Privacy in encrypted content distribution using private broadcast encryption

Adam Barth
abarth@cs.stanford.edu

Dan Boneh*
dabo@cs.stanford.edu

Brent Waters
bwaters@cs.stanford.edu

Abstract

In many content distribution systems it is important to both restrict access of content to authorized users and to protect the identities of these users. We discover that current systems for encrypting content to set of users are subject to attacks on user privacy. We propose a new mechanism, private broadcast encryption, to protect the privacy of users of encrypted file systems and content delivery systems. We construct a private broadcast scheme, with a strong privacy guarantee against an active attacker, while achieving ciphertext length, encryption time, and decryption time comparable with the non-private schemes currently used in encrypted file systems.

1 Introduction

In both large and small scale content distribution systems it is often important to make certain data available to only a select set of users. In commercial content distribution, for example, a company may wish for its digital media to be available only to paying users. On a smaller scale, suppose a department's faculty need to access the academic transcripts of graduate applicants. If electronic copies of the transcripts were stored on the department's fileserver, they should only be accessible by the faculty and other users (e.g. other students).

It is often equally important to protect the identities of the users who are able to access protected content. The clientele of a website that distributes adult material likely would wish to keep their identities private. Commercial sites will often not want to disclose identities of customers because competitors might use this information for targeted advertising. If an employee is up for promotion, a company might wish to hide who is on his promotion committee and therefore who is able to read his performance evaluation file.

The most commonly used method for protecting both electronic content and the privacy of users who can access it is to employ a trusted server. Whenever a user wishes to access content, the user contacts the server, authenticates him or herself, and is sent the content over a secure channel. As long as the server behaves correctly, only authorized users will be able to access the content and which users are authorized to access which content will not be divulged, even to other authorized users.

While this simple method of data protection is adequate for some applications, it has some significant drawbacks. First, both data content and user privacy are subject to attack if the server is compromised. Additionally, content providers will often not distribute their data directly, but for economic reasons outsource distribution to third parties or use peer-to-peer networks. In this case, the content owners will no longer be directly in control of data distribution.

*Supported by NSF.

(a) $A : \{K_F\}_{K_A}; B : \{K_F\}_{K_B}; C : \{K_F\}_{K_C}; \{F\}_{K_F}$

(b) $\{K_F\}_{K_B}; \quad \{K_F\}_{K_C}; \quad \{K_F\}_{K_A}; \{F\}_{K_F}$

Figure 1: Simple constructions of broadcast encryption systems. File F is encrypted under the key K_F, which in turn is encrypted under the public keys of users A, B, and C. (a) The scheme typically used by encrypted file systems reveals the set of users authorized to access F. (b) Modifying this scheme by removing the labels, using a key-private cryptosystem, and randomly reordering the users yields a private broadcast scheme resistant to passive attacks on recipient privacy. These simple schemes are both vulnerable to active attacks, however.

For these reasons we examine the problem of efficiently distributing encrypted content in such a way that only authorized users can read the content and that the identities of authorized users is hidden. We study this problem for the case of encrypted file systems. However, our results can be generalized to larger content distribution systems.

Encrypted File Systems. Encrypted file systems implement read access control by encrypting the contents of files such that only users with read permission will be able to perform decryption. Typical encrypted file systems, such as Windows EFS, encrypt each file under its own symmetric key, K_F, and then encrypt the symmetric key separately under the public keys of the users authorized to access the file (Figure 1(a)).

While these systems protect the content of the file from unauthorized users, they do little to protect the identities of users allowed to access the file. Who can access a file, however, is often more sensitive than the contents of the file itself. Suppose, for example, a university provides a document on its file server concerning a substance abuse program to the students enrolled in the program. To maintain the privacy of the students, the set of authorized users should be kept private, not only from outsiders, but from the students in the group as well.

Current implementations expose the identities of authorized users in two different ways. First, the individual public key encryptions of the symmetric key, K_F, are labeled with the identity of the user as shown in Figure 1(a). This is done so that an authorized user A can quickly locate the encryption of K_F encrypted under A's public key. Second, even if these labels were removed, an adversary can examine the actual ciphertexts to learn information about the user's identity. For example, suppose an attacker wants to determine whether user A or user B has access to a particular file. Further, suppose user A has a 1024-bit key while user B has a 2048-bit key. Then, by examining the encryption of K_F, specifically the ciphertext length, an attacker can easily determine which of the two has access. Thus, the encryptions of K_F leak some information about who has access to the file.

Our goal is to provide recipient privacy—an encrypted file should hide who can access the content. We approach the problem of recipient privacy by introducing a notion we call *private broadcast encryption*. A private broadcast encryption scheme is used to encrypt a message to several recipients while hiding the identities of the recipients, even from each other.

The most straightforward construction of a private broadcast encryption system is to modify the scheme currently used in encrypted file systems by removing the identifying labels and using a public key system that does not reveal the public key associated with a ciphertext, such as ElGamal or Cramer-Shoup [1] (Figure 1(b)). While this scheme is secure against passive attacks on recipient

(a) $c_1; c_2; c_3; \{F\}_{K_F}$ (b) $c_1; c_2; c_3; \{F'\}_{K_F}$

Figure 2: Active attack on recipient privacy. (a) The sensitive document, F, encrypted for three recipients. If the attacker is a recipient, he or she learns K_F. (b) The malicious document created by an attacker, which can be decrypted by the same users as the original document, contains F' of the attacker's choice. Recipients of the original document can be discovered by tricking them into decrypting the malicious document.

privacy, an active attacker can mount a chosen-ciphertext attack and learn whether or not a user can decrypt a message.

Returning to our example, consider an active attacker who is authorized to decrypt the substance support group document, where the list of authorized users should be private. Now, suppose that the attacker wishes to determine whether Alice can read the document. Because the attacker is a legitimate recipient, he or she knows K_F and can maliciously prepare a different encrypted file by replacing the encrypted contents of the original file with content of the attacker's choice, encrypted under K_F. Alice is able to read this maliciously created file if, and only if, she can read the original file. For example, a malicious legitimate recipient of the substance abuse document could copy the document header, but replace the document body with the message "please visit the following URL for free music," as illustrated in Figure 2. This will expose the members of the substance support group because they are the only ones who can read the message and visit the given URL.

While we could avoid this attack by giving separate encryptions for each user of the bulk data, this would greatly increase the overall storage demands, as the contents of each file would need to be replicated for each authorized user. We solve this problem by building efficient *private broadcast encryption* systems that are secure under chosen-ciphertext attacks. Our construction achieves storage space, encryption time, and decryption time comparable to schemes currently employed in encrypted file systems.

The remainder of the paper is organized as follows. We define private broadcast encryption in Section 2, giving a game definition of recipient privacy under a chosen-ciphertext attack. In Section 3, we examine the PGP encryption system and demonstrate attacks against recipient privacy. We present our private broadcast encryption constructions in Section 4. Finally, we conclude in Section 5.

1.1 Related work

The notion of key privacy in the public key setting was first formalized by Bellare et. al. [1]. A public key encryption system is key-private if ciphertexts do not leak information about the public keys for which they were encrypted. Specifically, an adversary viewing a chosen message encrypted under one of two public keys is unable to guess (with non-negligible advantage) which public key was used to produce the ciphertext. The authors formalize these definitions for key privacy under chosen-plaintext attack (IK-CPA) and chosen-ciphertext attacks (IK-CCA). They show that ElGamal and Cramer-Shoup are secure under these definitions, respectively, when public keys share a common prime moduli.

Our constructions will use a key-private public key system as a component in building a private broadcast encryption system. One interesting observation is that the straightforward construction of a private broadcast encryption scheme using an IK-CCA secure encryption scheme does not result in a private broadcast encryption system resistant to chosen-ciphertext attacks.

Previous work on broadcast encryption has focused on increasing collusion resistance and reducing the length of the ciphertext [6, 11, 10]. We differ from these works in that we focus on maintaining the privacy of users, but do not attempt to achieve ciphertext overhead that is sub-linear in the number of users. Whether private broadcast encryption systems can be realized with smaller ciphertext overhead is an open problem.

2 Private broadcast encryption

In this section, we define private broadcast encryption in terms of its correctness and security properties. A private broadcast encryption system consists of four algorithms.

- $I \leftarrow \mathsf{Setup}(\lambda)$. Given a security parameter λ, generates global parameters I for the system.
- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Keygen}(I)$. Given the global parameters I, generates public-secret key pairs.
- $C \leftarrow \mathsf{Encrypt}(S, M)$. Given a set of public keys $S = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_n\}$ generated by $\mathsf{Keygen}(I)$ and a message M, generates a ciphertext C.
- $M \leftarrow \mathsf{Decrypt}(\mathsf{sk}, C)$. Given a ciphertext C and a secret key sk, returns M if the corresponding public key $\mathsf{pk} \in S$, where S is the set used to generate C. $\mathsf{Decrypt}$ can also return $\perp$ if $\mathsf{pk} \notin S$ or if C is malformed.

Note that users can run the Keygen algorithm to generate public-secret key pairs for themselves. For ElGamal-like systems the global parameters I simply contain the prime p and generator $g \in \mathbb{Z}_p$.

The definition above departs from the standard definition of broadcast encryption in that the standard definition explicitly provides S, the set of recipients, to the $\mathsf{Decrypt}$ algorithm. Here we omit this parameter in order to capture systems that hide S. There is no loss of generality, however, as S can be included in the ciphertext, C, directly. We use the standard definition of semantic security of a broadcast encryption system (see for example [4]).

Recipient privacy. We define a notion of recipient privacy under a chosen-ciphertext attack for private broadcast encryption systems using a game between a challenger and an adversary. This game captures the fact that the adversary cannot distinguish a ciphertext intended for recipient set S_0 from a ciphertext intended for recipient set S_1. We require that S_0 and S_1 have the same size so that the ciphertext length will not give away the intended set. To model a chosen-ciphertext attack we allow the adversary to issue decryption queries. More precisely, the game defining privacy of a private broadcast encryption system is as follows:

Init: The challenger runs $I \leftarrow \mathsf{Setup}(\lambda)$ and gives the adversary the global parameters I. The adversary outputs $S_0, S_1 \subseteq \{1, \ldots, n\}$ such that $|S_0| = |S_1|$.

Setup: The challenger generates keys for each potential recipient, $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{Keygen}(I)$, and sends to the adversary each pk_i for $i \in S_0 \cup S_1$ as well as each sk_i for $i \in S_0 \cap S_1$.

Phase 1: The adversary makes decryption queries of the form (u, C) and the challenger returns the decryption $\mathsf{Decrypt}(\mathsf{sk}_u, C)$. The adversary may repeat this step as desired.

Challenge: The adversary gives the challenger a message M. The challenger picks a random $b \in \{0, 1\}$, runs $C^* \leftarrow \mathsf{Encrypt}(\{\mathsf{pk}_i \mid i \in S_b\}, M)$, and sends ciphertext C^* to the adversary.

Phase 2: The adversary makes more decryption queries, with the restriction that the query ciphertext $C \neq C^*$. The adversary may repeat this step as desired.

Guess: The adversary outputs its guess $b' \in \{0, 1\}$.

We say that the adversary wins the game if $b' = b$.

Def. A private broadcast encryption system is (t, q, n, ϵ)-CCA-Recipient-Private if, for all t-time adversaries A, the probability A wins the above game using recipient sets of size at most n and making at most q decryption queries is at most $1/2 + \epsilon$.

Def. A private broadcast encryption system is (t, n, ϵ)-CPA-Recipient-Private if it is $(t, 0, n, \epsilon)$-CCA-Recipient-Private.

A standard hybrid argument [2] shows that our definition also implies unlinkability among sets of ciphertexts. We also observe our definition of recipient privacy allows C to leak the number of recipients, just as semantic security allows a ciphertext to leak the length of the plaintext. If we wish to hide the number of recipients we can always pad the recipient set to a given size using dummy recipients.

Just as public key encryption is a special case of broadcast encryption, key privacy is a special case of recipient privacy. In key privacy [1] the adversary is restricted to $n = 1$, that is to using recipient sets S_0 and S_1 of size 1, mirroring the restriction on the public key $\mathsf{Encrypt}$ algorithm to encrypting only for a single recipient. Therefore, the IK-CCA definition is equivalent to our recipient privacy definition with $n = 1$.

3 Broadcast encryption in practice

In this section, we make our discussion of privacy problems in broadcast encryption systems concrete by examining two broadcast encryption systems used in practice. We study the widely used OpenPGP [13] encryption standard and the GNU Privacy Guard (GPG) [15] implementation.

3.1 The PGP encryption system

While OpenPGP is commonly associated with encrypted email, it can be used as a general encryption system. When encrypting a message to multiple recipients, OpenPGP functions as a broadcast encryption system: it encrypts each message under a symmetric key K and then encrypts K to each user using his or her public key. Either ElGamal or RSA encryption can be used for the public key encryption.

```
C:\gpg>gpg --verbose -d message.txt
gpg: armor header: Version: GnuPG v1.2.2 (MingW32)
gpg: public key is 3CF61C7B
gpg: public key is 028EAE1C
```

Figure 3: Transcript of an attempted GPG decryption of a file encrypted for two users. The identities of the users are completely exposed by their key IDs. These key IDS can then be translated to real identities by a reverse lookup on a public key directory.

Key IDs and recipient privacy. In standard operation, GPG completely exposes recipient identities. Figure 3 contains a transcript of an attempted GPG decryption of a message created with a PGP implementation. The message reveals the key IDs of two BCC recipients. A key's ID is essentially its hash. PGP uses key IDs for two purposes. First, public keys in the Web of Trust are indexed by key ID. For example, the MIT PGP Public Key Server [9], when queried for a specific name, returns the key ID, date, name, and email address of principals with the specified name. A principal's public key can then be retrieved by querying the server by key ID. Second, key IDs are used in ciphertexts to label encryptions of the message key (Figure 1(a)). These labels speed decryption because the decryptor knows his or her key ID and is able to locate the encryption of the message key he or she is able to decrypt. Unfortunately, attackers also know key IDs. Moreover, after examining a ciphertext, an attack need only query a public key server to learn the full name and email address of the owner of the associated public key.

Throwing key IDs. The OpenPGP standard allows implementation to omit key IDs from ciphertexts by replacing them with zeros (ostensibly to foil traffic analysis [5]). This option is available in GPG using the `--throw-keyids` command line option, but is disabled by default and thus will not be used if the command is not given. Omitting key IDs increases the amount of work required to decrypt a message. A message without key IDs, encrypted to n recipients, contains n unidentified ciphertexts. To decrypt the message, every recipient must attempt to decrypt each ciphertext, thus performing, on average, $n/2$ decryption operations.

Even when omitting key IDs, GPG does not achieve recipient privacy. When GPG generates an ElGamal public key, it does so in the group of integers modulo a random prime. Thus, different principals are very likely to have public keys in different groups, making GPG encryptions vulnerable to passive key privacy attacks. These attacks can be directly translated into attacks on CPA recipient privacy. GPG could defend against these attacks by using the same prime for every public key, for example one standardized by NIST [12].

Active attack. While omitting key IDs and standardizing the group used for public keys achieves CPA recipient privacy, it would not achieve CCA recipient privacy. An active attacker could determine the recipients as follows. Suppose Charlie, the attacker, received the encrypted message $\{K\}_{K_A}||\{K\}_{K_C}||\{M\}_K$ and wishes to determine whether Alice or Bob was the other recipient. As Charlie possesses his secret key K_C^{-1}, he can recover K, the message key. He can then encrypt a new message M' for the same recipient as the original message, $\{K\}_{K_A}||\{M'\}_K$, by copying the first portion of the header and encrypting M' under K. When Alice decrypts this message, she will obtain M', whereas when Bob decrypts this message, he will not obtain M'.

This type of attack is potentially much more dangerous than the passive attack in practice. If an attacker wishes to determine a recipient from a large pool of recipients, the passive attack will likely only eliminate some fraction of them. However, in an active attack the attacker could probe each of the potential receivers individually and learn exactly which ones were recipients.

4 Constructions

In this section, we present two constructions for private broadcast encryption that achieve CCA recipient privacy. The first construction is a generic construction from any asymmetric key encryption scheme that has key indistinguishability from chosen-ciphertext attacks (IK-CCA) [1]. The disadvantage of this first scheme is that decryption time is linear in the number of recipients because the decryption algorithm must try each ciphertext component until it successfully decrypts.

Our second construction is a specialized system in which the decryption algorithm performs one asymmetric key operation and uses the result to find the ciphertext component intended for it (if one exists). This construction is more efficient for decryptors than the first because no trial decryptions are required. We describe our two schemes and give intuition for their security. Formal proofs are given in the appendices.

Both constructions require the underlying public key scheme to be *strongly correct*. Essentially, a public key scheme is strongly correct if decrypting a ciphertext encrypted for one key with another key results in $\perp$, the reject symbol, with high probability. While this property is not ensured by the standard public key definitions, most CCA-secure cryptosystems, such as Cramer-Shoup, are strongly correct. Before giving a formal definition of strong correctness, we define a function that generates a random encryption of a given message and then returns the decryption of that ciphertext with a different random key.

$\mathsf{Test}(M)$**:** $I \leftarrow \mathsf{Init}(\lambda)$; $(\mathsf{pk}_0, \mathsf{sk}_0) \leftarrow \mathsf{Gen}(I)$; $C \leftarrow \mathsf{Enc}_{\mathsf{pk}_0}(M)$; $(\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathsf{Gen}(I)$; Return $\mathsf{Dec}_{\mathsf{sk}_1}(C)$.

Def. A public key scheme $(\mathsf{Init}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is ϵ-strongly-correct if, for all M, the probability $\mathsf{Test}(M) \neq \perp$ is at most ϵ.

4.1 Generic CCA recipient private construction

We realize our first construction by modifying the simple CPA recipient private construction (Figure 1(b)). First, the encryption algorithm uses a public key encryption scheme that has key-indistinguishability under CCA attacks (IK-CCA) to encrypt the ciphertext component for each recipient. Second, Encrypt generates a random signature and verification key for a one-time, strongly[1] unforgeable signature scheme [8, 14] such as RSA full-domain hash. The encryption algorithm includes the verification key in each public key encryption and then signs the entire ciphertext with the signing key.

The decryption algorithm attempts to decrypt each ciphertext component. If the public key decryption is successful (i.e. returns non-$\perp$), Decrypt will continue decryption only if the signature verifies under the extracted verification key. Intuitively, an adversary cannot extract a ciphertext component from the challenge ciphertext and use it in another ciphertext because it will be unable

[1] In a strongly unforgeable signature scheme, an adversary cannot output a new signature, even on a previously signed message.

to sign the new ciphertext under the same verification key. We now give a formal description of our scheme.

Given a strongly-correct, IK-CCA public key scheme $(\mathsf{Init}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, a strongly existentially unforgeable signature scheme $(\mathsf{Sig\text{-}Gen}, \mathsf{Sig}, \mathsf{Ver})$, and semantically secure symmetric key encryption and decryption algorithms (E, D), we construct a private broadcast encryption system as follows.

$\mathsf{Setup}(\lambda)$**:** Return $\mathsf{Init}(\lambda)$.

$\mathsf{Keygen}(I)$**:** For each user i, run $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{Gen}(I)$, return $(\mathsf{pk}_i, \mathsf{sk}_i)$ to user i, and publish pk_i.

$\mathsf{Encrypt}(S, M)$**:**

1. $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{Sig\text{-}Gen}(\lambda)$.
2. Choose a random symmetric key K.
3. For each $\mathsf{pk} \in S$, $c_{\mathsf{pk}} \leftarrow \mathsf{Enc}_{\mathsf{pk}}(\mathsf{vk}||K)$.
4. Let C_1 be the concatenation of the c_{pk}, in random order.
5. $C_2 \leftarrow E_K(M)$.
6. $\sigma \leftarrow \mathsf{Sig}_{\mathsf{sk}}(C_1||C_2)$.
7. Return the ciphertext $C = \sigma||C_1||C_2$.

$\mathsf{Decrypt}(\mathsf{sk}, C)$**:** Parse C as $\sigma||C_1||C_2$ and $C_1 = c_1||\cdots||c_n$. For each $i \in \{1, \ldots, n\}$:

1. $p \leftarrow \mathsf{Dec}(\mathsf{sk}, c_i)$.
2. If p is $\perp$, then continue to the next i.
3. Otherwise, parse p as $\mathsf{vk}||K$.
4. If $\mathsf{Ver}_{\mathsf{vk}}(C_1||C_2, \sigma)$, return $M = D_K(C_2)$.

If none of the c_i decrypt and verify, return $\perp$.

Notice the time taken by $\mathsf{Decrypt}$ to execute could leak information. Recipient privacy relies on the attacker being unable to determine whether a decryption fails because $p = \perp$ or because the signature did not verify. Implementations must take care to prevent such timing attacks. We state our main theorem as follows. We prove it in Appendix A.

Theorem 1. *If* $(\mathsf{Init}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is both* ϵ_1*-strongly-correct and* (t, q, ϵ_2)*-CCA-key-private and* $(\mathsf{Sig\text{-}Gen}, \mathsf{Sig}, \mathsf{Ver})$ *is* $(t, 1, \epsilon_3)$*-strongly-existentially-unforgeable, the above construction is* $(t, q, n, n(\epsilon_1 + \epsilon_2 + \epsilon_3))$*-CCA-recipient-private.*

The semantic security of our scheme follows in a straightforward manner. Because our scheme achieves broadcast encryption by concatenating public key encryptions, each user can generate his or her own public key and have an authority issue a certificate binding it to his or her identity.

4.2 CCA recipient privacy with efficient decryption

To decrypt a ciphertext in the CCA recipient private scheme above, a recipient must attempt to decrypt $n/2$ components of the ciphertext, on average, where n is the number of recipients. Non-private schemes improve performance by labeling ciphertext components with recipient identities, directing the attention of decryptors to appropriate ciphertext components. However, these labels reveal the identities of the recipients.

In this section, we construct a private broadcast encryption system that requires only a constant number of cryptographic operations in order to decrypt, regardless of the number of recipients. To achieve this we use a group G where the computational Diffie-Hellman problem is believed to be hard, but there exists an efficient algorithm for testing Diffie-Hellman tuples. For example, we could use groups with efficiently computable bilinear maps [7, 3].

Our scheme is similar to the previous one with small modifications. First, each user i in this scheme has a public key value g^{a_i}, for which he or she knows the exponent a_i, in addition to the public key for the encryption scheme. The encryption algorithm first chooses a random exponent r and labels the ciphertext component for user i with $H(g^{ra_i})$, where the hash function H is viewed as a random oracle. When decrypting, user i first calculates $H(g^{ra_i})$ and then uses the result to locate the ciphertext component encrypted for him or her. User i need only perform one public key decryption to recover the message.

Let G be a group, with generator g, where the computational Diffie-Hellman problem (CDH) is hard and the decisional Diffie-Hellman problem (DDH) is easy and let $H : G \to \{0,1\}^\lambda$ be a hash function that is modeled as a random oracle (for some security parameter λ). Given a strongly correct, CCA-key-private public key scheme $(\mathsf{Init}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, a strongly existentially unforgeable signature scheme $(\mathsf{Sig\text{-}Gen}, \mathsf{Sig}, \mathsf{Ver})$, and semantically secure symmetric key encryption and decryption algorithms (E, D), we construct a private broadcast encryption system as follows.

$\mathsf{Setup}(\lambda)$**:** Return $\mathsf{Init}(\lambda)$.

$\mathsf{Keygen}(I)$**:** For each user i, run $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{Gen}(I)$ and choose a random exponent a_i. Let $\mathsf{pk}'_i = (\mathsf{pk}_i, g^{a_i})$ and $\mathsf{sk}'_i = (\mathsf{sk}_i, a_i)$. Return $(\mathsf{pk}'_i, \mathsf{sk}'_i)$ to user i and publish pk'_i.

$\mathsf{Encrypt}(S, M)$**:**

1. $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{Sig\text{-}Gen}(\lambda)$.
2. Choose a random symmetric key K.
3. Choose a random exponent r and set $T = g^r$.
4. For each $(\mathsf{pk}, g^a) \in S$, $c_{\mathsf{pk}} \leftarrow H(g^{ra})||\mathsf{Enc}_{\mathsf{pk}}(\mathsf{vk}||g^{ra}||K)$.
5. Let C_1 be the concatenation of the c_{pk}, ordered by their values of $H(g^{ra})$.
6. $C_2 \leftarrow E_K(M)$
7. $\sigma \leftarrow \mathsf{Sig}_{\mathsf{sk}}(T||C_1||C_2)$.
8. Return the ciphertext $C = \sigma||T||C_1||C_2$.

$\mathsf{Decrypt}((\mathsf{sk}, a), C)$**:** Parse C as $\sigma||T||C_1||C_2$ and $C_1 = c_1||\cdots||c_n$.

1. Calculate $l = H(T^a) = H(g^{ra})$.
2. Find c_j such that $c_j = l||c$. If no such j exists, return $\perp$ and stop.

3. Calculate $p \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$.
4. If p is $\perp$, return $\perp$ and stop.
5. Otherwise, parse p as $\mathsf{vk}||x||K$.
6. If $x \neq T^a$, return $\perp$ and stop.
7. If $\mathsf{Ver}_{\mathsf{vk}}(T||C_1||C_2, \sigma)$, return $M = D_K(C_2)$; otherwise, return $\perp$.

Observe that the DDH algorithm is not used in either the encryption or decryption algorithms. It is needed only by the simulator in our proof, given in Appendix B.

Theorem 2. *If* $(\mathsf{Init}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is* ϵ_1*-strongly-correct,* (t, q, ϵ_2)*-CCA-semantically-secure and* (t, q, ϵ_3)*-CCA-key-private,* $(\mathsf{Sig\text{-}Gen}, \mathsf{Sig}, \mathsf{Ver})$ *is* $(t, 1, \epsilon_4)$*-strongly-existentially-unforgeable, CDH is* (t, ϵ_5)*-hard in* G*, and DDH is efficiently computable in* G*, then the above construction is* $(t, q, n, n(\epsilon_1 + 2\epsilon_2 + \epsilon_3 + \epsilon_4 + 2\epsilon_5))$*-CCA-recipient-private.*

5 Conclusions

In many content distribution applications it is important to protect both the content being distributed and the identities of users allowed to access content. Currently, encrypted file systems fail to protect the privacy of users. User privacy is compromised because the underlying encryption methods disclose the identities of a ciphertext's recipients. Many such systems simply give away the identities of the users in the form of labels attached to the ciphertext. Additionally, those systems that attempt to avoid disclosing the recipient's identity, such as GnuPG, are vulnerable to having their user's privacy compromised by a new chosen-ciphertext attack that we introduced.

Our proposed mechanism, private broadcast encryption, enables the efficient encryption of a message to multiple recipients without revealing the identities of the recipients of the message, even to other recipients. We presented two constructions of private broadcast encryption systems. Both of these satisfy a strong definition of recipient privacy in the face of active attacks. The second additionally achieves decryption in a constant number of cryptographic operations, performing comparably to current systems that do not provide user privacy.

References

[1] M. Bellare, A. Boldyreva, A. Desai, and D. Pointcheval. Key-privacy in public-key encryption. In *ASIACRYPT '01: Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security*, pages 566–582. Springer-Verlag, 2001.

[2] M. Bellare, A. Boldyreva, and S. Micali. Public-key encryption in a multi-user setting: Security proofs and improvements. In *Proceedings of Eurocrypt 2000*, volume 1807 of *LNCS*, page 259, 2000.

[3] D. Boneh and M. K. Franklin. Identity-based encryption from the Weil pairing. In *CRYPTO '01: Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology*, pages 213–229, London, UK, 2001. Springer-Verlag.

[4] D. Boneh, C. Gentry, and B. Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In *CRYPTO '05*, 2005.

[5] J. Callas, L. Donnerhacke, H. Finney, and R. Thayer. RFC 2440: OpenPGP message format, 1998. http://www.ietf.org/rfc/rfc2440.txt.

[6] A. Fiat and M. Naor. Broadcast encryption. In *CRYPTO '93: Proceedings of the 13th Annual International Cryptology Conference on Advances in Cryptology*, pages 480–491, New York, NY, USA, 1994. Springer-Verlag New York, Inc.

[7] A. Joux and K. Nguyen. Separating Decision Diffie-Hellman from Diffie-Hellman in cryptographic groups. Technical Report eprint.iacr.org/2001/003, 2001.

[8] L. Lamport. Constructing digital signatures from a one way function. Technical report, SRI International, 1979.

[9] MIT. MIT PGP public key server, 2005. http://pgpkeys.mit.edu/.

[10] 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.

[11] M. Naor and B. Pinkas. Efficient trace and revoke schemes. In *Financial cryptography 2000*, volume 1962 of *LNCS*, pages 1–20. Springer-Verlag, 2000.

[12] National Institute of Standards and Technology. Digital signature standard (DSS), 2000. http://www.csrc.nist.gov/publications/fips/.

[13] OpenPGP. The OpenPGP alliance home page, 2005. http://www.openpgp.org/.

[14] J. Rompel. One-way functions are necessary and sufficient for secure signatures. In *STOC '90: Proceedings of the Twenty-Second Annual ACM Symposium on Theory of Computing*, pages 387–394, New York, NY, USA, 1990. ACM Press.

[15] Werner Koch. The gnu privacy guard, 2005. http://www.gnupg.org/.

A Proof of first construction

Assume $(\mathsf{Init}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is both ϵ_1-strongly-correct and (t, q, ϵ_2)-CCA-key-private. Assume $(\mathsf{Sig\text{-}Gen}, \mathsf{Sig}, \mathsf{Ver})$ is $(t, 1, \epsilon_3)$-strongly-existentially-unforgeable. We first prove a lemma that the scheme is CCA-recipient-private when the adversary selects recipient sets that differ by only one recipient. The general case then follows by a hybrid argument.

Claim 3. *For all t-time adversaries A, the probability A, in the recipient privacy game using the construction from Section 4.1, makes a decryption query containing a signature that verifies with* vk *from the challenge ciphertext is at most ϵ_3.*

Proof. Given an adversary A that makes a decryption query with a forged signature with probability greater than ϵ_3, we construct a machine B that breaks the $(t, 1, \epsilon_3)$-strong-existential-unforgeability of the signature scheme as follows.

The algorithm B first receives a forgeability challenge vk. Next, B exactly simulates Init and Setup, generating all public-secret key pairs itself. If any signature in Phase 1 is a forgery under vk then B can immediately output a forgery. Notice that B is able to decrypt each message to search for forgeries because it has all the secret keys. In the Challenge phase, B runs the $\mathsf{Encrypt}$

algorithm, choosing a symmetric key itself and using vk as the signature key. It uses the oracle in the game it is playing to compute the signature, σ, for the ciphertext. If, in Phase 2, the adversary produces a decryption query that contains a signature $\sigma' \neq \sigma$ that verifies with vk, B presents σ' as a forgery.

B will win the strong unforgeability game with at least the probability that A presents a ciphertext that has a forged signature. Therefore, by our assumption about the signature scheme, A must present this with probability less than ϵ_3. □

Lemma 4. *For all t-time adversaries A, the probability A wins the n-recipient privacy game is at most ϵ_2, given that A does not output a forged signature, the simulation does not output a ciphertext component that violates strong correctness, and that A outputs recipient sets S_0 and S_1 with $|S_0 \cap S_1| = n - 1$.*

Proof. Given an adversary A that wins the n-recipient privacy game with probability greater than ϵ_2 without forging signatures, we construct a machine B that breaks the (t, q, ϵ_2)-CCA-key-privacy of the public key scheme as follows.

Init: The algorithm B receives I, pk_0, and pk_1 from the key privacy challenger, and gives I to A.

Setup: B runs Keygen for each recipient in $S_0 \cap S_1$ and gives the public keys to A, keeping the secret keys for itself. Additionally, B gives the public keys pk_0 and pk_1 to A.

Phase 1: Given a decryption oracle query (u, C), B runs $\mathsf{Decrypt}$ directly for $u \in S_0 \cap S_1$ using the secret keys it has. Otherwise, if the adversary requests decryption from the user corresponding to pk_0 or pk_1, B uses the decryption oracle to simulate decryption.

Challenge: Given message M from A, B runs $\mathsf{Encrypt}$ until the call to Enc for either pk_0 or pk_1. B simulates this encryption by asking the key privacy challenger for a challenge ciphertext c^* on the appropriate message. B then continues running $\mathsf{Encrypt}$, producing an entire challenge ciphertext C^*. Notice B does not know the value of b it is simulating. It is the key privacy challenger who selects b.

Phase 2: Given an oracle query (u, C), with $C \neq C^*$, B must simulate the decryption algorithm in order to respond. If $u \in S_0 \cap S_1$, then B possesses sk_u and can run $\mathsf{Decrypt}$ directly using its knowledge of the secret keys.

Otherwise B must decrypt for the user corresponding to pk_0 or pk_1. To do this B first parses the ciphertext as $\sigma||C_1||C_2$ and $C_1 = c_1||\cdots||c_n$. It then simulates the $\mathsf{Decrypt}$ routine. For each $c_i \neq c^*$ it encounters, B makes a decryption request to the key privacy challenger. If $c_i = c^*$ for some i, the simulation ignores c^*. To see why this is a correct simulation, we consider two cases.

1. If the key privacy challenger selected $b = 1$ and $u \in S_0$, then the key privacy challenger encrypted c^* for pk_1 and B is attempting to decrypt c^* with pk_0. In this case, the actual attempted decryption would output $\perp$ because c^* is encrypted for another user and this experiment is conditioned on the challenger not outputting a challenge ciphertext that violates strong correctness. (Symmetrically, if $b = 0$ and $u \in S_1$, then c^* was encrypted for pk_0 and B is attempting to decrypt with pk_1.) Thus, $\mathsf{Decrypt}$ should ignore c^*.

2. If the key privacy challenger selected $b = 0$ and $u \in S_0$ (symmetrically, $b = 1$ and $u \in S_1$), then the decryption of c^* will contain the same vk used in the challenge ciphertext C^*. By the condition that A does not forge signatures, the σ contained in C does not verify with vk because $C \neq C^*$. Thus $\mathsf{Decrypt}$ should ignore c^*.

In either case, B's simulation of $\mathsf{Decrypt}$ correctly ignores c^*. Thus, B can simulate decryption.

Guess: B forwards the adversary's guess b' to the key privacy challenger.

In this experiment B will perfectly simulate the game for A, where the coin flip in the recipient privacy game will be the same as the coin flip, b, in the IK-CCA game that B plays. Therefore, B will win the IK-CCA game if and only if A wins its game. □

Lemma 5. *No adversary who outputs recipient sets S_0 and S_1 with $|S_0 \cap S_1| = n-1$ can break the $(t, q, n, \epsilon_1 + \epsilon_2 + \epsilon_3)$-CCA-recipient-privacy of the construction in Section 4.1.*

Proof. The probability that A wins the recipient privacy game (with $|S_0 \cap S_1| = n - 1$) at most the probability A wins the recipient privacy game when it outputs no forgeries and correctness is not violated plus the probability A outputs a forgery plus the probability that strong correctness is violated. By our assumptions, Claim 3, and Lemma 4, we have that A does not break the $(t, q, n, \epsilon_1 + \epsilon_2 + \epsilon_3)$-CCA-recipient-privacy of the construction in Section 4.1. □

Theorem 6. *The construction in Section 4.1 is $(t, q, n, n(\epsilon_1 + \epsilon_2 + \epsilon_3))$-CCA-recipient-private.*

Proof. Given an adversary A, we show that A cannot break $(t, q, n, n(\epsilon_1 + \epsilon_2 + \epsilon_3))$-CCA-recipient-privacy by a hybrid reduction argument. Suppose A outputs recipient sets S_0 and S_1. For each $i = 0, \ldots, m = n - |S_0 \cap S_1|$, define L_i to be S_0 with the first i elements (that are not in S_1) replaced with the first i elements of S_1 (that are not in S_0). Thus $L_0 = S_0$, $L_m = S_1$, and $L_i \cap L_{i-1} = n - 1$.

Suppose A successfully breaks recipient privacy with advantage greater than $n(\epsilon_1 + \epsilon_2 + \epsilon_3)$. Then, there exists an i, with $1 \leq i \leq m \leq n$, such that A distinguishes the game L_{i-1} from L_i with probability greater than $(\epsilon_1 + \epsilon_2 + \epsilon_3)$. However, we could then use A to break a recipient privacy game with two sets S_0' and S_1' that differ by only one element. We define S_0' to consist of the elements of $S_0 \cap S_1$, the first $i - 1$ elements of S_0 that are not in S_1, and the last $m - (i - 1)$ elements from S_1 that are not in S_0. Similarly, we define S_1' to consist of the elements of $S_0 \cap S_1$, the first i elements in S_0 that are not in S_1, and the last $m - i$ elements from S_1 that are not in S_0. We can then use A to break this game with advantage greater than $(\epsilon_1 + \epsilon_2 + \epsilon_3)$ by simulating the game in which it distinguishes L_{i-1} from L_i (i.e. by withholding the extra secret keys from A).

However, by Lemma 5, no adversary can distinguish S_0' from S_1' with such advantage. Therefore, A cannot distinguish S_0 from S_1 with advantage greater than $n(\epsilon_1 + \epsilon_2 + \epsilon_3)$. □

B Proof of second construction

Assume $(\mathsf{Init}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is ϵ_1-strongly-correct, (t, q, ϵ_2)-CCA-semantically-secure, and (t, q, ϵ_3)-CCA-key-private. Assume $(\mathsf{Sig\text{-}Gen}, \mathsf{Sig}, \mathsf{Ver})$ is $(t, 1, \epsilon_4)$-strongly-existentially-unforgeable. Assume CDH in G is (t, ϵ_5)-hard and algorithm D decides DDH in G. We follow the hybrid reduction strategy from the previous section, but the case in which the recipient sets differ by only one recipient is more involved. Observe that the proof of Claim 3 applies to the construction in Section 4.2.

Lemma 7. *For all t-time adversaries A, the probability A wins the n-recipient privacy game is at most $2\epsilon_2 + \epsilon_3 + 2\epsilon_5$, given that A does not output a forged signature, the simulation does not output a ciphertext component that violates strong correctness, and A outputs recipient sets S_0 and S_1 with $|S_0 \cap S_1| = n - 1$.*

Proof. Given a t-time adversary A that wins the n-recipient privacy game with probability greater than $2\epsilon_2 + \epsilon_3 + 2\epsilon_5$ without forging signatures, we proceed by a sequence of hybrid experiments. Let v be the unique element of $S_0 - S_1$, let w be the unique element of $S_1 - S_0$, and let c be the ciphertext component that corresponds to either v or w. The hybrids are as follows:

L_0: The challenge ciphertext C^* is correctly encrypted for recipient set S_0.

L_1: c is replaced with $H(g^{ra_v})||\mathsf{Enc}_{\mathsf{pk}_v}(R)$, where R is a random string of the same length as $\mathsf{vk}||g^{ra_v}||K$.

L_2: c is replaced with $R'||\mathsf{Enc}_{\mathsf{pk}_v}(R)$, where R' is a random string of length λ.

L_3: c is replaced with $R'||\mathsf{Enc}_{\mathsf{pk}_w}(R)$. Notice the component is now encrypted for w instead of v.

L_4: c is replaced with $H(g^{ra_w})||\mathsf{Enc}_{\mathsf{pk}_w}(R)$.

L_5: c is replaced with $H(g^{ra_w})||\mathsf{Enc}_{\mathsf{pk}_w}(\mathsf{vk}||g^{ra_w}||K)$. Notice the challenge ciphertext is now correctly encrypted for recipient set S_1.

A t-time adversary can distinguish L_0 from L_1 with advantage at most ϵ_2 as follows. Given a t-time adversary A_1, we construct a machine B_1 that simulates either L_0 or L_1. We use B_2 to break CCA-semantic-security.

Init: B_1 receives I and pk from the CCA semantic security challenger, and gives I to A_1.

Setup: B_1 runs Keygen, but replaces the public key for user v with pk.

Phase 1: B_1 simulates $\mathsf{Decrypt}$ using the challenger's decryption oracle for pk.

Challenge: B_1 uses the challenger to encrypt a challenge c^* containing either $\mathsf{vk}||g^{ra_v}||K$ or R. B_1 uses this ciphertext in its simulation of $\mathsf{Encrypt}$.

Phase 2: B_1 simulates $\mathsf{Decrypt}$ using the challenger's decryption oracle for pk. If B_1 would need to query the challenger on c^*, it instead assumes the decryption is $\mathsf{vk}||g^{ra_v}||K$.

Guess: B_1 forwards the adversary's guess b' to the challenger.

If the challenger encrypts $\mathsf{vk}||g^{ra_v}||K$, then B_2 is simulating L_1. Otherwise, the challenger encrypts R, and B_2 is simulating L_2. Therefore, by our assumption about CCA-semantic-security, A_2 can distinguish L_1 from L_2 with advantage at most ϵ_2.

A t-time adversary can distinguish L_1 from L_2 with advantage at most ϵ_5 as follows. Given a t-time adversary A_2, we construct a machine B_2 that simulates the recipient privacy game with A_2, except it uses a CDH challenger to generate g^a and g^b. If A_2 or B_2 ever makes an oracle query x for which D, the DDH algorithm, accepts (g, g^a, g^b, x), B_2 suspends the simulation and reports the computed CDH value x. B_2 picks a random $b \in \{0, 1\}$. If $b = 0$, then B runs an exact simulation of L_1. Otherwise, it runs the following simulation of L_2.

Init: B_2 exactly simulates Init.

Setup: B_2 exactly simulates Setup, except that instead of choosing g^{a_v} at random, it uses the value g^a supplied by the CDH challenger.

Phase 1: Given a decryption query (u, C), if $u \neq v$, B_2 exactly simulates decryption (using information from Setup). Otherwise, B_2 processes each $c_j = l||c$ in turn, as follows:

1. Calculate $p \leftarrow \mathsf{Dec}(\mathsf{sk}_v, c)$.
2. If p is $\perp$, continue to the next j.
3. Otherwise, parse p as $\mathsf{vk}||x||K$.
4. If D accepts (g, g^r, g^a, x) and $l = H(x)$, then
   (a) if the signature verifies with vk, then return the decryption of message using K,
   (b) else output $\perp$.

Challenge: Instead of choosing g^r at random, B_2 uses g^b as g^r. B_2 is able to simulate $\mathsf{Encrypt}$ on users $u \neq v$ because it knows a_u from the Setup step. To prepare the ciphertext component for v, B_2 proceeds as follows:

1. Pick a random R of the same length as $\mathsf{vk}||g||K$.
2. $c^* \leftarrow \mathsf{Enc}_{\mathsf{pk}_v}(R)$.
3. Pick a random $R' \in \{0,1\}^\lambda$.
4. $c \leftarrow R'||c^*$.

Phase 2: Given a decryption query (u, C), B_2 simulates decryption in the same manner as it simulates Phase 1, except if $u = v$ and some $c_j = l||c^*$. If that c_j is processed, then return the decryption of the message using K if $l = R'$. Otherwise, continue to the next j.

Guess: B_2 records A_2's guess.

A_2 can distinguish the simulation of L_0 from L_1 only if it queries the random oracle on g^{ab} or queries the decryption oracle on a ciphertext containing g^{ab}. Either query, however, causes B_2 to suspend the simulation and win the CDH game (notice the query to H in step 4 of the decryption simulation). Thus, A_2 can distinguish L_0 from L_1 with advantage at most ϵ_5.

A t-time adversary can distinguish L_2 from L_3 with advantage at most ϵ_3 as follows. Given a t-time adversary A_3, we construct a machine B_3 that simulates either L_1 or L_2. Machine B_3 functions in a manner analogous to machine B from Lemma 4, with the following exceptions.

1. B_3 uses the construction from Section 4.2 (modified to use R and R' as appropriate).
2. In the Phase 1 and 2 steps, B_3 examines only correctly labeled ciphertexts.

Notice B_3 can prepare a ciphertext component for either v or w because the label and the plaintext are independent of the recipient. If the key privacy challenger encrypts for v, then B_3 is simulating L_2. Otherwise, the challenger encrypts for w and B_3 is simulating L_3. Therefore, by our assumption about CCA-key-privacy, A_3 can distinguish L_2 from L_3 with advantage at most ϵ_3.

The case for distinguishing L_3 from L_4 is symmetric with the case for distinguishing L_1 from L_2. The case for distinguishing L_4 from L_5 is symmetric with the case for distinguishing case L_0

from L_1. If A can distinguish $S_0 = L_0$ from $S_1 = L_5$ with probability greater than $2\epsilon_2 + \epsilon_3 + 2\epsilon_5$, then A can distinguish L_0 from L_1 with probability greater than ϵ_2, or L_1 from L_2 with probability greater than ϵ_5, or L_2 from L_3 with probability greater than ϵ_3, or L_3 from L_4 with probability greater than ϵ_5, or L_4 from L_5 with probability greater than ϵ_2. However, none of these cases can hold. Therefore, A cannot distinguish S_0 from S_1 with advantage greater than $2\epsilon_2 + \epsilon_3 + 2\epsilon_5$. □

Theorem 8. *The scheme in Section 4.2 is* $(t, q, n, n(\epsilon_1+2\epsilon_2+\epsilon_3+\epsilon_4+2\epsilon_5))$*-CCA-recipient-private.*

Proof. This theorem follows from the same argument as Theorem 6. □