

# Collusion Resistant Broadcast Encryption With Short Ciphertexts and Private Keys

Dan Boneh\*  
dabo@cs.stanford.edu

Craig Gentry  
cgentry@docomolabs-usa.com

Brent Waters  
bwaters@cs.stanford.edu

## Abstract

We describe two new public key broadcast encryption systems for stateless receivers. Both systems are fully secure against any number of colluders. In our first construction both ciphertexts and private keys are of constant size (only two group elements), for any subset of receivers. The public key size in this system is linear in the total number of receivers. Our second system is a generalization of the first that provides a tradeoff between ciphertext size and public key size. For example, we achieve a collusion resistant broadcast system for  $n$  users where both ciphertexts and public keys are of size  $O(\sqrt{n})$  for any subset of receivers. We discuss several applications of these systems.

## 1 Introduction

In a broadcast encryption scheme [FN93] a broadcaster encrypts a message for some subset  $S$  of users who are listening on a broadcast channel. Any user in  $S$  can use his private key to decrypt the broadcast. However, even if all users outside of  $S$  collude they can obtain no information about the contents of the broadcast. Such systems are said to be collusion resistant. The broadcaster can encrypt to any subset  $S$  of his choice. We use  $n$  to denote the total number of users.

Broadcast encryption has several applications including access control in encrypted file systems, satellite TV subscription services, and DVD content protection. As we will see in Section 4 we distinguish between two types of applications:

- Applications where we broadcast to large sets, namely sets of size  $n - r$  for  $r \ll n$ . The best systems [NNL01, HS02, GST04] achieve a broadcast message containing  $O(r)$  ciphertexts where each user's private key is of size  $O(\log n)$ .
- Applications where we broadcast to small sets, namely sets of size  $t$  for  $t \ll n$ . Until now, the best known solution was trivial, namely encrypt the broadcast message under each recipient's key. This broadcast message contains  $t$  ciphertexts and each user's private key is of size  $O(1)$ .

In this paper we construct fully collusion secure broadcast encryption systems with short ciphertexts and private keys for *arbitrary* receiver sets. Our constructions use groups with an efficiently computable bilinear map. Our first construction provides a system in which both the broadcast message and user private keys are of constant size (a precise statement is given in the next section).

---

\*Supported by NSF.

No matter what the receiver set is, our broadcast ciphertext contains only two group elements. Each user's private key is just a single group element. Thus, for example, when broadcasting to small sets our system is far better than the trivial solution discussed above. However, we point out that the public key size in this system is linear in the number of recipients. This is not a large problem in applications such as encrypted file systems where the receivers have access to a large shared storage medium in which the public key can be stored. For other applications, such as content protection, we need to minimize both public key and ciphertext size.

Our second system is a generalization of the first that enables us to tradeoff public key size for ciphertext size. One interesting parametrization of our scheme gives a system where both the public key and the ciphertext are of size  $O(\sqrt{n})$ . This means that we can attach the public key to the encrypted broadcast and still achieve ciphertext size of  $O(\sqrt{n})$ . Consequently, we obtain a fully collusion secure broadcast encryption scheme with  $O(\sqrt{n})$  ciphertext size (for any subset of users) where the users have a constant size private key.

Section 2 defines our security model and the complexity assumption we use. In Section 3 we describe our systems and prove their security. Finally, in Section 4 we discuss in detail several applications for these systems.

## 1.1 Related Work

Fiat and Naor [FN93] were the first to formally explore broadcast encryption. They presented a solution which was secure against a collusion of  $t$  users and has ciphertext size of  $O(t \log^2 t \log n)$ .

Naor et al. [NNL01] presented a fully collusion secure broadcast encryption system that is efficient for broadcasting to all, but a small set of revoked users. Their scheme is useful for content protection where broadcasts will be sent to all, but a small set of receivers whose keys have been compromised. Their scheme can be used to encrypt to  $n-r$  users with a header size of  $O(r)$  elements and private keys of size  $O(\log^2 n)$ . Further improvements [HS02, GST04] reduce the private key size to  $O(\log n)$ . Dodis and Fazio [DF02] extend the NNL (subtree difference) method into a public key broadcast system for a small size public key.

Other broadcast encryption methods for large sets include Naor and Pinkas [NP00] and Dodis and Fazio [DF03]. For some fixed  $t$  these systems can revoke any  $r < t$  users where ciphertexts are always of size  $O(t)$  and private keys are constant size. By running  $\log n$  of these systems in parallel, where the revocation bound of the  $i$ 'th system is  $t_i = 2^i$ , one obtains a broadcast encryption system with the same parameters as [GST04]. Private key size is  $O(\log n)$  and, when revoking  $r$  users, ciphertext size is proportional to  $2^{\lceil \log_2 r \rceil} = O(r)$ . This simple extension to the Naor and Pinkas system gives a broadcast system with similar parameters as the latest NNL derivative.

Wallner et al. [WHA97] and Wong [WGL98] independently discovered the logical-tree-hierarchy scheme (LKH) for group multicast. Using these methods receivers must maintain state and remain connected to receive key-updates. The parameters of the original schemes are improved in further work [CGI<sup>+</sup>99, CMN99, SM03].

The security of our broadcast encryption relies on computational assumptions. Several other works [Sti97, ST98, SW98, GSY99, GSW00] explore broadcast encryption and tracing from an information theoretic perspective.

Boneh and Silverberg [BS03] show that  $n$ -linear maps give the ultimate fully collusion secure scheme with constant public key, private key, and ciphertext size. However, there are currently no known implementations of cryptographically useful  $n$ -linear maps for  $n > 2$ . Our results show that

we can come fairly close using bilinear maps alone.

## 2 Preliminaries

We begin by formally defining what is a secure public-key broadcast encryption system. For simplicity we define broadcast encryption as a key encapsulation mechanism. We then state the complexity assumption needed for our proof of security.

### 2.1 Broadcast Encryption Systems

A broadcast encryption system is made up of three randomized algorithms:

**Setup**( $n$ ) Takes as input the number of receivers  $n$ . It outputs  $n$  private keys  $d_1, \dots, d_n$  and a public key  $PK$ .

**Encrypt**( $S, PK$ ) Takes as input a subset  $S \subseteq \{1, \dots, n\}$ , and a public key  $PK$ . It outputs a pair  $(\text{Hdr}, K)$  where  $\text{Hdr}$  is called the header and  $K \in \mathcal{K}$  is a message encryption key. We will often refer to  $\text{Hdr}$  as the broadcast ciphertext.

Let  $M$  be a message to be broadcast to the set  $S$  and let  $C_M$  be the encryption of  $M$  under the symmetric key  $K$ . The broadcast to users in  $S$  consists of  $(S, \text{Hdr}, C_M)$ . The pair  $(S, \text{Hdr})$  is often called the full header and  $C_M$  is often called the broadcast body.

**Decrypt**( $S, i, d_i, \text{Hdr}, PK$ ) Takes as input a subset  $S \subseteq \{1, \dots, n\}$ , a user id  $i \in \{1, \dots, n\}$  and the private key  $d_i$  for user  $i$ , a header  $\text{Hdr}$ , and the public key  $PK$ . If  $i \in S$ , then the algorithm outputs the message encryption key  $K \in \mathcal{K}$ . The key  $K$  can then be used to decrypt the broadcast body  $C_M$  and obtain the message body  $M$ .

As usual, we require that the system be correct, namely that for all  $S \subseteq \{1, \dots, n\}$  and all  $i \in S$ , if  $(PK, (d_1, \dots, d_n)) \stackrel{R}{\leftarrow} \text{Setup}(n)$  and  $(\text{Hdr}, K) \stackrel{R}{\leftarrow} \text{Encrypt}(S, PK)$  then  $\text{Decrypt}(S, i, d_i, \text{Hdr}, PK) = K$ .

We define chosen ciphertext security of a broadcast encryption system against a static adversary. Security is defined using the following game between an attack algorithm  $\mathcal{A}$  and a challenger. Both the challenger and  $\mathcal{A}$  are given  $n$ , the total number of users, as input.

**Init.** Algorithm  $\mathcal{A}$  begins by outputting a set  $S^* \subseteq \{1, \dots, n\}$  of receivers that it wants to attack.

**Setup.** The challenger runs  $\text{Setup}(n)$  to obtain a public key  $PK$  and private keys  $d_1, \dots, d_n$ . It gives  $\mathcal{A}$  the public key  $PK$  and all private keys  $d_j$  for which  $j \notin S^*$ .

**Query phase 1.** Algorithm  $\mathcal{A}$  adaptively issues decryption queries  $q_1, \dots, q_m$  where a decryption query consists of the triple  $(u, S, \text{Hdr})$  where  $S \subseteq S^*$  and  $u \in S$ . The challenger responds with  $\text{Decrypt}(S, u, d_u, \text{Hdr}, PK)$ .

**Challenge.** The challenger runs algorithm  $\text{Encrypt}$  to obtain  $(\text{Hdr}^*, K) \stackrel{R}{\leftarrow} \text{Encrypt}(S, PK)$  where  $K \in \mathcal{K}$ . Next, the challenger picks a random  $b \in \{0, 1\}$ . It sets  $K_b = K$  and picks a random  $K_{1-b}$  in  $\mathcal{K}$ . It then gives  $(\text{Hdr}^*, K_0, K_1)$  to algorithm  $\mathcal{A}$ .

**Query phase 2.** Algorithm  $\mathcal{A}$  continues to adaptively issue decryption queries  $q_{m+1}, \dots, q_{q_D}$  where  $q_i = (u, S, \text{Hdr})$  with  $S \subseteq S^*$  and  $u \in S$ . The only constraint is that  $\text{Hdr} \neq \text{Hdr}^*$ . The challenger responds as in phase 1.

**Guess.** Algorithm  $\mathcal{A}$  outputs its guess  $b' \in \{0, 1\}$  for  $b$  and wins the game if  $b = b'$ .

Let  $\text{AdvBr}_{\mathcal{A},n}$  denote the probability that  $\mathcal{A}$  wins the game when the challenger is given  $n$  as input.

**Definition 2.1.** We say that a broadcast encryption system is  $(t, \epsilon, n, q_D)$  CCA secure if for all  $t$ -time algorithms  $\mathcal{A}$  who make a total of  $q_D$  decryption queries, we have that  $|\text{AdvBr}_{\mathcal{A},n} - \frac{1}{2}| < \epsilon$ .

The game above models an attack where all users not in the set  $S$  collude to try and expose a broadcast intended for users in  $S$  only. The set  $S$  is chosen by the adversary. Note that the adversary is non-adaptive; it chooses  $S$ , and obtains the keys for users outside of  $S$ , before it even sees the public key  $PK$ . An adaptive adversary could request user keys adaptively. We only prove security of our system in the non-adaptive settings described above. It is an open problem to build a broadcast encryption system with the performance of our system which is secure against adaptive adversaries. We note that similar formal definitions for broadcast encryption security were given in [BS03, DF03].

**Semantic Security.** As usual we define semantic security for a broadcast encryption scheme by preventing the attacker from issuing decryption queries.

**Definition 2.2.** We say that a broadcast encryption system is  $(t, \epsilon, n)$  semantically secure if it is  $(t, \epsilon, n, 0)$  CCA secure.

In Section 3 we first construct *semantically secure* systems with constant ciphertext and private key size. We come back to chosen ciphertext security in Section 5.

## 2.2 Bilinear Maps

We briefly review the necessary facts about bilinear maps and bilinear map groups. We use the following standard notation [Jou00, JN01, BF01]:

1.  $\mathbb{G}$  and  $\mathbb{G}_1$  are two (multiplicative) cyclic groups of prime order  $p$ ;
2.  $g$  is a generator of  $\mathbb{G}$ .
3.  $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_1$  is a bilinear map.

Let  $\mathbb{G}$  and  $\mathbb{G}_1$  be two groups as above. A bilinear map is a map  $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_1$  with the following properties:

1. Bilinear: for all  $u, v \in \mathbb{G}$  and  $a, b \in \mathbb{Z}$ , we have  $e(u^a, v^b) = e(u, v)^{ab}$ .
2. Non-degenerate:  $e(g, g) \neq 1$ .

We say that  $\mathbb{G}$  is a bilinear group if the group action in  $\mathbb{G}$  can be computed efficiently and there exists a group  $\mathbb{G}_1$  and an efficiently computable bilinear map  $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_1$  as above. Note that  $e(\cdot, \cdot)$  is symmetric since  $e(g^a, g^b) = e(g, g)^{ab} = e(g^b, g^a)$ .

### 2.3 Complexity Assumptions

Security of our system is based on a complexity assumption called the bilinear Diffie-Hellman Exponent assumption (BDHE). This assumption was recently used to construct a hierarchical identity based encryption system with constant size ciphertext [BBG05].

Let  $\mathbb{G}$  be a bilinear group of prime order  $p$ . The  $\ell$ -BDHE problem in  $\mathbb{G}$  is stated as follows: given a vector of  $2\ell + 1$  elements

$$\left(h, g, g^\alpha, g^{(\alpha^2)}, \dots, g^{(\alpha^\ell)}, g^{(\alpha^{\ell+2})}, \dots, g^{(\alpha^{2\ell})}\right) \in \mathbb{G}^{2\ell+1}$$

as input, output  $e(g, h)^{(\alpha^{\ell+1})} \in \mathbb{G}_1$ . Note that the input vector is missing the term  $g^{(\alpha^{\ell+1})}$  so that the bilinear map seems to be of little help in computing the required  $e(g, h)^{(\alpha^{\ell+1})}$ .

As shorthand, once  $g$  and  $\alpha$  are specified, we use  $g_i$  to denote  $g_i = g^{(\alpha^i)} \in \mathbb{G}$ . An algorithm  $\mathcal{A}$  has advantage  $\epsilon$  in solving  $\ell$ -BDHE in  $\mathbb{G}$  if

$$\Pr[\mathcal{A}(h, g, g_1, \dots, g_\ell, g_{\ell+2}, \dots, g_{2\ell}) = e(g_{\ell+1}, h)] \geq \epsilon$$

where the probability is over the random choice of generators  $g, h$  in  $\mathbb{G}$ , the random choice of  $\alpha$  in  $\mathbb{Z}_p$ , and the random bits used by  $\mathcal{A}$ .

The decisional version of the  $\ell$ -BDHE problem in  $\mathbb{G}$  is defined analogously. Let  $\vec{y}_{g,\alpha,\ell} = (g_1, \dots, g_\ell, g_{\ell+2}, \dots, g_{2\ell})$ . An algorithm  $\mathcal{B}$  that outputs  $b \in \{0, 1\}$  has advantage  $\epsilon$  in solving decision  $\ell$ -BDHE in  $\mathbb{G}$  if

$$\left| \Pr[\mathcal{B}(g, h, \vec{y}_{g,\alpha,\ell}, e(g_{\ell+1}, h)) = 0] - \Pr[\mathcal{B}(g, h, \vec{y}_{g,\alpha,\ell}, T) = 0] \right| \geq \epsilon$$

where the probability is over the random choice of generators  $g, h$  in  $\mathbb{G}$ , the random choice of  $\alpha$  in  $\mathbb{Z}_p$ , the random choice of  $T \in \mathbb{G}_1$ , and the random bits consumed by  $\mathcal{B}$ . We refer to the distribution on the left as  $\mathcal{P}_{BDHE}$  and the distribution on the right as  $\mathcal{R}_{BDHE}$ .

**Definition 2.3.** We say that the (decision)  $(t, \epsilon, \ell)$ -BDHE assumption holds in  $\mathbb{G}$  if no  $t$ -time algorithm has advantage at least  $\epsilon$  in solving the (decision)  $\ell$ -BDHE problem in  $\mathbb{G}$ .

Occasionally we drop the  $t$  and  $\epsilon$  and refer to the (decision)  $\ell$ -BDHE in  $\mathbb{G}$ . We note that the  $\ell$ -BDHE assumption is a natural extension of the bilinear-DHI assumption previously used in [BB04, DY05]. Furthermore, Boneh et al. [BBG05] show that the  $\ell$ -BDHE assumption holds in generic bilinear groups [Sho97].

## 3 Construction

We are now ready to present our broadcast encryption system. We first present a special case system where ciphertexts and private keys are always constant size, but the public key grows linearly with the number of users. We then present a generalization that allows us to balance the public key size and the ciphertext size. Private keys are still constant size. We prove security of this general system.

### 3.1 A Special Case

We begin by describing a broadcast encryption system for  $n$  users where the ciphertexts and private keys are constant size, but the public key grows linearly in the number of users.

**Setup( $n$ ):** Let  $\mathbb{G}$  be a bilinear group of prime order  $p$ . The algorithm first picks a random generator  $g \in \mathbb{G}$  and a random  $\alpha \in \mathbb{Z}_p$ . It computes  $g_i = g^{(\alpha^i)} \in \mathbb{G}$  for  $i = 1, 2, \dots, n, n+2, \dots, 2n$ . Next, it picks a random  $\gamma \in \mathbb{Z}_p$  and sets  $v = g^\gamma \in \mathbb{G}$ . The public key is:

$$PK = (g, g_1, \dots, g_n, g_{n+2}, \dots, g_{2n}, v) \in \mathbb{G}^{2n+1}$$

The private key for user  $i \in \{1, \dots, n\}$  is set as:  $d_i = g_i^\gamma \in \mathbb{G}$ . Note that  $d_i = v^{(\alpha^i)}$ . The algorithm outputs the public key  $PK$  and the  $n$  private keys  $d_1, \dots, d_n$ .

**Encrypt( $S, PK$ ):** Pick a random  $t$  in  $\mathbb{Z}_p$  and set  $K = e(g_{n+1}, g)^t \in \mathbb{G}$ . The value  $e(g_{n+1}, g)$  can be computed as  $e(g_n, g_1)$ . Next, set

$$\text{Hdr} = \left( g^t, \left( v \cdot \prod_{j \in S} g_{n+1-j} \right)^t \right) \in \mathbb{G}^2$$

and output the pair  $(\text{Hdr}, K)$ .

**Decrypt( $S, i, d_i, \text{Hdr}, PK$ ):** Let  $\text{Hdr} = (C_0, C_1)$  and recall that  $d_i \in \mathbb{G}$ . Then, output

$$K = e(g_i, C_1) / e(d_i \cdot \prod_{\substack{j \in S \\ j \neq i}} g_{n+1-j+i}, C_0)$$

Note that a private key is only one group element in  $\mathbb{G}$  and the ciphertext,  $\text{Hdr}$ , is only two group elements. Furthermore, since  $e(g_{n+1}, g)$  can be precomputed, encryption requires no pairings. Nevertheless, the system is able to broadcast to any subset of users and is fully collusion resistant. We prove security in Section 3.3 where we discuss a more general system.

We first verify that the system is correct, namely that the decryption algorithm works correctly. We use the fact that  $g_i^{(\alpha^j)} = g_{i+j}$  for any  $i, j$ . Suppose  $i \in S$ , then user  $i$  decrypts as:

$$\begin{aligned} K &= e(g_i, C_1) / e(d_i \cdot \prod_{\substack{j \in S \\ j \neq i}} g_{n+1-j+i}, C_0) \\ &= e(g^{(\alpha^i)}, (v \cdot \prod_{j \in S} g_{n+1-j})^t) / e(v^{(\alpha^i)} \cdot \prod_{\substack{j \in S \\ j \neq i}} g_{n+1-j+i}, g^t) \\ &= e(g^{(\alpha^i)}, (g_{n+1-i})^t) \cdot e(g^{(\alpha^i)}, (v \cdot \prod_{\substack{j \in S \\ j \neq i}} g_{n+1-j})^t) / e(v^{(\alpha^i)} \cdot \prod_{\substack{j \in S \\ j \neq i}} g_{n+1-j+i}, g^t) \\ &= e(g_{n+1}, g)^t \cdot e(g^{(\alpha^i)}, v \cdot \prod_{\substack{j \in S \\ j \neq i}} g_{n+1-j})^t / e(v^{(\alpha^i)} \cdot \prod_{\substack{j \in S \\ j \neq i}} g_{n+1-j+i}, g)^t \\ &= e(g_{n+1}, g)^t \cdot e(g, v^{(\alpha^i)} \cdot \prod_{\substack{j \in S \\ j \neq i}} g_{n+1-j+i})^t / e(v^{(\alpha^i)} \cdot \prod_{\substack{j \in S \\ j \neq i}} g_{n+1-j+i}, g)^t \\ &= e(g_{n+1}, g)^t \end{aligned}$$

as required.

**Efficient implementation.** For any large number of receivers, decryption time will be dominated by the  $|S| - 2$  group operations needed to compute  $\prod_{\substack{j \in S \\ j \neq i}} g_{n+1-j+i}$ . However, we observe that if the receiver had previously computed the value  $w = \prod_{\substack{j \in S' \\ j \neq i}} g_{n+1-j+i}$  for some receiver set  $S'$  that is similar to  $S$  then, the receiver can compute  $\prod_{\substack{j \in S \\ j \neq i}} g_{n+1-j+i}$  with just  $\delta$  group operations using the cached value  $w$ , where  $\delta$  is the size of the set difference between  $S$  and  $S'$ .

This observation is especially useful when the broadcast system is intended to broadcast to large sets, i.e. sets of size  $n - r$  for  $r \ll n$ . The private key  $d_i$  could include the value  $\prod_{\substack{j=1 \\ j \neq i}}^n g_{n+1-j+i} \in \mathbb{G}$  which would enable the receiver to decrypt using only  $r$  group operations. Furthermore, user  $i$  would only need  $r$  elements from the public key  $PK$ .

We note that computation time for encryption will similarly be dominated by the  $|S| - 1$  group operations to compute  $\prod_{j \in S} g_{n+1-j}^t$  and similar performance optimizations (e.g. precomputing  $\prod_{j=1}^n g_{n+1-j}$ ) apply. We also note that in the secret key settings (where the encryptor is allowed to keep secret information) the encryptor need only store  $(g, v, \alpha)$  as opposed to storing the entire public key vector.

### 3.2 General Construction

Next, we present our general broadcast encryption system. The idea is to run  $A$  parallel instances of the system in the previous section where each instance can broadcast to at most  $B \ll n$  users. As a result we can handle as many as  $n = AB$  users. However, we substantially improve performance by sharing information between the  $A$  instances. In particular, all instances will share the same public key values  $g, g_1, \dots, g_B, g_{B+2}, \dots, g_{2B}$ .

This generalized system enables us to tradeoff the public key size for ciphertext size. Setting  $B = n$  gives the system of the previous section. However, setting  $B = \lfloor \sqrt{n} \rfloor$  gives a system where both public key and ciphertext size are about  $\sqrt{n}$  elements. Note that either way, the private key is always just one group element.

Let  $B$  be a fixed positive integer. The  $B$ -broadcast encryption system works as follows:

**Setup $_B(n)$ :** The algorithm will set up  $A = \lceil \frac{n}{B} \rceil$  instances of the scheme. Let  $\mathbb{G}$  be a bilinear group of prime order  $p$ . The algorithm first picks a random generator  $g \in \mathbb{G}$  and a random  $\alpha \in \mathbb{Z}_p$ . It computes  $g_i = g^{(\alpha^i)} \in \mathbb{G}$  for  $i = 1, 2, \dots, B, B + 2, \dots, 2B$ . Next, it picks random  $\gamma_1, \dots, \gamma_A \in \mathbb{Z}_p$  and sets  $v_1 = g^{\gamma_1}, \dots, v_A = g^{\gamma_A} \in \mathbb{G}$ . The public key is:

$$PK = (g, g_1, \dots, g_B, g_{B+2}, \dots, g_{2B}, v_1, \dots, v_A) \in \mathbb{G}^{2B+A}$$

The private key for user  $i \in \{1, \dots, n\}$  is defined as follows: write  $i$  as  $i = (a - 1)B + b$  for some  $1 \leq a \leq A$  and  $1 \leq b \leq B$  (i.e.  $a = \lceil i/B \rceil$  and  $b = i \bmod B$ ). Set the private key for user  $i$  as:

$$d_i = g_b^{\gamma_a} \in \mathbb{G} \quad (\text{note that } d_i = v_a^{(\alpha^b)})$$

The algorithm outputs the public key  $PK$  and the  $n$  private keys  $d_1, \dots, d_n$ .

**Encrypt( $S, PK$ ):** For each  $\ell = 1, \dots, A$  define the subsets  $\hat{S}_\ell$  and  $S_\ell$  as

$$\hat{S}_\ell = S \cap \{(\ell - 1)B + 1, \dots, \ell B\} \quad \text{and} \quad S_\ell = \{x - \ell B + B \mid x \in \hat{S}_\ell\} \subseteq \{1, \dots, B\}$$

In other words,  $\hat{S}_\ell$  contains all users in  $S$  that fall in the  $\ell$ 'th interval of length  $B$  and  $S_\ell$  contains the indices of those users relative to the beginning of the interval. Pick a random  $t$  in  $\mathbb{Z}_p$  and set  $K = e(g_{B+1}, g)^t \in \mathbb{G}$ . As before, the value  $e(g_{B+1}, g)$  can be computed as  $e(g_B, g_1)$ . Set

$$\text{Hdr} = \left( g^t, (v_1 \cdot \prod_{j \in S_1} g_{B+1-j})^t, \dots, (v_A \cdot \prod_{j \in S_A} g_{B+1-j})^t \right) \in \mathbb{G}^{A+1}$$

Output the pair  $(\text{Hdr}, K)$ . Note that  $\text{Hdr}$  contains  $A + 1$  elements.

**Decrypt** $(S, i, d_i, \text{Hdr}, PK)$ : Let  $\text{Hdr} = (C_0, C_1, \dots, C_A)$  and recall that  $d_i \in \mathbb{G}$ . Write  $i$  as  $i = (a - 1)B + b$  for some  $1 \leq a \leq A$  and  $1 \leq b \leq B$ . Then

$$K = e(g_b, C_a) / e(d_i \cdot \prod_{\substack{j \in S_a \\ j \neq b}} g_{B+1-j+b}, C_0)$$

Verifying that the decryption algorithm works correctly is analogous to the calculation in the previous section. We note that when  $B = n$  then  $A = 1$  and we obtain the system of the previous section.

### 3.2.1 Efficiency

A user's private key size again will only consist of one group element. The ciphertext consists of  $A + 1$  group elements and the public key is  $2B + A$  elements. Our choice of  $B$  depends on the application. As we will see, in some cases we want  $B = n$  to obtain the smallest possible ciphertext. In other cases we want  $B = \sqrt{n}$  to minimize the concatenation of the ciphertext and public key.

The decryption time for user  $i = (a - 1)B + b$  will be dominated by  $|S_a| - 2 < B$  group operations. Similar caching techniques to those described in the end of Section 3.1 can be used to improve performance.

## 3.3 Security

We now prove the semantic security of the general system of Section 3.2.

**Theorem 3.1.** *Let  $\mathbb{G}$  be a bilinear group of prime order  $p$ . For any positive integers  $B, n$  ( $n > B$ ) our  $B$ -broadcast encryption system is  $(t, \epsilon, n)$  semantically secure assuming the decision  $(t, \epsilon, B)$ -BDHE assumption holds in  $\mathbb{G}$ .*

*Proof.* Suppose there exists a  $t$ -time adversary,  $\mathcal{A}$ , such that  $\text{AdvBr}_{\mathcal{A}, n} > \epsilon$  for a system parameterized with a given  $B$ . We build an algorithm,  $\mathcal{B}$ , that has advantage  $\epsilon$  in solving the  $B$ -BDHE problem in  $\mathbb{G}$ . Algorithm  $\mathcal{B}$  takes as input a random  $B$ -BDHE challenge  $(g, h, \vec{y}_{g, \alpha, B}, Z)$ , where  $\vec{y}_{g, \alpha, B} = (g_1, \dots, g_B, g_{B+2}, \dots, g_{2B})$  and  $Z$  is either  $e(g_{B+1}, h)$  or a random element of  $\mathbb{G}_1$  (recall that  $g_i = g^{(\alpha^i)}$ ). Algorithm,  $\mathcal{B}$ , proceeds as follows.

**Init** Algorithm  $\mathcal{B}$  runs  $\mathcal{A}$  and receives the set  $S$  of users that  $\mathcal{A}$  wishes to be challenged on.



**Setup**  $\mathcal{B}$  needs to generate a public key  $PK$  and private keys  $d_i$  for  $i \notin S$ . The crux of the proof is in the choice of  $v_1, \dots, v_A$ . Algorithm  $\mathcal{B}$  chooses random  $u_i \in \mathbb{Z}_p$  for  $1 \leq i \leq B$ . We again define the subsets  $\hat{S}_i$  and  $S_i$  as

$$\hat{S}_i = S \cap \{(i-1)B+1, \dots, iB\} \quad \text{and} \quad S_i = \{x - iB + B \mid x \in \hat{S}_i\} \subseteq \{1, \dots, B\}$$

For  $i = 1, \dots, A$  algorithm  $\mathcal{B}$  sets  $v_i = g^{u_i} \left( \prod_{j \in S_i} g_{B+1-j} \right)^{-1}$ . It gives  $\mathcal{A}$  the public key

$$PK = (g, g_1, \dots, g_B, g_{B+2}, \dots, g_{2B}, v_1, \dots, v_A) \in \mathbb{G}^{2B+A}$$

Note that since  $g, \alpha$  and the  $u_i$  values are chosen uniformly at random, this public key has an identical distribution to that in the actual construction.

Next, the adversary needs all private keys that are not in the target set  $S$ . For all  $i \notin S$  we write  $i$  as  $i = (a-1)B + b$  for some  $1 \leq a \leq A$  and  $1 \leq b \leq B$ . Algorithm  $\mathcal{B}$  computes  $d_i$  as

$$d_i = g_b^{u_i} \cdot \prod_{j \in S_a} (g_{B+1-j+b})^{-1}$$

Indeed, we have that  $d_i = (g^{u_i} \prod_{j \in S_a} (g_{B+1-j})^{-1})^{(\alpha^b)} = v_a^{(\alpha^b)}$  as required. The main point is that since  $i \notin S$  we know that  $b \notin S_a$  and therefore algorithm  $\mathcal{B}$  has all the necessary values to compute  $d_i$ .

**Challenge** To generate the challenge,  $\mathcal{B}$  computes  $\text{Hdr}$  as  $(h, h^{u_1}, \dots, h^{u_A})$ . It then randomly chooses a bit  $b \in \{0, 1\}$  and sets  $K_b = Z$  and picks a random  $K_{1-b}$  in  $\mathbb{G}_1$ . It gives  $(\text{Hdr}, K_0, K_1)$  as the challenge to  $\mathcal{A}$ .

We claim that when  $Z = e(g_{B+1}, h)$  (i.e. the input to  $\mathcal{B}$  is a  $B$ -BDHE tuple) then  $(\text{Hdr}, K_0, K_1)$  is a valid challenge to  $\mathcal{A}$  as in a real attack. To see this, write  $h = g^t$  for some (unknown)  $t \in \mathbb{Z}_p$ . Then, for all  $i = 1, \dots, A$  we have

$$h^{u_i} = (g^{u_i})^t = (g^{u_i} \left( \prod_{j \in S_i} g_{B+1-j} \right)^{-1} \left( \prod_{j \in S_i} g_{B+1-j} \right))^t = (v_i \prod_{j \in S_i} g_{B+1-j})^t$$

Therefore, by definition,  $(h, h^{u_1}, \dots, h^{u_A})$  is a valid encryption of the key  $e(g_{B+1}, g)^t$ . Furthermore,  $e(g_{B+1}, g)^t = e(g_{B+1}, h) = Z = K_b$  and hence  $(\text{Hdr}, K_0, K_1)$  is a valid challenge to  $\mathcal{A}$ .

On the other hand, when  $Z$  is random in  $\mathbb{G}_1$  (i.e. the input to  $\mathcal{B}$  is a random tuple) then  $K_0, K_1$  are just random independent elements of  $\mathbb{G}_1$ .

**Guess** The adversary,  $\mathcal{A}$  outputs a guess  $b'$  of  $b$ . If  $b' = b$  the algorithm  $\mathcal{B}$  outputs 0 (indicating that  $Z = e(g_{B+1}, h)$ ). Otherwise, it outputs 1 (indicating that  $Z$  is random in  $\mathbb{G}_1$ ).

We see that if  $(g, h, \vec{y}_{g, \alpha, B}, Z)$  is sampled from  $\mathcal{R}_{BDHE}$  then  $\Pr[\mathcal{B}(g, h, \vec{y}_{g, \alpha, B}, Z) = 0] = \frac{1}{2}$ . If  $(g, h, \vec{y}_{g, \alpha, B}, Z)$  is sampled from  $\mathcal{P}_{BDHE}$  then  $|\Pr[\mathcal{B}(g, h, \vec{y}_{g, \alpha, B}, Z) = 0] - \frac{1}{2}| = \text{AdvBr}_{\mathcal{A}, n} \geq \epsilon$ . It follows that  $\mathcal{B}$  has advantage at least  $\epsilon$  in solving  $B$ -BDHE in  $\mathbb{G}$ . This concludes the proof of Theorem 3.1.  $\square$

Note that the proof of Theorem 3.1 does not use the random oracle model. The system can be proved secure using the weaker *computational*  $\ell$ -BDHE assumption (as opposed to decision  $\ell$ -BDHE), using the random oracle model. In that case the advantage of the simulator is at least  $\epsilon/q$ , where  $q$  is the maximum number of random oracle queries made by the adversary.

## 4 Applications

We describe how our system can be used for a number of specific applications. The first application, file sharing in encrypted file systems, is an example of broadcasts to small sets. The second application, encrypted email for large mailing lists, shows that the majority of the public-key can be shared by many broadcast systems so that the public-key for a new broadcast system is constant size. The third application, DVD content protection, is an example of broadcasts to large sets.

### 4.1 File Sharing in Encrypted File Systems

Encrypted file systems let users encrypt files on disk. For example, Windows EFS, encrypts the file contents using a file encryption key  $K_F$  and then places an encryption of  $K_F$  in the file header. If  $n$  users have access to the file, EFS encrypts  $K_F$  under the public keys of all  $n$  users and places the resulting  $n$  ciphertexts in the file header. Related designs can be found in the SiRiUS [GSMB03] and Plutus [KRS<sup>+</sup>03] file systems.

Abstractly, access control in an encrypted file system can be viewed as a broadcast encryption problem. The file system is the broadcast channel and the key  $K_F$  is broadcast (via the file header) to the subset of users that can access file  $F$ . Many encrypted file systems implement the straight forward broadcast system where the number of ciphertexts in the file header grows linearly in the number of users that can access the file. As a result, there is often a hard limit on the number of users that can access a file. For example, the following quote is from Microsoft’s knowledge base:

“EFS has a limit of 256KB in the file header for the EFS metadata. This limits the number of individual entries for file sharing that may be added. On average, a maximum of 800 individual users may be added to an encrypted file.”

A natural question is whether we can implement file sharing in an encrypted file system without resorting to large headers. Remarkably, there is no known combinatorial solution that performs better than the straight forward solution used in EFS. The broadcast system of Section 3.1 performs far better and provides a system with the following parameters:

- The public key (whose size is linear in  $n$ ) is stored somewhere on the file system. Even for a large organization of 100,000 users this file is only 4MB long (using a standard security parameter where each group element is 20 bytes).
- Each user is given a private key that contains only one group element.
- Each file header contains  $([S], C)$  where  $[S]$  is a description of the set  $S$  of users who can access  $F$  and  $C$  is a fixed size ciphertext consisting of only two group elements.

Since  $S$  tends to be small relative to  $n$  its shortest description is simply an enumeration of the users in  $S$ . Assuming 32-bit user ID’s, a description of a set  $S$  of size  $r$  takes  $4r$  bytes. Hence, the file header grows with the size of  $S$ , but only at a rate of 4 bytes per user. In EFS the header grows by one public key ciphertext per user. For comparison, we can accommodate sharing among 800 users using a header of size  $4 \times 800 + 40 = 3240$  bytes which is far less than EFS’s header size. Even if EFS were using short ElGamal ciphertexts on elliptic curves, headers would grow by 44 bytes per user which would result in headers that are 11 times bigger than headers in our system.

Our system has two more properties that make it especially useful for cryptographic access control. We describe these next.

**1. Incremental sharing.** Suppose a file header contains a ciphertext  $C = (C_0, C_1)$  which is the encryption of  $K_F$  for a certain set  $S$  of users. Let  $C_0 = g^t$  and  $C_1 = (v \cdot \prod_{j \in S} g_{n+1-j})^t$ . Suppose the file owner wishes to add access rights for some user  $u \in \{1, \dots, n\}$ . This is easy to do given  $t$ . Simply set  $C_1 \leftarrow C_1 \cdot g_{n+1-u}^t$ . Similarly, to revoke access rights for user  $u$  set  $C_1 \leftarrow C_1 / g_{n+1-u}^t$ .

This incremental sharing mechanism requires the file owner to remember the random value  $t \in \mathbb{Z}_p$  for every file. Alternatively, the file owner can embed a short nonce  $T_F$  in every file header and derive the value  $t$  for that file by setting  $t \leftarrow PRF_k(T_F)$  where  $k$  is a secret key known only to the file owner. Hence, changing access permissions can be done efficiently with the file owner only having to remember a single key  $k$ . Note that when access rights to a file  $F$  change it is sometimes desirable to pick a new key  $K_F^{new}$ . Modifying the existing header to encrypt a new  $K_F^{new}$  for the updated access list is just as easy.

**2. Incremental growth of the number of users.** In many cases a broadcast encryption system must be able to handle the incremental addition of new users. It is desirable to have a system that does not a-priori restrict the total number of users it can handle. Our system supports this by slowly expanding the public key as the number of users in the system grows. To do so, at system initialization the setup algorithm picks a large value of  $n$  (say  $n = 2^{64}$ ) that is much larger than the maximum number of users that will ever use the system. At any one time if there are  $j$  users in the system the public key will be  $g_{n-j+1}, \dots, g_n, g_{n+2}, \dots, g_{n+j}$ . Whenever a new user joins the system we simply add two more elements to the public key. Note that user  $i$  must also be given  $g_i$  as part of the private key and everything else remains the same.

## 4.2 Sending Encrypted Email to Mailing Lists

One interesting aspect of our system is that the public values  $\vec{y}_{g,\alpha,n} = (g_1, \dots, g_n, g_{n+2}, \dots, g_{2n})$  can be shared among many broadcast systems and  $\alpha$  can be erased. Suppose this  $\vec{y}_{g,\alpha,n}$  is distributed globally to a large group of users (for example, imagine  $\vec{y}_{g,\alpha,n}$  is pre-installed on every computer). Then creating a new broadcast system is done by simply choosing a random  $\gamma \in \mathbb{Z}_p$ , setting  $v = g^\gamma$ , and assigning private keys as  $d_i = g_i^\gamma$ . Since all broadcast systems use the same pre-distributed  $\vec{y}_{g,\alpha,n}$ , the actual public key for this new broadcast system is just one element,  $v$ . Theorem 3.1 shows that using the same  $\vec{y}_{g,\alpha,n}$  for many broadcast systems is secure.

We illustrate this property with an example of secure mailing lists. Sending out encrypted email to all members of a mailing list is an example of a broadcast encryption system. Suppose the global public vector  $\vec{y}_{g,\alpha,n}$  is shipped with the operating system and installed on every computer. We arbitrarily set  $n = 50,000$  in which case the size of  $\vec{y}_{g,\alpha,n}$  is about 2MB.

For every secure mailing list, the administrator creates a separate broadcast encryption system. Thus, every mailing list is identified by its public key  $v = g^\gamma$ . We assume the maximum number of members in a mailing list is less than  $n = 50,000$  (larger lists can be partitioned into multiple smaller lists). Each time a new user is added onto the list, the user is assigned a previously unused index  $i \in \{1, \dots, n\}$  and given the secret key  $d_i = g_i^\gamma$ . The broadcast set  $S$  is updated to include  $i$  and all mailing list members are notified of the change in  $S$ . Similarly, if a user with index  $j$  is removed from the list, then  $j$  is removed from the set  $S$  and all members are notified. We note that any member,  $i$ , does not need to actually store  $S$ . Instead, member  $i$  need only store the value  $\prod_{j \in S, j \neq i} g_{n+1-j+i}$  needed for decryption. The member updates this value every time a membership update message is sent. To send email to a mailing list with public key  $v$  the server simply does a broadcast encryption to the current set of members  $S$  using  $(\vec{y}_{g,\alpha,n}, v)$  as the public key.

In this mail system, the header of email messages sent to the list are constant size. Similarly,

membership update messages are of constant size. A mailing list member only needs to store two group elements for each list he belongs to (although we have to keep in mind the cost of storing  $\bar{y}_{g,\alpha,n}$  which is amortized over all mailing lists). It is interesting to compare our solution to one using an LKH scheme [WHA97, WGL98]. In LKH email messages are encrypted under a group key. Using this type of a system each update message contains  $O(\log(m))$  ciphertexts and private keys are of size  $O(\log(m))$  (per system) where  $m$  is the current group size. In our system, update messages and private user storage are much smaller.

### 4.3 Content Protection

Broadcast encryption applies naturally to protecting DVD content, where the goal is to revoke compromised DVD players. Recall that the public key in our system is needed for decryption and hence it must be embedded in the header of every DVD disk. Consequently, we are interested in minimizing the total length of the header and public key, namely minimize  $|\text{Hdr}| + |PK|$ .

Let  $n$  be the total number of DVD players (e.g.  $n = 2^{32}$ ) and let  $r$  be the number of revoked players. Let  $\ell_{\text{id}} = \log_2 n$  (e.g.  $\ell_{\text{id}} = 32$ ) and let  $\bar{k}$  be the size of a group element (e.g.  $\bar{k} = 160$  bits). Then using our  $\sqrt{n}$ -broadcast system ( $B = \sqrt{n}$ ) we can broadcast to sets of size  $n - r$  using the following parameters:

$$\text{priv-key-size} = 4\bar{k}, \quad \text{and} \quad |\text{Hdr}| + |PK| + |S| = 4\bar{k}\lceil\sqrt{n}\rceil + r\ell_{\text{id}}$$

In comparison, the NNL system [NNL01] and its derivatives [HS02, GST04] can broadcast to sets of size  $n - r$  using:

$$\text{priv-key-size} = O(k \log n), \quad \text{and} \quad \text{header-size} = O((k + \ell_{\text{id}}) \cdot r)$$

where  $k$  is the length of a symmetric key (e.g.  $k = 128$  bits). Note that the broadcast header grows by  $O(k + \ell_{\text{id}})$  bits per revoked player. With our system the broadcast header only grows by  $\ell_{\text{id}}$  bits per revoked player.

**Example.** Plugging in real numbers we obtain the following. When  $n = 2^{32}$ ,  $\bar{k} = 20$  bytes, and  $\ell_{\text{id}} = 4$  bytes, header size in our system is 5.12MB and each revocation adds 4 bytes to the header. In NNL-like systems, using  $k = 128$ -bit symmetric keys, each revocation adds about 40 bytes to the header, but there is no upfront 5MB fixed cost.

The best system is obtained by combining NNL with our system (using NNL when  $r < \sqrt{n}$  and our system when  $r > \sqrt{n}$ ). Thus, as long as things are stable, DVD disk distributors use NNL. In case of a disaster where, say, a DVD player manufacturer loses a large number of player keys, DVD disk distributors can switch to our system where the header size grows slowly beyond  $O(\sqrt{n})$ .

## 5 Chosen Ciphertext Secure Broadcast Encryption

We show how to extend the system of Section 3.1 to obtain chosen ciphertext security. The basic idea is to compose the system with the IBE system of [BB04] and then apply the ideas of [CHK04]. The resulting system is chosen ciphertext secure without using random oracles.

We will need a signature scheme ( $\text{SigKeyGen}, \text{Sign}, \text{Verify}$ ). We will also need a collision resistant hash function that maps verification keys to  $\mathbb{Z}_p$ . However, rather than carry the hash function

everywhere, we will simply assume that signature verification keys are encoded as elements of  $\mathbb{Z}_p$ . This greatly simplifies the notation.

As we will see, security of the CCA-secure broadcast system for  $n$  users is based on the  $(n+1)$ -BDHE assumption (as opposed to the  $n$ -BDHE assumption for the system of Section 3.1). Hence, to keep the notation consistent with Section 3.1 we will describe the CCA-secure system for  $n-1$  users so that security will depend on the  $n$ -BDHE assumption as before. The system works as follows:

**Setup**( $n-1$ ): Same as in Section 3.1. Pick a random generator  $g \in \mathbb{G}$  and random  $\alpha, \gamma \in \mathbb{Z}_p$ . Compute  $g_i = g^{(\alpha^i)} \in \mathbb{G}$  for  $i = 1, 2, \dots, 2n$  and  $v = g^\gamma \in \mathbb{G}$ . The public key is:

$$PK = (g, g_1, \dots, g_n, g_{n+2}, \dots, g_{2n}, v) \in \mathbb{G}^{2n+1}$$

The private key for user  $i \in \{1, \dots, n-1\}$  is set as:  $d_i = g_i^\gamma \in \mathbb{G}$ . The algorithm outputs the public key  $PK$  and the  $n-1$  private keys  $d_1, \dots, d_{n-1}$ .

**Encrypt**( $S, PK$ ): Run the *SigKeyGen* algorithm to obtain a signature signing key  $K_{\text{SIG}}$  and a verification key  $V_{\text{SIG}}$ . Recall that for simplicity we assume  $V_{\text{SIG}} \in \mathbb{Z}_p$ . Next, pick a random  $t$  in  $\mathbb{Z}_p$  and set  $K = e(g_{n+1}, g)^t \in \mathbb{G}$ . Set

$$C = \left( g^t, \left( v \cdot g_1^{V_{\text{SIG}}} \cdot \prod_{j \in S} g_{n+1-j} \right)^t \right) \in \mathbb{G}^2$$

$$\text{Hdr} = (C, \text{Sign}(C, K_{\text{SIG}}), V_{\text{SIG}})$$

and output the pair  $(\text{Hdr}, K)$ . Note that the only change to the ciphertext from Section 3.1 is the term  $g_1^{V_{\text{SIG}}}$  and the additional signature data.

**Decrypt**( $S, i, d_i, \text{Hdr}, PK$ ): Let  $\text{Hdr} = ((C_0, C_1), \sigma, V_{\text{SIG}})$ .

1. Verify that  $\sigma$  is a valid signature of  $(C_0, C_1)$  under the key  $V_{\text{SIG}}$ . If invalid, output ‘?’.
2. Otherwise, pick a random  $w \in \mathbb{Z}_p$  and compute

$$\hat{d}_0 = \left( d_i \cdot g_{i+1}^{V_{\text{SIG}}} \cdot \prod_{\substack{j \in S \\ j \neq i}} g_{n+1-j+i} \right) \cdot \left( v \cdot g_1^{V_{\text{SIG}}} \cdot \prod_{j \in S} g_{n+1-j} \right)^w \quad \text{and} \quad \hat{d}_1 = g_i g^w$$

3. Output  $K = e(\hat{d}_1, C_1) / e(\hat{d}_0, C_0)$ .

Correctness can be shown with a similar calculation to the one in Section 3.1. Note that private key size and ciphertext size are unchanged.

Unlike the system of Section 3.1, decryption requires a randomization value  $w \in \mathbb{Z}_p$ . This randomization ensures that for any  $i \in S$  the pair  $(\hat{d}_0, \hat{d}_1)$  is chosen from the following distribution

$$\left( g_{n+1}^{-1} \cdot \left( v \cdot g_1^{V_{\text{SIG}}} \cdot \prod_{j \in S} g_{n+1-j} \right)^r, g^r \right)$$

where  $r$  is uniform in  $\mathbb{Z}_p$ . Note that this distribution is independent of  $i$  implying that all members of  $S$  generate a sample from the same distribution. Although this randomization slows down decryption by a factor of two, it is necessary for the proof of security.

Before proving security we briefly recall that a signature scheme  $(\text{SigKeyGen}, \text{Sign}, \text{Verify})$  is  $(t, \epsilon, q_s)$  strongly existentially unforgeable if no  $t$ -time adversary who makes at most  $q_s$  signature queries is able to produce some new (message, signature) pair with probability at least  $\epsilon$ . A complete definition is given in, e.g., [CHK04]. The following theorem proves chosen ciphertext security.

**Theorem 5.1.** *Let  $\mathbb{G}$  be a bilinear group of prime order  $p$ . For any positive integer  $n$ , the broadcast encryption system above is  $(t, \epsilon_1 + \epsilon_2, n - 1, q_D)$  CCA-secure assuming the decision  $(t, \epsilon_1, n)$ -BDHE assumption holds in  $\mathbb{G}$  and the signature scheme is  $(t, \epsilon_2, 1)$  strongly existentially unforgeable.*

*Proof.* Suppose there exists a  $t$ -time adversary,  $\mathcal{A}$ , such that  $\text{AdvBr}_{\mathcal{A}, n-1} > \epsilon_1 + \epsilon_2$ . We build an algorithm,  $\mathcal{B}$ , that has advantage  $\epsilon_1$  in solving the  $n$ -BDHE problem in  $\mathbb{G}$ . Algorithm  $\mathcal{B}$  takes as input a random  $n$ -BDHE challenge  $(g, h, \vec{y}_{g, \alpha, n}, Z)$ , where  $\vec{y}_{g, \alpha, n} = (g_1, \dots, g_n, g_{n+2}, \dots, g_{2n})$  and  $Z$  is either  $e(g_{n+1}, h)$  or a random element of  $\mathbb{G}_1$  (recall that  $g_i = g^{\alpha^i}$ ). Algorithm,  $\mathcal{B}$ , proceeds as follows.

**Init.** Algorithm  $\mathcal{B}$  runs  $\mathcal{A}$  and receives the set  $S^*$  of users that  $\mathcal{A}$  wishes to be challenged on.

**Setup.**  $\mathcal{B}$  needs to generate a public key  $PK$  and private keys  $d_i$  for  $i \notin S^*$ . Algorithm  $\mathcal{B}$  first runs the  $\text{SigKeyGen}$  algorithm to obtain a signature signing key  $K_{\text{SIG}}^*$  and a verification key  $V_{\text{SIG}}^* \in \mathbb{Z}_p$ . Next, it chooses random  $\gamma \in \mathbb{Z}_p$  and sets  $v = g^\gamma g_1^{-V_{\text{SIG}}^*} \left( \prod_{j \in S^*} g_{n+1-j} \right)^{-1}$ . It gives  $\mathcal{A}$  the public key

$$PK = (g, g_1, \dots, g_n, g_{n+2}, \dots, g_{2n}, v) \in \mathbb{G}^{2n+1}$$

Note that since  $g, \alpha, \gamma$  are chosen uniformly at random, this public key has an identical distribution to that in the actual construction.

Next, the adversary needs all private keys that are not in the target set  $S^*$ . For  $i \notin S^*$  algorithm  $\mathcal{B}$  computes  $d_i$  as

$$d_i = g_i^\gamma \cdot g_{1+i}^{-V_{\text{SIG}}^*} \cdot \prod_{j \in S^*} (g_{n+1-j+i})^{-1}$$

Indeed, we have that

$$d_i = \left( g^\gamma \cdot g_1^{-V_{\text{SIG}}^*} \cdot \prod_{j \in S^*} (g_{n+1-j})^{-1} \right)^{(\alpha^i)} = v^{(\alpha^i)}$$

as required. The important thing to note is that since  $i \notin S^*$  algorithm  $\mathcal{B}$  has all the necessary values to compute  $d_i$ .

**Query phase 1.** Algorithm  $\mathcal{A}$  issues decryption queries. Let  $(u, S, \text{Hdr})$  be a decryption query where  $S \subseteq S^*$  and  $u \in S$ . Let  $\text{Hdr} = ((C_0, C_1), \sigma, V_{\text{SIG}})$ . Algorithm  $\mathcal{B}$  responds as follows:

1. Run *Verify* to check the signature  $\sigma$  on  $(C_0, C_1)$  using the verification key  $V_{\text{SIG}}$ . If the signature is invalid,  $\mathcal{B}$  responds with ‘?’.
2. If  $V_{\text{SIG}} = V_{\text{SIG}}^*$  algorithm  $\mathcal{B}$  outputs a random bit  $b \stackrel{\text{R}}{\leftarrow} \{0, 1\}$  and aborts the simulation.

3. Otherwise, the challenger picks a random  $r \in \mathbb{Z}_p$  and sets

$$\begin{aligned}\hat{d}_1 &= g^r \cdot g_n^{1/(V_{\text{SIG}} - V_{\text{SIG}}^*)} \\ \hat{d}_0 &= \hat{d}_1^\gamma \cdot g_1^{r(V_{\text{SIG}} - V_{\text{SIG}}^*)} \cdot \prod_{j \in S^* \setminus S} g_{n+1-j}^{-r} \cdot \prod_{j \in S^* \setminus S} g_{2n+1-j}^{-1/(V_{\text{SIG}} - V_{\text{SIG}}^*)}\end{aligned}$$

4.  $\mathcal{B}$  responds with  $K = e(\hat{d}_1, C_1)/e(\hat{d}_0, C_0)$ .

To see that  $\mathcal{B}$ 's response is as in a real attack game define  $\tilde{r} = r + \frac{\alpha^n}{V_{\text{SIG}} - V_{\text{SIG}}^*}$  and observe that

$$\hat{d}_0 = g_{n+1}^{-1} \cdot (v \cdot g_1^{V_{\text{SIG}}} \cdot \prod_{j \in S} g_{n+1-j})^{\tilde{r}} \quad \text{and} \quad \hat{d}_1 = g^{\tilde{r}}$$

Furthermore, since  $r$  is uniform in  $\mathbb{Z}_p$  we know that  $\tilde{r}$  is uniform in  $\mathbb{Z}_p$ . Thus,  $\mathcal{B}$ 's response is identical to  $\text{Decrypt}(S, u, d_u, \text{Hdr}, PK)$ , as required.

**Challenge.** To generate the challenge,  $\mathcal{B}$  computes

$$\begin{aligned}C &= (h, h^\gamma) \in \mathbb{G}^2 \\ \text{Hdr}^* &= (C, \text{Sign}(C, K_{\text{SIG}}^*), V_{\text{SIG}}^*)\end{aligned}$$

It then randomly chooses a bit  $b \in \{0, 1\}$  and sets  $K_b = Z$  and picks a random  $K_{1-b}$  in  $\mathbb{G}_1$ . It gives  $(\text{Hdr}^*, K_0, K_1)$  as the challenge to  $\mathcal{A}$ .

We claim that when  $Z = e(g_{n+1}, h)$  (i.e. the input to  $\mathcal{B}$  is an  $n$ -BDHE tuple) then  $(\text{Hdr}^*, K_0, K_1)$  is a valid challenge to  $\mathcal{A}$  as in a real attack. To see this, write  $h = g^t$  for some (unknown)  $t \in \mathbb{Z}_p$ . Then,

$$h^\gamma = g^{\gamma t} = \left( g^\gamma \cdot (g_1^{V_{\text{SIG}}^*} \prod_{j \in S} g_{n+1-j})^{-1} \cdot (g_1^{V_{\text{SIG}}} \prod_{j \in S} g_{n+1-j}) \right)^t = (v \cdot g_1^{V_{\text{SIG}}} \cdot \prod_{j \in S} g_{n+1-j})^t$$

Therefore, by definition,  $(h, h^\gamma)$  is a valid encryption of the key  $e(g_{n+1}, g)^t$ . Furthermore,  $e(g_{n+1}, g)^t = e(g_{n+1}, h) = Z = K_b$  and hence  $(\text{Hdr}, K_0, K_1)$  is a valid challenge to  $\mathcal{A}$ .

On the other hand, when  $Z$  is random in  $\mathbb{G}_1$  (i.e. the input to  $\mathcal{B}$  is a random tuple) then  $K_0, K_1$  are just random independent elements of  $\mathbb{G}_1$ .

**Query phase 2.** Same as in query phase 1.

**Guess.** The adversary,  $\mathcal{A}$  outputs a guess  $b'$  of  $b$ . If  $b' = b$  the algorithm  $\mathcal{B}$  outputs 0 (indicating that  $Z = e(g_{n+1}, h)$ ). Otherwise, it outputs 1 (indicating that  $Z$  is random in  $\mathbb{G}_1$ ).

We see that if  $(g, h, \vec{y}_{g, \alpha, n}, Z)$  is sampled from  $\mathcal{R}_{BDHE}$  then  $\Pr[\mathcal{B}(g, h, \vec{y}_{g, \alpha, n}, Z) = 0] = \frac{1}{2}$ . Now, let **abort** be the event that  $\mathcal{B}$  aborted during the simulation. Then, when  $(g, h, \vec{y}_{g, \alpha, n}, Z)$  is sampled from  $\mathcal{P}_{BDHE}$  we have

$$\left| \Pr[\mathcal{B}(g, h, \vec{y}_{g, \alpha, n}, Z) = 0] - \frac{1}{2} \right| \geq \text{AdvBr}_{\mathcal{A}, n-1} - \Pr[\text{abort}] > (\epsilon_1 + \epsilon_2) - \Pr[\text{abort}]$$

The first inequality follows from the fact that when  $(g, h, \vec{y}_{g,\alpha,n}, Z)$  is sampled from  $\mathcal{P}_{BDHE}$  the simulation is perfect when  $\mathcal{B}$  does not abort. It now follows that  $\mathcal{B}$  has advantage at least  $\epsilon_1 + \epsilon_2 - \Pr[\text{abort}]$  in solving  $n$ -BDHE.

To conclude the proof of Theorem 5.1 it remains to bound the probability that  $\mathcal{B}$  aborts the simulation as a result of one  $\mathcal{A}$ 's decryption queries. We claim that  $\Pr[\text{abort}] < \epsilon_2$ . Otherwise one can use  $\mathcal{A}$  to forge signatures with probability at least  $\epsilon_2$ . Briefly, we can construct another simulator that knows the private key,  $\gamma$ , but receives  $K_{\text{SIG}}^*$  as a challenge in an existential forgery game. In the above experiment,  $\mathcal{A}$  causes an abort by submitting a query that includes an existential forgery under  $K_{\text{SIG}}^*$  on some ciphertext. Our simulator is able to use this forgery to win the existential forgery game. Note that during the game the adversary makes only one chosen message query to generate the signature needed for the challenge ciphertext. Thus,  $\Pr[\text{abort}] < \epsilon_2$ .

It now follows that  $\mathcal{B}$ 's advantage is at least  $\epsilon_1$  as required. This completes the proof of Theorem 5.1.  $\square$

Note that the proof of Theorem 5.1 does not use the random oracle model implying that the system is chosen-ciphertext secure in the standard model.

We also note that instead of the signature-based method of [CHK04] we could have used the more efficient MAC-based method of [BK04]. We chose to present the construction using the signature method to simplify the proof. The MAC-based method would also work.

## 6 Conclusions and Open Problems

We presented the first fully collusion resistant broadcast encryption scheme with constant size ciphertexts and private keys for arbitrary receiver sets. In Section 5 we built a chosen-ciphertext secure broadcast system with the same parameters. A generalization of our basic scheme gave us a tradeoff between public key size and ciphertext size. With the appropriate parametrization we achieve a broadcast encryption scheme with  $O(\sqrt{n})$  ciphertext and public key size. We discussed several applications such as encrypted file systems and content protection.

We leave as an open problem the question of building a public-key broadcast encryption system with the same parameters as ours which is secure against adaptive adversaries. We note that any non-adaptive scheme that is  $(t, \epsilon, n)$  secure is also  $(t, \epsilon/2^n, n)$  secure against adaptive adversaries. However, in practice this reduction is only meaningful for small values of  $n$ .

Another problem is to build a tracing traitors system [CFN94] with the same parameters as our system. Ideally, one could combine the two systems to obtain an efficient trace-and-revoke system. Finally, it is interesting to explore alternate systems with similar performance that can be proved secure under a weaker assumption.

## References

- [BB04] Dan Boneh and Xavier Boyen. Efficient selective-ID identity based encryption without random oracles. In *Proceedings of Eurocrypt 2004*, LNCS, pages 223–238. Springer-Verlag, 2004.
- [BBG05] D. Boneh, X. Boyen, and E. Goh. Hierarchical identity based encryption with constant size ciphertext. In *Proceedings of Eurocrypt '05*, 2005. to appear.



- [BF01] Dan Boneh and Matt Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *Proceedings of Crypto 2001*, volume 2139 of *LNCS*, pages 213–29. Springer-Verlag, 2001.
- [BK04] Dan Boneh and Jonathan Katz. Improved efficiency for CCA-secure cryptosystems built using identity based encryption. Submitted for publication, 2004.
- [BS03] Dan Boneh and Alice Silverberg. Applications of multilinear forms to cryptography. *Contemporary Mathematics*, 324:71–90, 2003.
- [CFN94] Benny Chor, Amos Fiat, and Moni Naor. Tracing traitors. In *Proceedings of Crypto '94*, volume 839 of *LNCS*, pages 257–270. Springer, 1994.
- [CGI<sup>+</sup>99] Ran Canetti, Juan Garay, Gene Itkis, Daniele Micciancio, Moni Naor, and Benny Pinkas. Multicast security: A taxonomy and some efficient constructions. In *Proc. IEEE INFOCOM'99*, volume 2, pages 708–716, New York, NY, March 1999. IEEE.
- [CHK04] Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. In *Proceedings of Eurocrypt 2004*, *LNCS*, pages 207–222, 2004.
- [CMN99] Ran Canetti, Tal Malkin, and Kobbi Nissim. Efficient communication-storage tradeoffs for multicast encryption. In *Proceedings of Eurocrypt 1999*, pages 459–474, 1999.
- [DF02] Yevgeniy Dodis and Nelly Fazio. Public key broadcast encryption for stateless receivers. In *Proceedings of the Digital Rights Management Workshop 2002*, volume 2696 of *LNCS*, pages 61–80. Springer, 2002.
- [DF03] Y. Dodis and N. Fazio. Public key broadcast encryption secure against adaptive chosen ciphertext attack. In *Workshop on Public Key Cryptography (PKC)*, 2003.
- [DY05] Yevgeniy Dodis and Aleksandr Yampolskiy. A verifiable random function with short proofs and keys. In *Proceedings of the Workshop on Theory and Practice in Public Key Cryptography 2005*, 2005.
- [FN93] A. Fiat and M. Naor. Broadcast encryption. In *Proceedings of Crypto '93*, volume 773 of *LNCS*, pages 480–491. Springer-Verlag, 1993.
- [Gem97] Peter Gemmel. An introduction to threshold cryptography. *RSA CryptoBytes*, 2(3):7–12, 1997.
- [GSMB03] E. Goh, H. Shacham, N. Modadugu, and D. Boneh. Sirius: Securing remote untrusted storage. In *Proceedings of the Internet Society (ISOC) Network and Distributed Systems Security (NDSS) Symposium*, pages 131–145, 2003.
- [GST04] M. T. Goodrich, J. Z. Sun, , and R. Tamassia. Efficient tree-based revocation in groups of low-state devices. In *Proceedings of Crypto '04*, volume 2204 of *LNCS*, 2004.
- [GSW00] Juan A. Garay, Jessica Staddon, and Avishai Wool. Long-lived broadcast encryption. In *CRYPTO '00: Proceedings of the 20th Annual International Cryptology Conference on Advances in Cryptology*, pages 333–352. Springer-Verlag, 2000.

- [GSY99] Eli Gafni, Jessica Staddon, and Yiqun Lisa Yin. Efficient methods for integrating traceability and broadcast encryption. In *CRYPTO '99: Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*, pages 372–387. Springer-Verlag, 1999.
- [HS02] D. Halevy and A. Shamir. The lsd broadcast encryption scheme. In *Proceedings of Crypto '02*, volume 2442 of *LNCS*, pages 47–60, 2002.
- [JN03] Antoine Joux and Kim Nguyen. Separating decision Diffie-Hellman from Diffie-Hellman in cryptographic groups. *J. of Cryptology*, 16(4):239–247, 2003. Early version in Cryptology ePrint Archive, Report 2001/003.
- [Jou00] Antoine Joux. A one round protocol for tripartite Diffie-Hellman. In Wieb Bosma, editor, *Proceedings of ANTS IV*, volume 1838 of *LNCS*, pages 385–94. Springer-Verlag, 2000.
- [KRS<sup>+</sup>03] Mahesh Kallahalla, Erik Riedel, Ram Swaminathan, QianWang, and Kevin Fu. Plutus: Scalable secure file sharing on untrusted storage. In *Proceedings of the 2nd USENIX Conference on File and Storage Technologies (FAST)*, 2003.
- [NNL01] D. Naor, M. Naor, and J. Lotspiech. Revocation and tracing schemes for stateless receivers. In *Proceedings of Crypto '01*, volume 2139 of *LNCS*, pages 41–62, 2001.
- [NP00] M. Naor and B. Pinkas. Efficient trace and revoke schemes. In *Financial cryptography 2000*, volume 1962 of *LNCS*, pages 1–20. Springer, 2000.
- [Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In *Proceedings of Eurocrypt 1997*. Springer-Verlag, 1997.
- [SM03] Alan T. Sherman and David A. McGrew. Key establishment in large dynamic groups using one-way function trees. *IEEE Trans. Softw. Eng.*, 29(5):444–458, 2003.
- [ST98] Doug R. Stinson and Tran Van Trung. Some new results on key distribution patterns and broadcast encryption. *Des. Codes Cryptography*, 14(3):261–279, 1998.
- [Sti97] Doug R. Stinson. On some methods for unconditionally secure key distribution and broadcast encryption. *Des. Codes Cryptography*, 12(3):215–243, 1997.
- [SW98] D. R. Stinson and R. Wei. Combinatorial properties and constructions of traceability schemes and frameproof codes. *SIAM J. Discret. Math.*, 11(1):41–53, 1998.
- [WGL98] C. K. Wong, M. Gouda, and S. Lam. Secure group communications using key graphs. In *Proceedings of SIGCOMM '98*, 1998.
- [WHA97] D.M. Wallner, E.J. Harder, and R.C. Agee. Key management for multiast: Issues and architectures. IETF draft wallner-key, 1997.

## A Threshold Key Generation

In the file system application of Section 4.1 the administrator hands private keys to users. Note that the administrator can therefore decrypt all files on disk. A standard approach for mediating

the risk of a single trusted administrator is using threshold cryptography [Gem97]. We briefly sketch how the administrator's secret key  $\gamma$ , where  $v = g^\gamma$ , can be broken into  $m$  shares so that a private key  $d_i$  can be derived from  $t$  of them in a threshold fashion. For simplicity we assume a trusted dealer generates the  $m$  shares of  $\gamma$ .

**Setup.** Let  $\mathbb{G}$  be a bilinear group of order  $p$ . Pick a random generator  $g$  of  $\mathbb{G}$  and random  $\alpha, \gamma \in \mathbb{Z}_p$  and, as usual, define  $g_i = g^{\alpha^i}$  and  $v = g^\gamma \in G$ . Output the public key  $PK = \{g, g_1, \dots, g_n, g_{n+2}, \dots, g_{2n}, v\}$  as in Section 3.1.

Next, the dealer generates  $m$  shares of  $\gamma$ . We use Shamir secret sharing to generate the shares. Let  $f \in \mathbb{Z}_p[x]$  be a random polynomial of degree  $t - 1$  satisfying  $f(0) = \gamma$ . For  $j = 1, \dots, m$  the  $j$ 'th share of  $\gamma$  is defined as  $s_j = f(j) \in \mathbb{Z}_p$ . Note that for any subset of  $t$  shares  $\{s_1, \dots, s_t\}$  we have that  $\gamma = \sum_{i=1}^t \lambda_i s_i$  where  $\lambda_i$  are the appropriate Lagrange interpolation coefficients.

Now, suppose user  $k \in \{1, \dots, n\}$  wants her private key  $d_k = g_k^\gamma \in \mathbb{G}$ . She picks  $t$  administrator servers to help her generate  $d_k$ . Without loss of generality we assume she picks servers  $1, \dots, t$ . To generate  $d_k$  she does the following:

1. For  $i = 1, \dots, t$  she receives  $g_k^{s_i}$  from the  $i$ th administrator.
2. She computes her private key as  $d_k = \prod_{i=1}^t (g_k^{s_i})^{\lambda_i}$ . Then  $d_k = g_k^{\sum_{i=1}^t \lambda_i s_i} = g_k^\gamma$  as required.

As usual, we assume all these messages are sent between the administrators and a user are over a private channel.