Correcting Privacy Violations in Blind-Carbon-Copy (BCC) Encrypted Email

Adam Barth
abarth@cs.stanford.edu

Dan Boneh*
dabo@cs.stanford.edu

Abstract

We show that many widely deployed email encryption systems reveal the identities of Blind-Carbon-Copy (BCC) recipients. For example, encrypted email sent using Microsoft Outlook *completely exposes the identity of every BCC recipient*. Additionally, several implementations of PGP expose the full name and email address of BCC recipients. In this paper, we present a number of methods for providing BCC privacy while preserving the existing semantics of email. Our constructions use standard public key systems such as RSA and ElGamal and suggest that BCC privacy can be implemented efficiently without changing the underlying broadcast semantics of the email system.

1 Introduction

Email messages should not reveal the identities of Blind-Carbon-Copy (BCC) recipients. We show that many widely deployed email encryption systems, however, reveal the identities of every BCC recipient to *all* email recipients and to anyone who examines the email message en route. In most cases, the BCC recipient's identity is exposed by a unique identifier that also exists in publicly accessible databases on the Internet. In some cases, however, the full name and email address of a BCC recipient is included *in the clear* in the ciphertext of the encrypted email message. Figure 1 summarizes BCC privacy behavior in various email clients.

To understand why encrypted email messages expose BCC recipient identities, we first review how email encryption is implemented. Typically, an email message is sent from a Mail User Agent (such as Outlook) to a Mail Transfer Agent (such as Sendmail). The Mail Transfer Agent then delivers the message to all recipients. Email encryption systems implemented by the Mail User Agent, such as S/MIME or PGP, usually create a single encrypted blob for each message and send the blob to every recipient. Loosely speaking, these systems encrypt an email message by first picking a random Message Encryption Key (MEK) and then encrypting MEK under the public key of each recipient. Hence, a message with n recipients contains n encryptions of the MEK. These n ciphertexts are placed in the encrypted blob header.

Now consider an encrypted email message with BCC recipients. The encrypted blob header contains an encryption of MEK under the public key of each BCC recipient. Clearly, this reveals the *number* of BCC recipients. Much worse, in many email encryption systems, the encrypted blob header actually reveals the *identity* of each BCC recipient.

*Supported by NSF.

	Completely Exposes	Partially Reveals	Protects Identity
S/MIME-based			
Apple Mail.app 2.622	✓		
Outlook 2003	✓		
Outlook Express 6	✓		
Thunderbird 1.02	✓		
Outlook Web Access			✓
PGP-based			
EudoraGPG 2.0	✓		
GPGshell 3.42	✓		
Hushmail		✓	
KMail 1.8			✓
PGP Desktop 9.0			✓
Turnpike 6.04			✓

Figure 1: BCC recipient behavior by Mail User Agent. When sending encrypted email, Apple Mail, Microsoft Outlook, Outlook Express, Thunderbird, EudoraGPG, and GPGshell all completely expose the identities of every BCC recipient to all recipients. Hushmail partially reveals the identities of BCC recipients. Outlook Web Access, KMail, PGP Desktop, and Turnpike protect the identity of BCC recipients by encrypting separate messages for each BCC recipient.

S/MIME. For each encrypted MEK, S/MIME records the identity of the intended recipient, enabling recipients to efficiently locate their MEKs inside the header. In doing so, many S/MIME implementations, including Microsoft Outlook, Apple Mail.app, and Mozilla Thunderbird, enable any message recipient to fully identify all BCC recipients.

PGP. Some PGP implementations, including EudoraGPG and GPGshell, also expose BCC recipients by sending identifying information intended to improve decryption efficiency. Other PGP implementations, including Hushmail, use an option in the PGP message format to withhold identifying information. However, as we discuss in Section 2, it is nevertheless possible to identify BCC recipients from a short list of potential candidates.

These BCC privacy violations can be fixed using three general techniques. We briefly discuss each in turn:

Separate encryptions. The Mail User Agent (MUA) can conceal BCC recipients by sending separate encryptions to each message recipient. This approach works well and is currently used by a few encrypted email clients, including Outlook Web Access, KMail, and Turnpike. However, sending separate messages greatly complicates the integration of secure email programs with some MUAs, such as Eudora, and it also incurs some performance penalty on the mail server. We discuss this method in more detail in Section 3.6.

Private broadcast encryption. Recipient privacy can be assured by fixing S/MIME so that the encrypted email blob does not reveal the identity of BCC recipients. To do so we introduce a new mechanism called *private* broadcast encryption, which is a broadcast encryption system in which a ciphertext reveals no information about the intended set of recipients. We give precise definitions in Section 3 and present three constructions of private broadcast encryption

```
444:d=9  hl=2 l=   3 prim: OBJECT            :commonName
449:d=9  hl=2 l=  11 prim: PRINTABLESTRING   :Henry Kyser
462:d=7  hl=2 l=  32 cons: SET
464:d=8  hl=2 l=  30 cons: SEQUENCE
466:d=9  hl=2 l=   9 prim: OBJECT            :emailAddress
477:d=9  hl=2 l=  17 prim: IA5STRING         :h9565@hotmail.com
```

Figure 2: Part of an ASN1 dump of an S/MIME segment from an encrypted email sent with Outlook Express 6. The BCC recipient's identity is readily apparent because he used a self-signed certificate. If his certificate had instead been issued by a third party, he could have been identified by his certificate's unique serial number.

---

schemes. Our construction using ElGamal encryption is the most efficient. Any one of these constructions can be used to fix BCC privacy violations in S/MIME while preserving the current behavior where the same encrypted blob is sent to all recipients. We view private broadcast encryption as the main technical contribution of this work.

MTA intervention. The Mail Transfer Agent (MTA) can intervene by stripping the appropriate entries from the encrypted blob header prior to forwarding the email to the intended recipients. However, this approach requires the MTA to understand various encrypted email formats, such as S/MIME and PGP, rendering this approach unattractive.

2 BCC privacy in existing email systems

We begin by surveying how existing encrypted email systems handle BCC recipients. We first examine S/MIME-based systems and then discuss PGP-based systems. As we will see, an S/MIME-encrypted email message often completely exposes the identities of all BCC recipients. Any recipient can recover the full name and email address of *every BCC recipient* either directly from the message or after consulting the Internet. Several PGP-based email encryption systems also violate BCC privacy.

2.1 S/MIME-based email encryption systems

Secure/Multipurpose Internet Mail Extensions (S/MIME) is an email encryption standard used by numerous Mail User Agents including Apple Mail.app, Microsoft Outlook, Microsoft Outlook Express, and Mozilla Thunderbird. In order to send an S/MIME-encrypted email message, the sender must obtain a certificate for each of the message's recipients. These certificates bind email addresses to public keys and are often signed by a trusted third party, such as VeriSign. Some email users, however, use self-signed certificates.

S/MIME messages contain a field named `RecipientInfo` for each recipient, which helps the recipient locate the encryption of the MEK under her public key. The `RecipientInfo` field contains a field named `IssuerAndSerialNumber`, which according to RFC 2315 [10], "specifies the recipient's certificate (and thereby the recipient's distinguished name and public key)." The S/MIME standard [8] does not address the privacy implications of this field and does not discuss BCC privacy. However, the `IssuerAndSerialNumber` field has significant privacy implications.

Outlook, Thunderbird, and Mail.app expose BCC recipients. Figure 2 contains part of an ASN1 dump of an encrypted email sent with Microsoft Outlook Express 6. The identity of the BCC recipient appears in plain text because, as he used a self-signed certificate, he himself was the issuer of his certificate. Even if he had instead used a certificate issued by a third party, his identity would nonetheless have been revealed because the message would have contained his certificate's unique serial number. For example, if he had used a VeriSign certificate, his complete certificate could be obtained from its serial number using VeriSign's web site [17]. The following Mail User Agents completely expose the identities of every BCC recipient in this way:

- Apple Mail.app 2.622,
- Microsoft Outlook 2003,
- Microsoft Outlook Express 6, and
- Mozilla Thunderbird 1.02.

A general attack on S/MIME BCC privacy (RSA). We stress that BCC privacy violations are inherent in how RSA is used in S/MIME. That is, even if the `RecipientInfo` field did not explicitly identify BCC recipients, BCC privacy would still be violated. Recall that an encrypted email contains an encryption of the message-encryption key (MEK) under each BCC recipient's public key. This ciphertext is contained in a field named `EncryptedKey`. We show that, when using RSA, this `EncryptedKey` field leaks information about the recipient's identity.

Suppose Alice receives an S/MIME-encrypted email message with a BCC recipient she knows to be either Bob or Carol, with RSA public keys (e_1, N_1) and (e_2, N_2) respectively. She can determine, with non-negligible probability, whether Bob or Carol is the BCC recipient as follows:

1. Parse the email message and locate the `EncryptedKey` field c.
2. If $N_2 \leq c < N_1$, then Bob is the BCC recipient.
3. If $N_1 \leq c < N_2$, then Carol is the BCC recipient.

When the ciphertext falls between N_1 and N_2, one of the candidate BCC recipients cannot have been the actual BCC recipient. Because RSA ciphertexts for a public key (e, N) are uniformly distributed in the group $\mathbb{Z}_N$, this occurs with non-negligible probability. For example, if Bob and Carol both use 1024-bit public keys, the expected difference between N_1 and N_2 is on the order of 2^{1022} and the probability a ciphertext encrypted using the larger public key lies between N_1 and N_2 is approximately 1/4. If Bob and Carol use public keys of different security levels, the attack succeeds with overwhelming probability.

Outlook Web Access sends separate messages. Outlook Web Access is a web-based Mail User Agent built into Microsoft Exchange Server. Interestingly, Microsoft Outlook Web Access 2003 does not expose the identities of BCC recipients because, "by default, [it] submits a single encrypted message for all visible recipients (those on the TO and CC lines) and a separate encrypted message for each invisible recipient (those on the BCC line)" [5]. This respects BCC privacy because recipients do not receive encryptions of MEK for other BCC recipients.

Outlook Web Access also supports two other message encryption options. The message encryption options are configured using the `SecurityFlags` key in the Windows Registry under

`HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\MSExchangeWeb\OWA\`

```
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
gpg: encrypted with ELG-E key, ID 028EAE1C
gpg: encrypted with ELG-E key, ID 3CF61C7B
gpg: decryption failed: secret key not available
```

Figure 3: Transcript of an attempted GPG decryption of a message having two BCC recipients. The BCC recipients' identities are completely exposed by their key IDs. If these were real PGP keys, each BCC recipient's full public key, including his full name and email address, could be recovered using a PGP Key Server. The message was created with the EudoraGPG 2.0 (GnuPG 1.2.2) plug-in for Eudora 6.2.

Enabling bit `0x040` in `SecurityFlags` causes Outlook Web Access to create two separate encryptions for each message, one for the visible recipients and one for the invisible (BCC) recipients. Using this setting, the visible recipients do not learn about BCC recipients, but each BCC recipient learns the identities of every other BCC recipient. Enabling bit `0x080` causes a single encryption to be sent to each recipient, revealing the identities of BCC recipients to every recipient. The Message Security Guide [5] notes that each of these settings "improves performance in comparison with the default setting." This approach to protecting privacy is discussed further in Section 3.6.

2.2 PGP-based email encryption systems

OpenPGP claims to be the most widely used email encryption standard in the world [15]. The OpenPGP standard has several implementations including PGP Desktop, the GNU Privacy Guard (GPG), EudoraGPG, GPGshell, Hushmail, Turnpike, and KMail. Unlike S/MIME, PGP does not rely on certificates, but instead uses a Web of Trust [11] to bind identities to public keys. PGP also supports several cryptographic algorithms, including RSA and ElGamal.

EudoraGPG and GPGshell expose BCC recipients. EudoraGPG is a plug-in for Qualcomm's Eudora Mail User Agent. GPGshell is a graphical interface for GPG enabling PGP encryption in Outlook Express. Both EudoraGPG and GPGshell completely expose the identities of BCC recipients. Figure 3 contains a transcript of an attempted GPG decryption of a message sent with EudoraGPG. The message reveals the key IDs of two BCC recipients. PGP uses key IDs to aid decryption, similar to the way in which S/MIME uses `RecipientInfo`. These key IDs, however, reveal the identities of BCC recipients because PGP users typically publish their public keys and key IDs in the Web of Trust and on public key servers, such as MIT's PGP Key Server [12]. A curious recipient need only use a key server's search facility to learn the full name and email address of every BCC recipient.

Hushmail leaks BCC identities. Hushmail is a web-based PGP email service. Hushmail does not include key IDs in encrypted email messages, but instead replaces them with zeros (ostensibly to foil traffic analysis [6]). Unfortunately, this increases the amount of work required of legitimate

recipients to decrypt messages. Such a message sent 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 without key IDs, Hushmail email messages leak information about the identities of BCC recipients. If a recipient uses an RSA public key, her RSA ciphertext will leak information about her identity as described in the previous section. If a recipient uses an ElGamal public key, GPG's implementation of ElGamal causes encrypted email messages to leak information about the identities of BCC recipients.

Attack on BCC privacy (ElGamal). Suppose Alice receives a GPG-encrypted email message with a BCC recipient she knows to be either Bob or Carol, with ElGamal public keys (p, g, g^a) and (q, h, h^b) respectively. She can determine, with non-negligible probability, whether Bob or Carol is the BCC recipient as follows:

1. Parse the message to obtain the encrypted MEK, (x, y).
2. If $q \leq x < p$ or $q \leq y < p$, then Bob is the BCC recipient.
3. If $p \leq x < q$ or $p \leq y < q$, then Carol is the BCC recipient.

This attack succeeds with non-negligible probability because GPG chooses a random group during key generation. Bob and Carol are likely, therefore, to have ElGamal keys in different groups. When the ciphertext contains group elements outside either Bob or Carol's group, then one of the candidate recipients is eliminated and privacy is breached.

ElGamal encryption can be used to protect the privacy of BCC recipients, as long as all recipients have keys in the same group and the Decision Diffie-Hellman (DDH) problem is hard for that group (see Section 3.2). However, not only does GPG generate keys in different groups, GPG also implements ElGamal in the full multiplicative group $\mathbb{Z}_p^*$, where the Legendre symbol renders DDH easy.

Turnpike, KMail, and PGP Desktop send separate messages. Demon's Turnpike is a Mail User Agent with integrated support for PGP. KMail is a Mail User Agent included with the K Desktop Environment for Linux. PGP Desktop is the official PGP distribution from PGP, Inc. Turnpike, KMail, and PGP Desktop protect the identities of BCC recipients by separately encrypting messages for each BCC recipient. Turnpike and KMail are able to send separate messages because each fully integrates PGP. PGP Desktop, however, is neither a Mail User Agent nor a plug-in. Instead, it runs in the background, intercepting outgoing SMTP sessions and generating multiple encryptions of email messages, one for each BCC recipient.

3 Private broadcast encryption

In this section we present cryptographic constructions that provide BCC privacy. We do so using a new mechanism we call *private broadcast encryption*, in which a single encrypted email blob is created that hides the intended set of recipients. The resulting encrypted email can then be safely broadcast by the Mail Transfer Agent to all email recipients. Every recipient can decrypt the email blob, but they learn nothing about the identities of other recipients. Thus, private broadcast encryption an be used to fix BCC privacy violations in S/MIME without resorting to sending separate emails.

We define private broadcast encryption in Section 3.1 and present constructions for private broadcast encryption schemes. All of our constructions use key-private cryptosystems [1], which we review in Section 3.2. Our first recipient-private scheme (Section 3.3) is a straightforward modification of PGP that omits key IDs and uses a key-private cryptosystem. Our section construction (Section 3.4) reduces recipient work to $O(\log n)$, where n is the number of message recipients, but requires a larger message length of $O(n \log n)$ to provide privacy. These first two schemes can be instantiated with any key-private cryptosystem. In our third scheme (Section 3.5), we consider a construction specific to key-private ElGamal. We achieve a message length of $O(n)$ and recipient work of only two modular exponentiations.

3.1 Security model and recipient privacy

We model encrypted email as a private broadcast encryption system. In a standard broadcast encryption (e.g. [13]), the recipient set is included *in the clear* as part of the ciphertext. In contrast, for private broadcast encryption systems, our goal is to conceal the recipient set from the recipients themselves (as is appropriate for BCC).

Broadcast encryption systems typically have a centralized "dealer" who generates secret keys for each recipient. To more closely model email encryption, we allow principals to generate their own keys using some global parameters. For example when using key-private ElGamal, these global parameters specify a generator g and a group G in which each principal is to choose her key. This eliminates the key-privacy attacks described in Section 2.

Security model. 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 keypairs.
- $(\mathsf{Hdr}, \mathsf{MEK}) \leftarrow \mathsf{Encrypt}(S)$. Given a set of public keys $S = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_n\}$ generated by $\mathsf{Keygen}(I)$, generates a header Hdr and a message encryption key MEK. Hdr contains information allowing members of S to recover MEK using their secret keys. Thus, the encryption of a message M consists of $C = (\mathsf{Hdr}, E_{\mathsf{MEK}}[M])$ where $E_{\mathsf{MEK}}[M]$ is the encryption of M under the symmetric key MEK.
- $\mathsf{MEK} \leftarrow \mathsf{Decrypt}(\mathsf{Hdr}, \mathsf{sk})$. Given a header Hdr and a secret key sk, returns MEK if the corresponding public key $\mathsf{pk} \in S$, where S is the set used to generate Hdr. A recipient can then use MEK to recover M from $E_{\mathsf{MEK}}[M]$.

Semantic security of a private broadcast encryption system is defined as in other broadcast encryption schemes (see for example [4]).

Recipient privacy. We define a notion of recipient privacy for private broadcast encryption systems using a game between a challenger and an adversary. In the game, the adversary provides two recipient sets, S_0 and S_1, of equal size. The challenger furnishes the adversary with all the public keys as well as the secret keys common to S_0 and S_1. The challenger then provides the adversary with an encryption using either S_0 or S_1. The adversary wins the game if he or she successfully guesses which of S_0 or S_1 the challenger used. The details of the game are as follows:

1. The challenger runs $I \leftarrow \mathsf{Setup}(\lambda)$ and gives the adversary A the global parameter I.
2. A outputs $S_0, S_1 \subseteq \{1, \ldots, n\}$ such that $|S_0| = |S_1|$.
3. The challenger generates keys for each potential recipient, $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{Keygen}(I)$, and sends to A each pk_i for $i \in S_1 \cup S_2$ as well as each sk_i for $i \in S_0 \cap S_1$.
4. The challenger picks a random $b \in \{0, 1\}$, runs $(\mathsf{Hdr}, \mathsf{MEK}) \leftarrow \mathsf{Encrypt}(\{\mathsf{pk}_i \mid i \in S_b\})$, and gives A the header Hdr and the message encryption key MEK.
5. A outputs its guess $b' \in \{0, 1\}$.

Adversary A wins the game if $b = b'$.

Definition. A private broadcast encryption system is (t, ϵ, n)-recipient-private if, for all t-time adversaries A, the probability A wins the above game using recipient sets of size $m < n$ is at most $1/2 + \epsilon$.

We note that a standard hybrid argument [2] shows that our definition also implies unlinkability among sets of ciphertexts. We also note that recipient privacy allows Hdr to leak the number of recipients, just as semantic security allows a ciphertext to leak the length of the plaintext. To hide the number of BCC recipients one could always pad the recipient set to a given size using dummy recipients.

3.2 Public key privacy

We recall the notion of a key-private cryptosystem defined in [1]. In a key-private cryptosystem, a ciphertext does not leak the public key used in its creation. 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. This notion is essential for email recipient privacy because encrypted email messages must not leak information about the public keys of their BCC recipients.

Key-private ElGamal. A variant of ElGamal encryption is key-private, assuming DDH is hard [1]. Recall that in standard ElGamal, a principal generates her public key by picking a generator, g, of a group G and an exponent, a, at random. Her public key is the pair (g, g^a) and her secret key is the exponent a. In key-private ElGamal, the group G and the generator g are fixed public parameters of the cryptosystem. To generate a public key, a principal picks an exponent, a, at random. Her public key is g^a and her secret key is the exponent a. Once G and g are fixed, key-privacy follows from the assumption that DDH is hard in G.

Key-private RSA. A key-private variant of RSA can be produced by modifying only the RSA encryption function [1]. Essentially, the standard randomized encryption process is repeated until a ciphertext is produced that is below some global upper bound. This defeats the attack on RSA key privacy from Section 2 for principals with key of the same security level and is sufficient to provide key privacy. Importantly, key privacy can be assured by *encryptors alone* without a need to reissue public keys.

Waters et al. [18] propose the stronger notion of an "incomparable public key" cryptosystem, in which the identity of the recipient is also concealed from the sender. Such a cryptosystem can be used, when desired, to extend our constructions to provide recipient anonymity.

3.3 Construction 1: Inefficient recipient privacy

A straightforward implementation of private broadcast encryption simply performs separate key-private encryptions for each recipient and concatenates the results. This construction is inefficient for recipients because each recipient must perform, on average, $n/2$ decryption operations, where n is the number of message recipients.

More precisely, the construction is as follows. The Setup and Keygen procedures are the same as for the key-private system. The encrypt and decrypt algorithms work as follows:

Encrypt. Given a set of recipient public keys $S = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_n\}$, compute $\mathsf{Encrypt}(S)$ as follows:

1. Pick a random message encryption key MEK.
2. For each $\mathsf{pk}_i \in S$, encrypt MEK under pk_i. Denote such ciphertexts $\{\mathsf{MEK}\}_{\mathsf{pk}_i}$.
3. Return $\mathsf{Hdr} = \{\mathsf{MEK}\}_{\mathsf{pk}_{\pi(1)}} || \cdots || \{\mathsf{MEK}\}_{\mathsf{pk}_{\pi(n)}}$, where π is a random permutation of $1, \ldots, n$.

Decrypt. To obtain MEK, a recipient attempts to decrypt each ciphertext in turn. Given a header Hdr and a secret key sk, compute $\mathsf{Decrypt}(\mathsf{Hdr}, \mathsf{sk})$ as follows:

1. Parse $\mathsf{Hdr} = c_1 || \cdots || c_n$.
2. For each i, attempt to decrypt c_i using sk.
3. Return MEK, the plaintext recovered from the first successful decryption. (We assume c_i contains an integrity check that fails when c_i is decrypted with the wrong private key.)

Privacy. Upon receiving such a message, the adversary learns only the number of message recipients. The encryptions of MEK do not reveal recipient identities to the adversary because they are encrypted in a key-private cryptosystem. An adversary would have to break the key privacy of the underlying cryptosystem to be able to guess a recipient. The complete proof is immediate and is omitted.

The space required by this construction is proportional to n because MEK is encrypted separately for each recipient. A linear message length seems unavoidable because, information theoretically, the message must describe S. Other schemes for encrypting a single message for multiple recipients [13, 14, 4] claim ciphertexts of sub-linear size, but those schemes transmit their recipient sets in addition to their ciphertexts.

Performance. This construction is inefficient for recipients. Each recipient receives n encryptions of MEK, but has no idea which encryption she is able to decrypt with her private key. To decrypt the message, a recipient must try decrypting each one until she discovers the encryption of MEK under her public key. Therefore, a recipient expects to perform $O(n)$ decryptions in order to decrypt the message.

3.4 Construction 2: Recipient privacy using hash tables

The previous scheme is inefficient because a recipient has no idea which of the encryptions of MEK is intended for her. To reduce recipient workload, we partition encryptions of MEK into buckets, enabling each recipient to limit her search to a single bucket. A recipient identifies which bucket to search by hashing her public key using a sender-specified $O(\log n)$-universal hash function [7].

Encrypt. Given a set of recipient public keys $S = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_n\}$, compute $\mathsf{Encrypt}(S)$ by building a hash table T with n buckets, each of size $O(\log n)$, as follows:

1. Pick hash function H uniformly at random from a $\lambda \log n$-universal family of hash functions, where λ is a security parameter (e.g. $\lambda = 16$).
2. For each $\mathsf{pk}_i \in S$, add $\{\mathsf{MEK}\}_{\mathsf{pk}_i}$ to bucket $H(\mathsf{pk}_i)$ of T.
3. If any bucket of T contains more than $\lambda \log n$ ciphertexts, empty T and return to step 1.
4. Add random ciphertexts to each bucket until each bucket contains exactly $\lambda \log n$ ciphertexts.
5. Randomly permute the contents of each bucket.
6. Return $\mathsf{Hdr} = H||T$.

Decrypt. To obtain MEK, a recipient need only attempt to decrypt ciphertexts in a single bucket. Given a header Hdr and a secret key sk (with associated public key pk), compute $\mathsf{Decrypt}(\mathsf{Hdr}, \mathsf{sk})$ as follows:

1. Parse $\mathsf{Hdr} = H||T$.
2. For each c in bucket $H(\mathsf{pk})$ of T, attempt to decrypt c using sk.
3. Return MEK, the plaintext recovered from the first successful decryption.

Privacy. The space required by this construction is $O(n \log n)$ because in Step 5 we pad each bucket of the hash table to the same length. (The hash function itself can be represented in $O(\log n)$ space, see for example [16].) Without this padding, recipient privacy would be breached. To see this, consider an adversary with two candidate recipient sets, S_0 and S_1, and a message encrypted for one of those sets. The adversary could build two candidate hash tables, T_0 and T_1, by applying H to the public keys for S_0 and S_1, respectively. Without padding, she could match T uniquely to either T_0 or T_1 with high probability and, in doing so, determine the actual recipient set.

The padded hash table protects recipient privacy because it looks the same to the adversary regardless of the recipient set. Even knowledge of some of the recipients' private keys does not help the adversary guess the identities of the other recipients. The hash table helps the adversary only if either T_0 or T_1 contains a bucket with more than $\lambda \log n$ recipients. However, this occurs with exponentially low probability in λ. As before, the full proof of security is immediate and is omitted.

Performance. The hash table saves each recipient a significant amount of work. To decrypt M, a recipient with public key pk need only attempt to decrypt ciphertexts in bucket $H(\mathsf{pk})$. Because each bucket contains $O(\log n)$ ciphertexts, each recipient performs at most $O(\log n)$ decryptions. This compares favorably to the $O(n)$ decryption required in the previous section.

3.5 Construction 3: Efficient recipient privacy using ElGamal

Using key-private ElGamal, we can greatly improve on the above generic constructions. Recall that in key-private ElGamal, all the public keys are in the same prime order group G, share the same generator $g \in G$, and are of the form g^a, where a is the corresponding secret key. We leverage these properties to define a *private hash function* on public keys.

Encrypt. Given a set of recipient public keys $S = \{g^{a_1}, \ldots, g^{a_n}\}$, compute $\mathsf{Encrypt}(S)$ by generating a list L associating private hash values with encryptions of MEK as follows:

1. Generate a fresh public-secret keypair g^b and b using $\mathsf{Keygen}(I)$. This keypair defines a function $H(x) = x^b$.
2. For each $\mathsf{pk}_i = g^{a_i} \in S$, let $h_i = H(\mathsf{pk}_i)$ be the private hash of pk_i.
3. Let $L = h_{\pi(1)}||\{\mathsf{MEK}\}_{\mathsf{pk}_{\pi(1)}}|| \cdots ||h_{\pi(n)}||\{\mathsf{MEK}\}_{\mathsf{pk}_{\pi(n)}}$, where π sorts L by $h_{\pi(i)}$.
4. Return $\mathsf{Hdr} = g^b||L$.

Decrypt. To obtain MEK, a recipient need only perform two modular exponentiations. Given a header Hdr and a secret key $\mathsf{sk} = a$, compute $\mathsf{Decrypt}(\mathsf{Hdr}, \mathsf{sk})$ as follows:

1. Parse $\mathsf{Hdr} = x||L$, where $L = h_1||c_1|| \cdots ||h_n||c_n$.
2. Let $h = x^a$ be the private hash of the recipients public key.
3. Perform a binary search on L for $h_i = h$.
4. Return MEK, the plaintext obtained by decrypting c_i using a.

Private hash functions. Consider the function $H(x) = x^b$. Given a presentation of H as b, a principal can compute H at any point of his or her choice. In particular, she can compute $H(g^a) = g^{ab}$ without knowledge of a. Conversely, given a presentation of H as g^b, a principal can compute $H(g^a)$ with knowledge of a. Under the DDH assumption, a principal given only g^b and g^a cannot compute $H(g^a)$. Furthermore, she cannot even distinguish $H(g^a)$ from a random group element. It is in this sense that we call H a private hash function.

Private hash functions are useful for constructing recipient-private encrypted email messages because the sender (who knows b) can label a recipient's encryption of MEK with the private hash of the recipient's public ElGamal key, g^a. The sender includes g^b in the email (the same g^b for each recipient), enabling each recipient to evaluate H only on her own public key. The labels on other recipients' ciphertexts appear meaningless. Essentially, this presentation of H allows the sender to transmit seemingly independent random values to an unlimited number of recipients in constant space. The labeled encryptions translate these random values into a shared secret, MEK.

Efficiency. To decrypt the message, a recipient uses her private key a and g^b to compute $H(\mathsf{pk}) = g^{ab}$ and then performs a binary search on L for $H(\mathsf{pk})$, which is associated with her encryption of MEK. She decrypts MEK and then decrypts the message. The recipient need only perform two modular exponentiations, one to compute $H(\mathsf{pk})$ and one to decrypt MEK, and examine, not attempt to decrypt, $\log n$ entries in L. Additionally, the space required is proportional to the number of recipients, n. To save additional space, the sender need not include all the bits of $H(\mathsf{pk})$. He or she need only include enough bits to uniquely identify each entry in L (approximately $2 \log n$ bits). Thus, transmission size is linear in n and decryption time is constant.

Privacy. Recipient privacy is achieved because only two principals can evaluate $H(\mathsf{pk})$: the sender and the holder of pk's corresponding secret key. To the adversary, the values of this private hash function look random and independent of both the candidate recipient public keys and the transmitted presentation of H. Consequently, the hash values are of no use to the adversary. The encryptions of MEK themselves are also of no use to the adversary in guessing the BCC recipients because of key privacy.

Theorem 1 (Privacy). *If DDH in the group G is (t, ϵ)-hard, then the private broadcast system detailed in this section is $(t, (n+1)\epsilon, n)$-recipient-private.*

Proof outline. The full proof is presented in the appendix. A formal definition of our construction is presented in Section A.1. In Section A.2, we prove the security of the private hash function used in our construction by reduction to DDH. Section A.3 contains the main reduction from our private broadcast system to DDH. We use the random self-reducibility of DDH to generate ciphertexts for recipients whose secret keys the adversary does not possess. □

3.6 Comparing two approaches to BCC privacy

So far we have discussed two approaches to ensuring BCC privacy: (1) encrypt separate messages for each recipient, and (2) change the underlying encryption mechanism to use one of the systems of this section so that the recipient set is hidden. We now compare these two approaches.

Advantages of private broadcast encryption

- Many encrypted email systems are unable to send separate messages to each recipient because they are implemented as Mail User Agent plug-ins. For example, the PGP plug-in for Eudora is unable to send multiple encryptions of a single email message. As another example, OpenSSL is used to S/MIME-encrypt email messages, but OpenSSL itself is incapable of sending multiple messages. In contrast, if OpenSSL produced a single message using private broadcast encryption, that message could be sent to every recipient without compromising recipient privacy.

- Sending a separate message to each recipient increases MTA workload and bandwidth requirements. Currently, if an email message is sent to multiple recipients whose email addresses share a common domain, one copy of the email message is sent to their common MTA, who then delivers copies to their individual mail spools. If a Mail User Agent instead sent separate encryptions of the message to each recipient, the sender's MTA would need to transfer multiple copies of the message, one for each recipient, to the recipient's MTA. Sending a single message, even one with a longer header, reduces workload and bandwidth because the encryption header is typically much shorter than the actual message.

- The privacy implications of key IDs in PGP were known to the PGP community as early as 1997, at which time it was recommended that PGP implementations send separate encryptions to each BCC recipient [9]. Eight years later, several PGP implementations still expose the identities of BCC recipients. Had PGP adopted a scheme for private broadcast encryption, it seems likely that today no PGP implementation would expose the identities of BCC recipients. Instead, sending separate messages relies on implementations to break the usual broadcast semantics of email.

Advantages of sending separate messages.

- By sending separate messages, Mail User Agents can protect the privacy of BCC recipients without changes to the encryption standard. Thus, this scheme can be deployed immediately and unilaterally, which is appealing for the short-term mitigation of privacy violations. By contrast, implementing private broadcast encryption requires modifying the S/MIME and/or PGP standards.
- The number of, and even the existence of, BCC recipients is protected when separate encryptions are sent to each recipient. This faithfully reproduces the semantics of unencrypted email, where the number of BCC recipients is protected. Both S/MIME and PGP reveal the number of recipients, as do our schemes for private broadcast encryption. The number of recipients can be hidden by padding the recipient list with dummy recipients, but this decrease efficiently by increasing the length of the message.
- The format of a ciphertext often reveals information about the cryptosystem in which it was generated, including the value of the security parameter. Private broadcast encryption requires recipients to agree on a common cryptosystem and security level in order to achieve recipient privacy, whereas sending separate messages achieves recipient privacy while allowing email recipients to choose the cryptosystem and security level of their public keys.

4 Conclusions

We showed that many encrypted email systems mishandle BCC recipients and violate privacy. The most severe violations are in implementations of S/MIME, including Outlook, Mail.app, and Thunderbird, where the identities of BCC recipients are completely exposed to anyone with a text editor. Implementations of PGP are generally less egregious, but some implementations, such as EudoraGPG and GPGshell, completely expose BCC recipients. Other implementations, such as Hushmail, are vulnerable to more subtle attacks and also leak information about BCC identities. These privacy violations are inherent in how these systems use the underlying RSA and ElGamal public key schemes.

A standard solution to this problem is to send seperate emails to BCC recipients [9], although many mail programs fail to implement this. We proposed a different solution based on a new mechanism called private broadcast encryption which has several advantages over sending seperate emails. For systems using RSA, our best private broadcast encryption construction uses a hash table, but still requires the decryptor to perform a few RSA decryption trials. For systems using ElGamal, our construction uses a private hash function and is much more efficient. Nevertheless, we believe that our hash table method is the best suited for practical use because email recipients can continue to use their existing RSA public keys.

We leave as an open problem the construction of a private broadcast encryption system with shorter ciphertext.

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*, 2000.

[3] D. Boneh. The decision diffie-hellman problem. In *Proceedings of ANTS III*, 1998.

[4] D. Boneh, C. Gentry, and B. Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. To appear in Crypto '05. IACR eprint Report 2005/018.

[5] C. Budd. Exchange server 2003 message security guide, 2004. http://www.microsoft.com/technet/prodtechnol/exchange/2003/library/exmessec.mspx.

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

[7] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. McGraw-Hill Higher Education, 1990.

[8] S. Dusse, P. Hoffman, B. Ramsdell, L. Lundblade, and L. Repka. RFC 2311: S/MIME version 2 message specification, 1998. http://www.ietf.org/rfc/rfc2311.txt.

[9] W. H. Geiger III. Re: KeyIDs and key fingerprints, 1997. http://www.imc.org/ietf-openpgp/mail-archive/msg00351.html.

[10] B. Kaliski. RFC 2315: PKCS #7: Cryptographic message syntax, 1998. http://www.ietf.org/rfc/rfc2315.txt.

[11] A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997.

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

[13] 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.

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

[15] OpenPGP. Website, 2005. http://www.openpgp.org/.

[16] 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.

[17] VeriSign. Search for digital IDs, 2005. https://digitalid.verisign.com/services/client/.

[18] B. R. Waters, E. W. Felten, and A. Sahai. Receiver anonymity via incomparable public keys. In *CCS '03: Proceedings of the 10th ACM Conference on Computer and Communications Security*, pages 112–121. ACM Press, 2003.

A Formal construction and proof of privacy

A.1 Formal construction

We present a formal definition of our private broadcast encryption system from Section 3.5.

- $\mathsf{Setup}(\lambda)$. Choose a group G of order p, where p is a λ-bit prime, and choose a random generator $g \in G$. Return (G, g).
- $\mathsf{Keygen}(G, g)$. Choose a random $a \in \mathbb{Z}_p$ and return (g^a, a).
- $\mathsf{Encrypt}(S)$. Choose a random $b \in \mathbb{Z}_p$ and a random message encryption key MEK. Construct Hdr as follows:
  1. $h_{\mathsf{pk}} \leftarrow \mathsf{pk}^b$ for each $\mathsf{pk} \in S$.
  2. $c_{\mathsf{pk}} \leftarrow (g^d, \mathsf{MEK}\mathsf{pk}^d)$ for each $\mathsf{pk} \in S$.
  3. $\mathsf{Hdr} \leftarrow (g^b, \{(h_{\mathsf{pk}}, c_{\mathsf{pk}}) \mid \mathsf{pk} \in S\})$, where the set is sorted by h_{pk}.

  Return $(\mathsf{Hdr}, \mathsf{MEK})$.
- $\mathsf{Decrypt}(\mathsf{Hdr}, \mathsf{sk})$. Obtain MEK from Hdr using sk as follows:
  1. Parse $\mathsf{Hdr} = (h, C)$.
  2. $h^* \leftarrow h^{\mathsf{sk}}$.
  3. Find c^* such that $(h^*, c^*) \in C$.
  4. Parse $c^* = (x, y)$.
  5. $\mathsf{MEK} \leftarrow y/x^a$.

  Return MEK.

A.2 Private hash lemma

At the core of our construction in the notion of a private hash function. We define the Decision Private Hash (DPH) problem as follows. Given $(g, x_0, x_1, y, z) \in G^5$ such that exactly one of (g, x_0, y, z) and (g, x_1, y, z) is a DDH tuple, output b if (g, x_b, y, z) is a DDH tuple.

Lemma 2 (Private hash). *If DDH is (t, ϵ)-hard in G, then DPH is (t, ϵ)-hard in G.*

Proof. We prove the contrapositive. Given A with advantage ϵ for DPH, we construct B^A with advantage ϵ for DDH. B^A proceeds as follows:

1. Given a DDH challenge (g, x, y, z), choose $\hat{x} \in G$ at random.
2. Choose $b \in \{0, 1\}$ at random. $x_b \leftarrow x$. $x_{1-b} \leftarrow \hat{x}$.
3. Run A on (g, x_0, x_1, y, z) and obtain result b'.
4. Guess (g, x, y, z) is a DDH tuple if, and only if, $b = b'$.

If the challenge is a not a DDH tuple, A has no information about b, so $b = b'$ with probability $1/2$. If the challenge is a DDH tuple, then B's DPH challenge is properly distributed. Therefore, B has advantage ϵ. □

A.3 Proof of privacy

We are now prepared to formally prove Theorem 1, namely that the construction of Section 3.5 is recipient private. We begin by defining an algorithm for reducing privacy in our system to DDH.

Definition. Given a recipient privacy adversary A, define DDH adversary B^A on input (g, x, y, z) in a group G of order p as follows:

1. Send A the global parameters $I = (G, g)$.
2. Receive from A recipient sets S_1 and S_2.
3. Proceed as follows:
   (a) Generate keys $(g^{a_i}, a_i) \leftarrow \mathsf{Keygen}(I)$ and set $\mathsf{pk}_i \leftarrow g^{a_i}$ for each $i \in S_1 \cap S_2$.
   (b) For each $j \in (S_1 \cup S_2) - (S_1 \cap S_2)$, construct DDH tuples (g, x_j, y_j, z_j) by DDH random self-reducibility [3]:
   $$x_j \leftarrow x g^{\alpha_j}, \qquad y_j \leftarrow y^{\gamma_j} g^{\beta_j}, \text{ and} \qquad z_j \leftarrow z^{\gamma_j} x^{\beta_j} y^{\alpha_j \gamma_j} g^{\alpha_j \beta_j},$$
   where α_j, β_j, and γ_j are chosen at random from $\mathbb{Z}_p$.
   (c) Set $\mathsf{pk}_j \leftarrow x_j$ for each $j \in (S_1 \cup S_2) - (S_1 \cap S_2)$.
   (d) Send A every pk_i for $i \in S_1 \cup S_2$ and every a_i for $i \in S_1 \cap S_2$.
4. Choose a random $b \in \{0, 1\}$. Choose MEK at random, and proceed as follows:
   (a) Set $h_i \leftarrow y^{a_i}$, pick $b_i \in \mathbb{Z}_p$ at random, and set $c_i \leftarrow (g^{b_i}, \mathsf{MEK} g^{a_i b_i})$ for each $i \in (S_1 \cap S_2)$.
   (b) Set $h_j \leftarrow z y^{\alpha_j}$ and set $c_j \leftarrow (y_j, \mathsf{MEK} z_j)$ for all $j \in S_b - (S_1 \cap S_2)$.
   (c) Set $\mathsf{Hdr} \leftarrow y || \{(h_i, c_i) \mid i \in S_b\}$ (sorted by h_i).
   (d) Send A the message encryption key MEK and the header Hdr.
5. Receive from A a guess b'. Guess (g, x, y, z) is a DDH tuple if, and only if, $b = b'$.

The following lemma states that algorithm B, when given a DDH tuple as input, correctly implements our private broadcast encryption system and acts as a recipient privacy challenger.

Lemma 3 (Correctness). *Given any adversary A, B^A on input (g, g^a, g^b, g^{ab}) interacts with A as a recipient privacy challenger for our construction.*

Proof. B^A challenge to A is properly distributed because the (g, x_j, y_j, z_j) and (g, x_j, y, h_j) are properly distributed DDH tuples. □

The follow lemma states an adversary A interacting with B, when given a randomly distributed tuple, cannot guess b reliably (if DDH is hard). This lemma reduces the problem of guessing b to n instances of DPH and applies the private hash lemma.

Lemma 4 (Containment). *If DDH is (t, ϵ)-hard in G, then no t-time A can guess b with advantage greater than $n\epsilon$ when interacting with B^A on input (g, x, y, z), where x, y, z are independent random elements of G.*

Proof. The variables in the program of B^A with indices $i \in S_1 \cap S_2$ give A no information about b because the manipulation of those variables is independent of b.

The variables in the program of B^A with indices $j \in S_b - (S_1 \cap S_2)$ could conceivably lead to information leakage. However, the only results of computing with j-indexed variables given to A are x_j, y_j, z_j, and h_j. The x_j, y_j, and z_j are independent random elements of G and so give A no information about b. However, h_j is a more subtle issue.

For each h_j, B could indicate to A two candidates, x_j and $x_k \in S_{1-b} - (S_1 \cap S_2)$, forming $|S_1 - S_2|$ instances of a DPH decision problem (g, x_j, x_k, y, h_j). By a hybrid argument, A cannot guess b with advantage greater than $|S_1 - S_2|\epsilon \leq n\epsilon$. □

We are now ready to prove the main privacy theorem. We apply our lemmas about B to reduce the recipient privacy of our system to DDH.

Theorem 5 (Privacy). *If DDH in the group G is (t, ϵ)-hard, then the private broadcast system in Section 3.5 is $(t, (n+1)\epsilon, n)$-recipient-private.*

Proof. We prove the contrapositive. Given an adversary A who breaks the $(t, n, (n+1)\epsilon)$-recipient-privacy of our system, B^A breaks the assumption that DDH is (t, ϵ)-hard.

We consider two cases. First, assume the probability that B^A guesses "DDH tuple" when given a non-DDH tuple is greater than $1/2 + n\epsilon$. By Lemma 4, this breaks the assumption that DDH is (t, ϵ)-hard.

Second, assume the probability that B^A guesses "DDH tuple" when given a non-DDH tuple is at most $1/2 + n\epsilon$. By Lemma 3, the probability B^A guesses "DDH tuple" when given a DDH tuple is at least $1/2 + (n+1)\epsilon$. Therefore, B^A has advantage $(n+1)\epsilon - n\epsilon = \epsilon$ in breaking the assumption that DDH is (t, ϵ)-hard. □