisfying 1-6 is a 'strong' AOWF. In [1], it is shown that weak AOWFs exist if P $\neq$ NP. Integer multiplication is a likely candidate for such an AOWF. Whether strong AOWFs exist or not (conditions 5, 6) is still an open question. In this paper, we do not attempt to go

further into this subject[1] (see also [8, 9]). We instead show how Rabi and Sherman's multiparty key agreement using a strong commutative AOWF [1] can be extended to a non-commutative one, provided that conditions 1-6 are satisfied. We describe two protocols that closely satisfy the requirements of a 'Quick Key' system discussed above.

5. QUICK GROUP KEY EXCHANGE

In this section we describe a way to achieve fully contributory group key agreement with only passive participation from the members. When a group is formed, each member obtains public keys of the other members in that group and generates the shared group key using a combination of his/her private key and those public keys. Also when there is a membership change, the new group key can be computed using only publicly available information *before* the membership change. We also assume the existence of a Public Key Infrastructure (PKI) for authentication purposes (i.e. to sign public keys and other plain-text information).

5.1 Group Key Agreement using a strong AOWF

The protocol called Quick Group Key Exchange (or QGKE) will work as follows. A notice board[2] is used for exchanging public keys. Assume that there are n users in a domain, which we denote by the set $\{U_1, U_2 \ldots U_n\}$. Assume also that we have a commutative AOWF, $\oplus$ over a set S that satisfies conditions 1-5 of section 4.2. QGKE will proceed as follows:

1) The group initiator generates a random element $a \in S$ and publishes it on the notice board.

2) Each user U_i wishing to join the group now generates a random element $x_i \in S$ and computes $y_i = (a \oplus x_i)$. Here x_i is the private key and y_i is the public key. All such users now publish their public keys along with their names on the notice board in the format: $\{y_i, U_i\}$. If all n users wish to join the group, there will be n sets of public keys on the notice board in the following form:

$\{y_1, U_1\}$
$\{y_2, U_2\}$
...
$\{y_n, U_n\}$

3) Now assume that only users U_p, U_q and U_r decide to be in the group. Assume further that U_p wants to send messages to the group. U_p will generate the group key $k_{\{pqr\}}$ using the following operation:
$k_{\{pqr\}} = (x_p \oplus y_q \oplus y_r) = (a^2 \oplus x_p \oplus x_q \oplus x_r)$

4) Users, U_p and U_q will obtain the same key $k_{\{pqr\}}$ by doing the following operations respectively:
$k_{\{pqr\}} = (x_q \oplus y_p \oplus y_r) = (a^2 \oplus x_p \oplus x_q \oplus x_r)$ for U_q
$k_{\{pqr\}} = (x_r \oplus y_p \oplus y_q) = (a^2 \oplus x_p \oplus x_q \oplus x_r)$ for U_r

Assuming that $\oplus$ is truly one way, given the value of any y_i $(=a \oplus x_i)$, it will be difficult to remove a from the product. Messages encrypted with the group key $k_{\{pqr\}}$ can only be recovered by three group members: U_p, U_q and U_r. Such a system can handle a dynamic number of users. Each time a user joins or leaves the group, the sender of a message updates his or her group key. All receivers can then decrypt the message only after updating their group keys.

The above protocol described an initial key generation operation. Further membership changes will require a new key to be computed as discussed in section 4.1. We note that the public keys in step 2 can be reused to generate new and secure group keys as users join or leave. Also each membership change requires only a short re-key message containing the identities of the newly added or removed members. As an example, suppose U_s joins the group. The four users can now independently compute the new key $k_{\{pqrs\}}$ in just one step as follows:

$$k_{\{pqrs\}} = (x_p \oplus y_q \oplus y_r \oplus y_s) = (y_p \oplus x_q \oplus y_r \oplus y_s)$$
$$= (y_p \oplus y_q \oplus x_r \oplus y_s) = (y_p \oplus y_q \oplus y_r \oplus x_s)$$
$$= (a^3 \oplus x_p \oplus x_q \oplus x_r \oplus x_s).$$

Note that knowledge of $k_{\{pqrs\}}$does not imply knowledge of $k_{\{pqr\}}$. We can see that QGKE appears to provide forward and backward secrecy. We further note that the public keys on the notice board can be long-term (i.e. as long as the lifetime of the group). To enable correct decryption of the message, we would require the identities of current group members to be sent with every message to enable the appropriate group key to be computed. Assuming that this is sent in a header, the size of the header increases as $O(n)$ where n is the size of the group. To counter this, we propose the use of group membership identifiers for a group where membership changes occur less frequently. To use external group signatures (discussed in section 5.2), however, we would still have to send the identities of all group members at the time of signature creation.

5.2 Group Signatures using QGKE.

As another application of QGKE, we propose 'group' signatures. Since every membership of the group is associated with a unique group key, we could use it to

[1] As an intuitive but useless example of an AOWF, assume a square matrix A such that the determinant, $|(A \bmod p)| = 0$. In other words, $(A \bmod p)$ is singular, in which case solving for $Y \equiv AX \pmod p$ given Y and A may be slightly more difficult than if $(A \bmod p)$ was non-singular. The symbol $(A \bmod p)$ denotes a matrix where each element of A is replaced by its residue modulo a prime p. Since matrix multiplication is associative, this could serve as an AOWF for '*kiddy*' crypto.

[2] A ***Notice Board*** is a virtual public directory using which users can exchange public keys and other plain-text messages. Users would initially require a certified public key to 'join' the notice board. After that, they could use their corresponding private key to sign further messages they wish to publish on the notice board. In short, the notice board is assumed to be tamper-proof. Note that a notice board is a hypothetical concept we use to encapsulate the multicast requirements of a group communication system. We see that a notice board requires the existence of a PKI and a robust multicast infrastructure. A notice board is in a way similar to a broadcast radio channel.

authenticate group messages. We discuss two types of signature schemes below. As before, we assume that $\oplus$ is a strong commutative AOWF over a large set S.

1. Internal group signatures

These are digital signature created by group members and verifiable only by group members. For simplicity, we assume that group messages are in plain text and anyone who is in possession of the group key, k can 'sign' those messages. The signature will be created as follows:

To sign a message $m \in S$, a group member computes the signature, $s = m \oplus k$. The tuple $\langle m, s \rangle$ then form a valid message-signature pair. We note that other group members in possession of k can verify such a signature by checking that $m \oplus k = s$.

2. External group signatures

These are digital signatures created by group members and verifiable by everyone who has access to the notice board.

Again, for simplicity, we assume messages are in plain text and any of the three members, U_p, U_q and U_r sharing group key $k_{\{pqr\}}$ would like to create a signature verifiable by everyone. Our signature protocol works as follows:

We know from above that $k_{\{pqr\}} = (a^2 \oplus x_p \oplus x_q \oplus x_r)$. To sign a message $m \in S$ we compute $s = k_{\{pqr\}} \oplus m$. We see that $\langle m, s \rangle$ now forms a valid message signature-pair. To verify the signature any user who has access to the notice board computes $y = y_p \oplus y_q \oplus y_r$ and checks that $m \oplus y = s \oplus a$.

Note that both the internal and external group signatures are identical since they were created using the same key however to verify an external signature, a non-member needs to know the identities of members in the group at the time of signature creation. Thus in addition to the message and the signature, the identities of all group members have to be attached to make a valid signature.

5.3 Generic signatures using a strong AOWF

We note that strong AOWFs can also be used to create generic digital signatures as shown below. Assume that Alice wants sign a message that Bob could later verify. The signature is created as follows:

1. Setup Phase

Alice and Bob first decide on a random $a \in S$. Then Alice generates a private key $x \in S$, computes her public key $y = x \oplus a$ and publishes it on the notice board.

2. Signature creation and verification

Alice can now sign a message $m \in S$ using her private key by computing $s = m \oplus x$. Alice sends $\langle m, s \rangle$ to Bob who verifies: $s \oplus a = m \oplus y$.

6. EXTENDED QUICK GROUP KEY EXCHANGE

The QGKE described above works only if $\oplus$ is both commutative and associative (and of course one-way). Assuming the existence of a function, $\oplus$ satisfying conditions 1-6 (section 4.2), here's how QGKE can be modified to work even if $\oplus$ is non-commutative: Step 1 will still be the same as above where the initiator posts his random element, a on the notice board. The protocol will be modified from step 2 onwards as follows:

1) Any user U_i wishing to join generates a secret random element $x_i \in G$ as the private key and computes $y_i = (a \oplus x_i)$ and $y_i' = (x_i \oplus a)$ as public keys. Here y_i will be called the *post-key* and y_i' the *pre-key*. All such users then publish the public keys along with their name on the notice board: $\{y_i, y_i', U_i\}$. If all n users wish to join the group, there will be n sets of public keys on the notice board in the following format.

   $\{y_1, y_1', U_1\}$
   $\{y_2, y_2', U_2\}$
   ...
   $\{y_n, y_n', U_n\}$

2) Now assume that only users U_p, U_q and U_r wish to be in the group. In this protocol the order is preserved so $\{U_p, U_q, U_r\}$ is different from $\{U_q, U_p, U_r\}$. Assume further that U_p wants to send messages to the group. U_p will generate the common group key $k_{\{pqr\}}$ using the following relation:
   $k_{\{pqr\}} = (x_p \oplus y_q \oplus y_r) = (x_p \oplus a \oplus x_q \oplus a \oplus x_r)$

3) Similarly, users U_q and U_r will obtain $k_{\{pqr\}}$ by doing the following operations respectively:
   $k_{\{pqr\}} = (y_p' \oplus x_q \oplus y_r) = (x_p \oplus a \oplus x_q \oplus a \oplus x_r)$ for U_q
   $k_{\{pqr\}} = (y_p' \oplus y_q' \oplus x_r) = (x_p \oplus a \oplus x_q \oplus a \oplus x_r)$ for U_r

In general if any users in the group want to generate a shared group key, they will first need to determine their respective positions in the group order. They will then combine in order, all the *pre-keys* of the members before them, their private key and the *post-keys* of the members after them. Since we have assumed that $\oplus$ is truly one-way, it will not be possible for any onlooker to invert $\oplus$ and recover the secret keys. Since this is essentially an extension of QGKE, we shall call it the Extended-QGKE (or simply EQGKE). As before, we can see that further membership changes will require short re-key messages containing the identities of newly added or removed members. A new key is quickly computed as long as all members have access to the notice board.

Like QGKE, we could also use EQGKE for creating digital group signatures discussed in section 5.2. We show how to create external group signatures. Internal signatures will be created in an identical way. As before, we assume messages are in plain text and the three members, U_p, U_q and U_r sharing group key $k_{\{pqr\}}$ would like to create signatures verifiable by everyone. We proceed as follows:

We know from above that $k_{\{pqr\}} = (x_p \oplus a \oplus x_q \oplus a \oplus x_r)$. To sign a message $m \in S$ we compute $s = k_{\{pqr\}} \oplus m$. The tuple $<m, s>$ forms a valid message signature-pair. To verify the signature any user who has access to the notice board computes $y = y_p \oplus y_q \oplus y_r$ and checks that $y \oplus m = a \oplus s$. The above procedure works because we have:

$y \oplus m = (a \oplus x_p \oplus a \oplus x_q \oplus a \oplus x_r) \oplus m$ and

$a \oplus s = a \oplus (x_p \oplus a \oplus x_q \oplus a \oplus x_r \oplus m)$.

We see that to enable verification of such a signature, the identity of the members holding the signing key $k_{\{pqr\}}$ also needs to be disclosed.

7. QUICK KEYS REVISITED

Assuming the existence of a strong AOWF, we see that two protocols discussed above satisfy the requirements of a 'Quick Key' system. As an intuitive but somewhat impractical example of QGKE (section 5), we describe a hypothetical system that can somehow use the colour of a paint mixture as the key. Assuming that mixing paint is both commutative and associative (i.e. the resulting colour of the mixture is independent of the order in which the constituent colours are mixed). We further assume that mixing paint is a 'one-way' operation. Without loss of generality we show how three users (Alice, Bob and Carol) can then use QGKE:

1) Alice, Bob and Carol agree on a common colour (say Yellow) and each chooses a secret colour (X, Y and Z respectively)
2) Alice makes two buckets of mixture X_1 containing 1 litre Yellow and 1 litre of X and sends one to Bob and one to Carol. Bob makes two buckets of mixture Y_1 containing 1 litre Yellow and 1 litre of Y and sends one to Alice and one to Carol. Similarly, Carol makes two buckets of mixture Z_1 containing 1 litre Yellow and 1 litre of Z and sends one to Alice and one to Bob.
3) Alice, Bob and Carol mix the two buckets they receive and add one litre of their secret colour to the mixture to get a common 'secret' colour, the mixture $K = X+Y_1+Z_1 = X_1+Y+Z_1 = X_1+Y_1+Z$ which no one else can obtain.

Note that the secret key was computed in step 3 independently by all three and the transactions in step 2 could be done in any order. Thus all three members contribute passively rather than actively to the secret key.

8. CONCLUSIONS AND OPEN PROBLEMS

In this paper, we discussed the concept of active and passive participation and showed that active participation increases communication overhead. We then described a novel way of using 'strong' Associative One-Way Functions (AOWFs) to do passive and fully contributory multiparty key agreement by presenting two protocols called QGKE and EQGKE. We showed that even non-commutative one-way functions could be used for multiparty key exchange as long as they are associative. We also discussed ways in which our key agreement protocols could also be used to create 'group' digital signatures.

To make our protocols work, we require a practical implementation of a strong AOWF. From the above discussions, we can informally list properties of the $\oplus$ operation in our AOWF (it should be associative and difficult to invert). By "difficult", it is meant that inverting $\oplus$ should be *provably* hard. We note that an existence proof of a strong AOWF is lacking and hope that our paper arouses further interest for research in this area. In this regard, we pose a few open questions:

1) What conditions are necessary for the existence of a deterministic polynomial time algorithm for inverting all associative one-to-one functions?
2) Assuming the existence of a strong AOWF (section 4.2), prove or disprove the security of EQGKE (section 6).
3) Is the existence of a strong AOWF necessary for creating 'Quick Keys' discussed in section 4.1?

REFERENCES

[1] Rabi Muhammad and Sherman. Alan "Associative one-way functions: a new paradigm for secret-key agreement and digital signatures," Computer Science Dept, University of Maryland Baltimore County, Nov 15 1993.

[2] Tsudik Gene and Waidner Michael, "Key agreement in dynamic peer groups", IEEE Transactions on Parallel and Distributed Systems, Aug 2000.

[3] Vesna Hassler, "Aspects of group communication security", Institut fur Angewandte Informationsverarbeitung und Kommunikationstechnologie, Graz University Of Technology, Austria., 1995.

[4] Yair Amir, Yongdae Kim, Cristina Nita-Rotaru, John Schultz, Jonathan Stanton and Gene Tsudik, "Exploring robustness in group key agreement", ICDCS 2001.

[5] Yongdae Kim, Adrian Perrig and Gene Tsudik, "Tree-based group key agreement", Dept. of Information and Computer Science, University of California at Irvine, CA, USA.

[6] Steiner Michael, Tsudik Gene and Waidner Michael, "Diffie-Hellman key distribution extended to group communication", ACM Conference on Computer and Communications Security, 1996

[7] Whitfield Diffie and Martin E. Hellman, "New directions in cryptography", IEEE Transactions On Information Theory, Vol IT-22, No. 6., November 1976.

[8] Lane A. Hemaspaandra and Jorg Rothe, "Creating strong total commutative one-way functions from any one-way function", Univ. Of Roch. Dept. of Computer Science, Technical Report 98-688, May 8, 1998.

[9] Lane A. Hemaspaandra and Kari Pasanen, "If P≠NP then some strongly noninvertible functions are invertible", Technical Report UR-CS-TR-2000-737, Aug 24 2000.