Broadcast Encryption

Krishnaram Kenthapadi

November 11, 2003

Introduction

Our system consists of a center and a set, U of n users. The center provides keys to the users when they join the system. Later on, the center wants to broadcast an encrypted message (such as the password to view "Matrix") which can be deciphered only by a *dynamically changing* privileged subset of users, T, i.e., the non-privileged users should not be able to learn the message. To achieve this, the members of T should be able to agree on a common secret key, based only on the keys present with each member and the broadcasts of the center.

One approach is for the center to give every user a key and broadcast an individually encrypted message $|T|$ times corresponding to each privileged user. Another approach is to provide every possible subset of users with a key (give each user the keys corresponding to the subsets it belongs to). The former requires a very long transmission whereas the latter requires each user to store exponentially many keys. Fiat and Naor [1] describe schemes that are efficient in both these measures and are also computationally efficient.

A broadcast scheme is *resilient* to a set of users S if for every subset T that is disjoint from S, no eavesdropper that has all the secrets of S, can obtain "knowledge" about the secret common to T. We could either consider the information-theoretic or the computational definition of security. The scheme is $k-resilient$ if it is resilient to any set $S \subset U$ of size k.

Basic Zero Message Scheme

In zero message schemes, the members of the privileged set, T can agree upon a common key *without* any broadcasts from the center, provided each member knows the set T itself. A simple $k-resilient$ scheme is as follows: For every set $B \subset U, 0 \leq |B| \leq k$, define a (fixed-length bit string) key K_B and give K_B to every user $x \in U \setminus B$. Thus every user x gets $O(n^k)$ keys corresponding to all subsets of size at most k, it is not a part of. The common key for a privileged set T is the XOR of all keys $K_B, B \subset U \setminus T$. Any colluding set S ($|S| \leq k$) will not have the key, K_S and hence cannot generate the common key for any disjoint set T.

A 1-Resilient Scheme using Pseudorandom Generators (PRG)

The above method provides a $1-resilient$ scheme in which each user stores n keys. We can reduce this to $\log n$ keys per user if the keys are psuedorandomly generated. Let $f : \{0,1\}^l \mapsto \{0,1\}^{2l}$ be a pseudorandom generator (known to everyone). Associate the users with the leaves of a balanced binary tree. The root is associated with a seed, $s \in \{0,1\}^l$. We associate the first l bits of $f(s)$

with left child of root and the last l bits with the right child and proceed recursively. Each leaf (user) x should get keys corresponding to all singleton sets except itself. We delete the nodes along the path from x to the root and provide the keys of the siblings of these nodes to x. Hence x can generate keys of all other leaf nodes, using the $\log n$ keys it has. Clearly two users can collude and obtain the keys of all leaf nodes - hence this is not $2-$ *resilient*.

A 1-Resilient Scheme using RSA Conjecture

The center picks a hard-to-factor composite $N = P.Q$ where P and Q are primes and chooses a secret element, $g \in \{1, 2, \ldots, N-1\}$ of high order. It also chooses mutually coprime numbers $p_i, 1 \leq i \leq n$ and gives user i the key, $g_i = g^{p_i} \bmod N$ (The (i, p_i) tuples are known to all users). The common key for a privileged set T is $g_T = g^{\prod_{i \in T} p_i} \bmod N$. Each user $i \in T$ computes g_T as $g_i^{\prod_{j \in T \setminus \{i\}} p_j} \bmod N$. If some user $j \notin T$ can compute g_T, then we can prove that it can also obtain the secret value, g. Thus, assuming the hardness of root extraction modulo composite (i.e., getting g given r and $g^r \bmod N$), we get a $1-$ *resilient* scheme with only *one* key per user.

	Basic scheme	Using PRG	Using RSA conjecture
Keys per user	n	$\log n$	1
Assumption	None	1-way functions exist (weaker)	RSA is 1-way function (stronger)
Security	Information theoretic	Computational	Neither

Comparison of $1-$ *resilient* **schemes**

k-Resilient Schemes

Any of these $1-$*resilient* schemes can be used as a building block to obtain $k-$*resilient* schemes. The center wants to send a secret key, M to a privileged set of users. The idea is to use a perfect family of hash functions and send a "share" of the secret, M, corresponding to each hash function. Each share is in turn broadcast with different encryptions. The privileged users can decrypt these messages as they are encrypted using the common keys from the independent $1-$*resilient* schemes. This is done in such a way that any colluding set of at most k users cannot obtain at least one of the shares (corresponding to the function which is perfect for this set) and no information about M is revealed if we miss even one of the shares.

These are not zero-message schemes. An open problem is to consider lower bounds:

- Can we obtain zero-message $k-$*resilience* using less than $O(n^k)$ keys per user? In particular, can we get zero-message $2-$ *resilience* using $o(n^2)$ keys per user? (As the keys need to be distributed initially in the database version of the problem (see below), lesser number of keys per user is desirable.)

Database Problems

Suppose we have a single attribute database, with an access control list (ACL) for each row. The database can be thought of as a set,S of n elements, which are encrypted and stored. There are

m users who want to access some elements of this set ($m << n$). For each element s_j, ACL_j is the set of users who should be able to access s_j. Thus we could use a zero-message $1 - resilient$ broadcast encryption scheme and encrypt s_j using the common key of ACL_j. As the common key of ACL_j is the XOR of the keys singleton sets, user revocation can be done in constant time: when some user i is removed from ACL_j, every other user in ACL_j can XOR the current common key and the key of the singleton set $\{i\}$ to agree on a new common key. There are two problems with this solution:

- This requires a table of very large size, $O(mn)$ to store the ACL's (as we have n rows, each with an encrypted item and an access control list). Can we trade-off user revocation time for space?
- The item s_j needs to be reencrypted every time ACL_j is modified. This is not desirable when the item itself is very large, say, a huge video file. On the other hand, if we encrypt s_j with a static key, k_i and encrypt and store k_i with the common key of ACL_j, user revocation cannot be done. Another approach is not to treat the encryption as a black box - as the new common key is obtained by modifying the old one, can we do efficient "incremental" encryption without sacrificing security?

Thus we want to apply broadcast encryption ideas to the above database problem, requiring only 1-resilience but with efficiency and the ability to modify user access lists online.

We can generalize to higher dimensions. Suppose the database has n_1 rows and n_2 columns. The users should be able to access only entries in some rectangles. We could achieve this by associating keys with every row and column and encrypting the $(i, j)^{th}$ entry by the key for row i followed by the key for column j. The user who can access a $l_1 * l_2$ rectangle is given $l_1 + l_2$ keys of the corresponding rows and columns. When the user has to lose access to say, a row i, the key for row i has to be changed and hence all entries in row i have to be reencrypted. Instead, we may associate keys with each entry in the table (resulting in $n_1 n_2$ keys), making the revocation faster.

References

[1] Amos Fiat and Moni Naor. Broadcast encryption. In Advances in Cryptology – CRYPTO'93, LNCS 773, pages 480–491. Springer-Verlag, 1994. `http://citeseer.nj.nec.com/fiat94broadcast.html`