ROUGH DRAFT
Privacy Preserving Key Reissuing Techniques for Hierarchical Multigroups using Robust Broadcast Encryption: Some Subtleties

Chris Laas

July 24, 2000

`$Id: key-reissue.tex,v 1.4 2000/07/25 23:01:49 golem Exp $`

1 Introduction

The "hierarchical multigroups" system proposed by Ohta, Okamoto, and Koyama in [?], based on [?] and [?] is nearly optimal as a basis for a privacy-preserving group-membership authorization system:

- Proofs of membership cannot be linked with individuals (fulfilling the basic privacy requirement).
- In addition, the group list of a particular individual is kept private.
- The master key of a user can be stored in a constant, small amount of memory.
- Membership verification can be performed in an off-line environment.

This scheme, however, is not stellar when group lists are dynamic and provers are usually offline: although adding members to a group is easy, revoking membership can only be accomplished by disabling the old group and creating a new group (minus the revoked members). When users must identify themselves uniquely in order to obtain new keys, frequent revocations pose a significant risk to the basic privacy requirement. In this paper, we attempt to solve this problem, without severely degrading the other desirable properties of the Ohta-Okamoto-Koyama system.

To this end, we first attempt to solve the problem using a modified form of Broadcast Encryption [**?**]; and, finding this arrangement to be flawed, we introduce a new primitive, Verifiable Broadcast Encryption, and use it to plug the leak.

2 The basic Ohta-Okamoto-Koyama system

XXX give overview of OOK here.

3 Broadcast Encryption and Key Predistribution Schemes

3.1 Overview

The concept of "broadcast encryption" was introduced by Fiat and Naor in [**?**]; generally speaking, its goal is to securely transmit a single message to an arbitrary subgroup of a larger "universe" while minimizing communications overhead. Obviously, this primitive has applications beyond encrypted cable TV, and indeed, several authors (XXX citations, Berkovits, Stinson, others?) have proposed "Key Predistribution Schemes" for key management based on the concept of broadcast encryption. However, most such schemes have communications or storage complexity at least linear in the number of members of the "privileged" group; for large groups in smart-card-based systems, this is hardly optimal. (Possibly mention the optimality results of Stinson and why they don't apply.)

In this section, we will consider an (apparantly) new Broadcast Encryption scheme; this scheme will be distinguished in that the communications complexity of a message broadcast will not be linear in the size of the "privileged" group, but rather $O(r \log r)$ in the size r of the "forbidden" group. (In addition, when the privileged group is small and the forbidden group large, the system will have communications complexity linear in the size of the privleged group; this added bonus will be a natural aspect of the scheme.) This will prove to be an efficient approach to the problem of revocation of group membership for a small number of revocations.

3.2 Definitions

Let $\mathcal{P}$ be a collection of "privileged sets" of users, and let $\mathcal{F}$ be a collection of "forbidden sets" of users. Then, following the convention of [**?**], we will call a scheme a $(\mathcal{P}, \mathcal{F})$-Broadcast Encryption Scheme (or $(\mathcal{P}, \mathcal{F})$-BES) if each user in a privileged set $P \in \mathcal{P}$ can decrypt a message encrypted by the center for P, and no forbidden subset $F \in \mathcal{F}$ disjoint from any privileged subset P can decrypt any message encrypted by the center for P.

Adapting the terminology of [?], our new BES would be a $(\leq n, \leq n)$-BES, in other words, one capable of maximum flexibility, in an overall group of size n. The storage requirement for each user will be $O(\log n)$ keys (although the RSA assumption will permit this to be reduced to $O(1)$) and the communications requirement for an individual broadcast will be $O(\min(k, r \log r))$, where k is the size of the privileged group and $r = n - k$ is the size of the forbidden group.

3.3 The basic scheme

Each user is given a unique identifier I such that $0 \leq I < n$. In the remainder of the system, I is treated as an (MSB first) bit string $I_1 I_2 \ldots I_\ell$ (where $\ell = \lceil \lg n \rceil$). The users are arranged in a balanced binary tree; each leaf represents a user, and each internal node (indexed by shorter bit strings) represents a group of users.

Each node, leaf or internal, has an associated key; hence, there are a total of roughly $2n$ keys for this system (there may be slightly more if n is not an exact power of two). These keys are generated and held by the central authority. Each user receives the keys corresponding to each ancestor; hence, the storage requirement for a user is $\ell + 1 = O(\log n)$.

To send a message m to a subset P of the users, the central authority must encrypt m with each key in a subset of the keys in the tree. This subset is determined by the following algorithm:

1. Begin by setting the set of priviledged nodes A to the set of leaves corresponding to users in P.

2. While any two nodes u, v in A share a common parent w, remove u and v from A and insert w into A.

At the end of this algorithm, A contains an optimal set of keys; and it is easily shown that $|A| = O(\min(k, r \log r))$ where $k = |P|$ and $r = n - k$. To complete the procedure, the central authority encrypts m in each of the keys in A[1], bundles the cyphertexts in a packet, and broadcasts this packet to all users, along with information about the membership of P.

[1] A symmetric cipher is assumed here, although modifications can be made to support public-key ciphers.

To decrypt the message, any member of P can determine which key he holds corresponding to a cyphertext in the broadcast packet, and can use it to decrypt that cyphertext to obtain m. Any individual not in P does not have any key which will decrypt any of the cyphertexts in the packet.

3.4 A memory efficiency improvement

In a device such as a smart card, even $O(\log n)$ keys may be too much to reasonably store in long-term memory. Luckily, we can use a trick due to Akl *et al* [**?**]: if each node in the tree is assigned a unique small odd prime[2], and if the central authority generates the node keys according to a specific procedure, each user need only store one "master key" from which all of his node keys can be generated. This reduces the memory requirements to a small constant amount of space; the cost is that the RSA assumption must be assumed to hold, whereas in the former scheme the RSA assumption was not necessary.

4 Application of this BES to a privacy protecting group membership revocation scheme

4.1 Description of the system

Let us assume that membership in a group is proven (in an external system) by demonstrating knowledge of a secret key corresponding to that group. Since every member of the group holds the same secret key, there is no way for a verifier to distinguish among group members; hence, the proving protocol is anonymous. However, this presents a difficulty if a verifier wishes to accept any member of a group except a few, perhaps because those few have had their membership in the group revoked (and the verifier has received a CRL or the like).

The only natural solution to this problem is to disable the original group's secret key, create a new group secret key, and distribute this new key to each still-valid member. However, this presents a difficulty, if it is assumed that the member only learns of this situation when he attempts to prove membership to the verifier: if he identifies himself in order to prove eligibility for the new key, he has lost his anonymity in this proving transaction. Broadcast encryption would seem to provide the perfect solution to

[2] In fact, the number of primes can be more than halved beyond that, since only leaves actually need unique small odd primes.

this problem: the central authority can assemble an encrypted "broadcast" transmission openable by exactly those members who have not been revoked. Whenever a user attempts to prove his membership in that group, the verifier forwards the user the entire broadcast packet (using only $O(\min(k, r \log r))$ bandwidth), and the user recovers the new group key, with which he proves his continued membership. Of course, if the user's membership has been revoked, he will not be able to obtain the new group key.

4.2 A few simple attacks on this system

This simple system needs a few obvious changes before it is even marginally private. First, the central authority must not be permitted arbitrarily to generate such key renewal packets, since placing different keys in different packets would allow it to distinguish users. One solution to this problem might be to require a signature by the party who initiated the revocation (the "revoker"), on the broadcast packet itself. Of course, this requires that one trust this party not to misbehave, but ultimately one must do this anyway, as he could always incrementally revoke memberships (possibly imaginary ones) in order to track members. This problem could also be attacked using quorums of trusted third parties, and by other cryptographic methods not relevant to the discussion at hand.

Second, the central authority and/or verifier might simply choose not to submit the new packet to some subset of users, which could then be identified by its continued use of the old key. This problem could be readily solved by having all provers begin by attempting to use the old key, and "pretending" not to know about the new key until explicitly told about it. In addition to solving this second problem, it would also help solve the first problem by aiding efforts to *detect* cheating on the authority's part. The cost of this solution, of course, is decreased efficiency.

Third, the authority might attempt to track or break provers by creating key sets A which are non-optimal or which are not generated by the given algorithm. This attack can easily be avoided: the authority must include a signed statement by the revoker regarding the identities of the revoked members along with the "broadcast" packet, and provers will perform the given algorithm themselves to verify that the authority's keyset is correct. If the first solution to the first problem above is adopted, as well, the revoker should perform the same check before signing the broadcast packet.

4.3 A more subtle attack on this system

The above problems all have more-or-less viable solutions which more-or-less leave the system as is. However, the authority has one more attack which is much harder to protect against.

For concreteness, let us consider a group with four members, Alice, Bob, Carol, and Dave. Alice has identfier 00, Bob is 01, Carol is 10, and Dave is 11; the induced tree structure and keys are as shown in the figure.

(XXX DRAW A FIGURE)

Now, say that Dave has been removed from the group; and say that the authority wishes to distinguish between verification attempts made by Carol and those made by Alice or Bob. To do this, the authority may construct the broadcast packet by properly encrypting the new key in k_{AB}, but substituting gibberish where it should have placed the encryption of the new key in k_C. Since the revoker does not know k_{AB} nor k_C, it cannot check the validity of the ciphertexts, and so must blindly sign what the authority presents.

Later, a verifier presents this broadcast packet to any prover who engages in the protocol. If the prover turns out to be Alice or Bob, then the protocol continues normally, since the AB ciphertext is intact, and neither Alice nor Bob can tell that the C ciphertext is corrupted. However, when Carol receives the packet, she is unable to retrieve anything intelligible from the C ciphertext; either she detects this, and must abort, or she attempts to continue, and fails to prove knowledge of the new key. In either case, Carol must abort, while Alice and Bob must accept, and the authority must be able to tell the difference.

This attack cannot be protected against easily without changes to the underlying protocol. The simplest option would be to provide (verifiably) the revoker with all the keys in the user tree before hand, so that he can check the ciphertexts' validity before signing; however, this suffers from several disadvantages. It gives the revoker access to anything encrypted with those keys, in addition to the new group keys; it places yet more trust in the revoker to protect privacy; and by giving the revoker all the tree keys, the revoker becomes more closely associated with the authority, who is considered the adversary in the privacy analysis. In a logistical sense, it may be unclear beforehand who the revoker will be, as well.

Another option would be to use a public-key cipher to encrypt the new group key. Then, after decrypting the AB ciphertext with sk_{AB}, Alice or Bob could verify that, indeed, the C ciphertext is the valid result of encrypting the new group key with pk_C. If the C ciphertext turned out to be bogus, Alice or Bob would simply abort, and the protocol view of the verifier (and the authority) would be indistinguishable from that of a run with Carol. However, this scheme suffers a major disadvantage in that all users must carry the public key associated with the secret key in every node in the user tree; hence, the storage requirement for the users balloons to linear in the number of group members, which is hardly satisfactory.

However, this second option points towards a valid solution: in short, Alice and Bob (and possibly the revoker) need some mechanism to verify that all ciphertexts in the broadcast packet are encipherments of the same value under different keys. To this end, we introduce a new primitive: Verifiable Broadcast Encryption.

5 Verifiable Broadcast Encryption

5.1 Overview

In the standard broadcast encryption framework, there exists a central authority and a set of users. In the setup of the system, the center generates keys for all the users and distributes them; and at some later point, the center wishes to send a single broadcast message which is encrypted such that exactly a certain dynamic "privileged" subset of the users will be able to decrypt the message.[3]

Verifiable Broadcast Encryption adds a single additional requirement: any member of the privileged set can not only determine the encrypted message m, but can also verify that every member of the privileged set will receive exactly the same message. If the authority attempts to cheat by sending a packet which will decrypt to diffent messages for different users, all members of the privileged set will detect this with high probability, and will abort.[4]

5.2 Definitions

(Sketch mode for now, although should probably fill in with full probabilistic arguments later.)

[3]Most papers on broadcast encryption also specify a resiliency k; however, in our revocation protocols, we will always wish the resiliency to be equal to the size of the forbidden set, and hence we omit it here.

[4]In the context of group membership revocation, it is also necessary that these aborts be indistinguishable (from the point of view of the verifier/authority) from an abort due to the prover having had his membership revoked.

Verifiable Broadcast Encryption Scheme (VBES): pair of protocols. Issuing protocol, for issuing keys to all users. Broadcast protocol, for broadcasting to a subset of users. List properties: what are "secret keys" of users? Is it meaningful to talk about "public keys" of users? What exactly does the verification add, in a statistical, measurable sense? How do we specify the security of various aspects of the system (computational, information-theoretic; also one-time, secure after polynomially many broadcasts in series, secure after polynomially many broadcasts in parallel).

5.3 Protocols

5.3.1 Overview

Unlike the generalized scheme described above, this VBES depends on the "hierarchical multigroups" data structure introduced by Akl ([**?**, **?**]). In essence, each broadcast performs a one-way transformation on this data structure; the transformed tree keys are then used as a sort of one-time pad, which retains computational (but not information-theoretic) security over multiply broadcasts due to the one-wayness of the transformation.

5.3.2 Setup and issuing protocol

(XXX This is confusing, and also needs illustrations.)

During the system setup, the central authority picks a large composite modulus n as the product of two large random safe primes.[5] As before, each user is given an ℓ-bit identifier, and a tree is constructed. Each identifier I is assigned a unique small odd prime p_I; then, each node $J \in \{0,1\}^{[0,\ell]}$ is assigned one of these primes by the following rules:

1. If $J = I$ for some identifier I, i.e. if $|J| = \ell$, i.e. if J is a leaf, then $p_J = p_I$.
2. Else, the identifier $I(J)$ is formed by padding J on the right by zeros until its length is equal to ℓ, and p_J is set to the value of $p_{I(J)}$ on that leaf.

[5] The safeness of the prime factors is necessary for the verifiability property, because it must be the case that multiple prime roots do not exist modulo n. Thus, there should be publicly available proofs that n is a product of two safe primes. (Certainly these protocols will need work, but they need not be terribly efficient, as they will be used only for monitoring purposes.)

In short, each node is assigned a small odd prime: the set of leaves have mutually unique primes, and the left ("zero") branch of any node has the same prime as that node.

Each node J is also assigned an exponent

$$e_J = \prod_{i=0}^{|J|} p_{J[0:i]}$$

(where $J[0:i]$ is taken to mean the node id formed by the first i bits of J).

The authority generates a random value $x \in_{\mathcal{R}} \mathbb{Z}_n^*$, and then generates the value

$$T = \prod_J p_J$$

The center then generates the value

$$y = x^T \pmod{n}$$

and publishes y as a "public key," along with n and the list of primes and associated nodes in the user tree.

Lastly, for each user, the authority generates the user's master key

$$z_I = x^{T/e_I} \pmod{n}$$

and sends z_I privately to the user identified by I. The user confirms the composition of e_I and also confirms that

$$y = z_I^{e_I} \pmod{n}$$

and accepts if both pass.

5.3.3 Broadcast protocol

Exactly as in the above protocol, the authority chooses the minimal set A of nodes for which exactly the privileged subset can generate keys. It also chooses a (possibly random) transformation parameter a: a must be odd and relatively prime to T, and must never be used again for another broadcast using the results of this system setup. However, within those constraints, a may be chosen small if desired, so that it may, for example, be chosen as the next prime number after the last p_J used and after all previously used values of a.

The message m is then encrypted with each key found in A: for each $J \in A$, the authority calculates

$$w_J = m + y^{a/e_J} (\text{mod } n)^6$$

The number a is packaged up along with all the ordered pairs (J, w_J) such that $J \in A$; this package becomes the broadcast packet which is sent to all verifiers to pass on to provers when the time comes.

When a user receives this broadcast packet, she first calculates the expected set A and compares it against the one used by the authority; if they do not match, or if she is not a member of that set, she aborts. If she a decendent of a member J of that set, she calculates the transformed secret key for J,

$$z_J = z_I^{ae_I/e_J} (\text{mod } n)$$

She then calculates $m = w_J - z_J$. Finally, she verifies that all other w_J in this packet encode the same m, by verifying the following relationship for every $J \in A$:

$$(w_J - m)^{e_J} = y^a (\text{mod } n)$$

She accepts if this holds true for all $J \in A$; if the test fails for any $J \in A$, she aborts (in a manner indistinguishable, to the verifier, from a failure to decrypt).

5.4 Security analysis

This is incomplete, and will be until in-depth proofs can be made. But first thoughts.

The issuing procedure is purely out of Akt, and is known to be secure under the RSA assumption. Here "secure" means "no user gets a root of y which he wasn't explicitly given by the authority."

The RSA assumption would seem to imply that no information about roots of y leak out no matter how many as you use, so long as the as are chosen relatively prime to all the previous as and to T. Look into this further, but a full proof shouldn't be too hard.

The w_Js leak the z_J values to the members of the privileged group, and as seen above, the z_Js don't seem to leak anything about the roots of y. And clearly, the non-privileged know even less than the privileged.

The secrecy of m seems to be protected by a one-time-pad, essentially. Basically, if one assumes the RSA assumption, then non-privileged users cannot determine any z_J, and hence, cannot determine m (since that would give them every z_J).

The validity check would seem to work only if y^a has only one e_Jth root; this should be OK so long as n is publicly known to be a product of two safe primes.

5.5 Trust analysis

As in all anonymous protocols, the users must trust the large body of other users also to act in an anonymous manner; following the protocol as described above should be sufficient.

It does not seem that the users must trust the central authority to maintain their privacy, and this is the primary goal of the modifications. The users must still, to some extent, trust the revoker not to abuse his power; finding ways to limit this power may prove to be an interesting study in itself.

If the security results hold, then the central authority and verifiers are free to be as skeptical of the users as they wish, since the combination of certification and revocation should ensure that the provers cannot ever falsely prove membership in a group.

5.6 Efficiency analysis

The efficiency of the verifiable system is not much worse than that of the unverifiable version: storage costs are still constant (although multiplied by a factor of at least two), except for a potential very-small-coefficient logarithmic term due to storage of small prime numbers and/or identifiers. Communication complexity remains at the same level, increased only by exactly one small prime a. The authority's computational burden is not significantly affected, possibly being multiplied by a constant factor. The prover's computational burden is significantly increased, however: while it was constant in the previous scheme, it would be roughly proportional to the communication complexity in the verifiable system. This is, of course, unavoidable, since the point of verification is to verify every ciphertext in the broadcast packet. Since all decryption and verification routines in this system primarily consist of a small number of exponentiations by small exponents, it is hoped that these verifications can be performed efficiently. If not, it may well be possible to perform them in a batching or probabilistic manner.

6 Application of VBES to the group membership scheme outlined above

The Verifiable Broadcast Encryption Scheme described above can be used as a drop-in replacement for the BES originally used; the only difference is the addition of the verification step, which prevents the subtle attack described in section 4.3.

(Insert any nuggets of wisdom here.)

7 Any other sections (Conclusions, Results, Acknowledgments)

(Any other sections.)

(There will be a bibliography here.)