600.641 Special Topics in Theoretical Cryptography April 2, 2007

Lecture 17: Re-encryption

Instructor: Susan Hohenberger *Scribe: Zachary Scott*

Today's lecture was given by Matt Green.

1 Motivation

Proxy re-encryption is a relatively newly-devised cryptographic primitive. The goal of proxy re-encryption is to securely enable the reencryption of ciphertexts from one secret key to another, without relying on trusted parties.

Example
Alice receives emails from many clients encrypted under her public key. When she leaves for summer vacation, she wants to delegate her email to Bob, but does not want to share a secret key with him.

$$C_a \longrightarrow \boxed{\text{mail server}} \longrightarrow C_b$$

2 Approaches

2.1 Naïve Way

Alice simply stores her secret key on the mail server. When mail arrives for her, the mail server decrypts using her stored secret key and reencrypts using Bob's public key.

$$C_a \longrightarrow \boxed{\text{mail server}} \longrightarrow C_b$$
$$C_a = Encrypt_{PK_a}(M)$$
$$C_b = Encrypt_{PK_b}(Decrypt_{SK_a}(C_a))$$

The obvious problem with this strategy is that the mail server must be completely trusted - an unlikely real-world expectation. Up til 1997, this was unfortunately the best solution available. That year, Mambo and Okamoto suggested that there may be more efficient approaches involving partial decryptions, but offered no additional security benefits for Alice's secret key. [MO97]

2.2 Atomic Re-encryption: Blaze, Bleumer & Strauss, 1998

The BBS approach is based on the ElGamal cryptosystem and introduces the notion of a "re-encryption key" $RK_{A \to B}$. [BBS98] Using $RK_{A \to B}$ allows the proxy to re-encrypt from one secret key to another without ever learning the plaintext. Let's recall the BBS re-encryption scheme.

- **Key Generation**:
  $< g >= \mathbb{G}$ of prime order q.
  $SK_a = a \in \mathbb{Z}_q^*$, randomly selected. $SK_b = b \in \mathbb{Z}_q^*$, randomly selected.
  $PK_a = g^a$ $PK_b = g^b$
  $RK_{A \to B} = b/a = b \cdot a^{-1} \ (mod\ q)$

- **Encryption**:
  $m \in \mathbb{G}$, random $r \in \mathbb{Z}_q^*$
  $C_a = (g^r \cdot m,\ g^{ar})$

- **Decryption**:
  $m = \frac{g^r \cdot m}{(g^{ar})^{1/a}}$

- **Re-encryption**:

$$C_a \longrightarrow \boxed{\text{mail server}} \longrightarrow C_b$$
$$C_a = (g^r \cdot m,\ g^{ar})$$
$$C_b = (g^r \cdot m,\ (g^{ar})^{RK_{A \to B}})$$
$$= (g^r \cdot m,\ (g^{ar})^{b/a})$$
$$= (g^r \cdot m,\ g^{br})$$

Since the BBS proxy re-encryption algorithm is based on ElGamal, the ciphertext is semantically secure. Furthermore, without knowing a or b the proxy (mail server) cannot distinguish between $RK_{A \to B} = b/a$ and any random element of $\mathbb{Z}_q^*$.

This scheme is very elegant. However, there are several issues that one might want to improve on:

- Bidirectionality: The proxy can compute $(RK_{A \to B})^{-1} = a/b$, which would enable it to re-encrypt Bob's messages under Alice's key - Bob may not like this.

- Collusion: If the proxy colludes with Alice, it is trivial for them both to learn SK_B. Likewise, the proxy and Bob may collude to learn Alice's secret key.

- Re-encryption key generation: In order to compute $RK_{A \to B}$, one party must share his or her secret key with the other or they must rely on a trusted third party or compute some secure multiparty computation with the proxy.

Can we do better?

2.3 Unidirectional Proxy Re-encryption

We wish to improve on BBS in the following ways:

- No pre-sharing: Only require the use of SK_a and PK_b to create a re-encryption key from Alice to Bob. That is, Bob's secret key is not required.

- Unidirectionality: A re-encryption key from Alice to Bob should *not* allow re-encryptions from Bob to Alice.

- Collusion-resistance: Even if working together, Alice and the proxy should not be able to learn anything about Bob's secret key. (Note that no pre-sharing seems to imply this nice collusion resistance property.)

2.3.1 Related Attempt: Dodis & Ivan, 2003

In 2003, Dodis and Ivan proposed a general scheme for proxy *encryption* (not re-encryption) using only standard public key cryptosystems. [DI03] However, their system required Alice and Bob to pre-share a secret. Although it is clear that such a pre-sharing phase is possible to accomplish securely (e.g., via Diffie-Hellman), it is undesireable. It may be that Alice and Bob have no prior relationship whatsoever and bidirectional communication is impossible.

2.3.2 Ateniese, Fu, Green & Hohenberger, 2005

In 2005, Ateniese et al. constructed the first unidirectional, collusion resistant re-encryption without any required pre-sharing between parties, based on bilinear maps[AFGH06]. We will first review this algebraic setting and then describe their scheme.

Bilinear Maps

Bilinear maps gained great popularity in cryptography in 2001 when Boneh and Franklin used them to construct a special type of encryption scheme. Let's review the basics here.

Let $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3$ be cyclic groups of prime order q.

Function $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_3$ is a bilinear map iff for all $g_1 \in \mathbb{G}_1, g_2 \in \mathbb{G}_2, a, b \in \mathbb{Z}_q$, that $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$.

This proxy re-encryption algorithm uses bilinear maps of the form $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$, where $\mathbb{G}_1 =< g >$. e must be efficiently computable. Also, e must be non-degenerate; that is, $< e(g, g) >= \mathbb{G}_2$.

Note that with bilinear maps, the traditional Diffie-Hellman problem is not computationally difficult:
DDH: Given (g, g^a, g^b, g^c), is $g^{ab} = g^c$?
This is equivalent to asking if $e(g, g^c) = e(g^a, g^b) \Rightarrow Z^c = Z^{ab}$ (where $Z = e(g, g)$). If e is efficiently computable (as is required by the re-encryption algorithm), then so is the DDH problem.

Therefore, cryptosystems using bilinear maps occasionally rely on the Decisional *Bilinear* Diffie-Hellman (DBDH) assumption and its related Inversion (DBDHI):

DBDH: Given $(g, g^a, g^b, g^c$ (all $\in \mathbb{G}_1$), $T \in \mathbb{G}_2)$, is $T = e(g, g)^{abc}$?
k-DBDHI: Given $(g, g^y, g^{y^2} ... g^{y^k}$ (all in $\mathbb{G}_1$), $T \in \mathbb{G}_2)$, is $T = e(g, g)^{1/y}$?

See [Bet] for more information on Bilinear Maps and complexity assumptions.

The AFGH Algorithm

- **Key Generation**:
  $< g >= \mathbb{G}_1$ of prime order q.
  $SK_a = a \in \mathbb{Z}_q^*$, randomly selected. $SK_b = b \in \mathbb{Z}_q^*$, randomly selected.
  $PK_a = g^a$ $PK_b = g^b$
  random $r \in \mathbb{Z}_q^*$
  $Z = e(g, g)$
  $RK_{A \to B} = (g^b)^{1/a} = g^{b/a}$

- **Encryption**:
  $m \in \mathbb{G}_2$.
  $C_a = (Z^r \cdot m,\ g^{ra})$

- **Decryption (Alice)**:
  $m = \frac{Z^r \cdot m}{e(g^{ra}, g^{1/a})} = \frac{Z^r \cdot m}{Z^r}$

- **Re-encryption**:

$$C_a \longrightarrow \boxed{\text{mail server}} \longrightarrow C_b$$
$$C_a = (Z^r \cdot m,\ g^{ra})$$
$$C_b = (Z^r \cdot m,\ e(g^{ra}, RK_{A \to B}))$$
$$= (Z^r \cdot m,\ e(g^{ra}, g^{b/a}))$$
$$= (Z^r \cdot m,\ Z^{rb})$$

- **Decryption (Bob)**:
  $m = \frac{Z^r \cdot m}{(Z^{rb})^{1/b}}$

Analysis

This algorithm offers several nice features (but it is still not perfect!). Let's take a closer look at its properties.

- No pre-sharing of secret keys. That is, Alice is able to compute $RK_{A \to B}$ using only SK_a and PK_b; Bob does not need to share SK_b with Alice or any other party in order for the proxy re-encryption to function.

- It is non-interactive: only Bob's public key and Alice's secret key are used to create the re-encryption key, so no prior arrangement or secret sharing is necessary. Taken together with the lack of pre-sharing, this means that Alice may use this scheme without *any* prior knowledge on the part of Bob.

- Proxy security is maintained. There is no way for the proxy to read the messages it is re-encrypting or extract either party's secret keys (on its own).

- It is unidirectional under the Inverse Exponent Assumption (equivalent to the Diffie-Hellman Assumption): the proxy cannot calculate $g^{a/b}$ from $g^{b/a}$, and so Bob's ciphertexts cannot be re-encrypted for Alice.

- The algorithm is collusion-resistant; it is hard for the proxy to extract b from $RK_{A\to B} = g^{b/a}$, even with the help of Alice.
- The semantic security of ElGamal with this re-encryption function is retained under the Decisional Bilinear Diffie-Hellman Inversion Assumption (DBDHI).

Note that AFGH ciphertexts have two different forms: The 'original' form $(Z^r \cdot m, g^{ra})$ and the re-encrypted form $(Z^r \cdot m, Z^{rb})$. In the first form the second element in the pair is an element of $\mathbb{G}_1$; in the second form, the second element is an element of $\mathbb{G}_2$. Aside from aesthetic objections about modifying the ciphertext in this way, the effect is that **AFGH ciphertexts may only be re-encrypted once**. Once $RK_{A\to B}$ has been used to produce C_b, $RK_{B\to C}$ cannot be used to produce a ciphertext for Charlie.

This has benefits and drawbacks. Alice is afforded an additional measure of control; she knows that even if she authorizes re-encryption for Bob, Bob can't then delegate to someone else. However, it is conceivable that in some situations the ability to further delegate might be highly desirable.

3 Other Applications

Distributed Encrypted Storage

Suppose we want to build an encrypted filesystem. Not just *any* encrypted file system - we want a distributed system where clients request the keys to files from a key server. Standard key server schemes suffer a big risk: the owners of the encrypted content must trust the key server to act correctly and securely. Also, the operator of the key server has complete control access to all of the encryption keys and must be trustworthy.

Adapting the AFGH re-encryption scheme allows us to reduce the trust needed in the key server. Under this system, only the content owners have the capability to grant access to files, without sharing any secret keys, and the server operator has no access to the stored keys. Proxy re-encryption enables this in a not-terribly-inefficient way.

1. First, encrypt the filesystem using a fast symmetric key cryptosystem such as AES. A separate session key is used for each file.
2. Group Manager Alice then encrypts all of the file session keys under her proxy re-encryption public key PK_a.
3. When coworker Bob requests access to file F, Alice generates the re-encryption key $RK_{A\to B}$ and gives it to the key server. (For the next file Bob requests, this step does not need to be done.)
4. The key server (which may or may not be the same host as the file server) re-encrypts the file containing the AES key to F from Alice's public key to Bob's.
5. Bob may now use his secret key to decrypt the AES key file, which he uses to decrypt file F.

Alice may also act as a Group Manager. She would then be responsible for encrypting all of the stored AES keys under her SK_a, and computing re-encryption keys as necessary for subordinates/coworkers to access the files they need.

AFGH [AFGH06] implemented this filesystem on top of the Chefs networked file system using 128-bit AES. Running on a 2.8Ghz Pentium IV using an elliptic-curve crypto library, the system took 7.7ms for encryption, 21.7ms for re-encryption and 3.4ms for the final decryption. This performance was impractical when each filesystem block was encrypted separately, but was within range of practicality when encryption was performed on a more conventional per-file basis.

Additional Applications

Proxy re-encryption has also been used as the basis for program obfuscation. It is also being studied in relation to Identity-Based Encryption (IBE) and re-signature schemes.

References

[AFGH06] G. Ateniese, K. Fu, M. Green, and S. Hohenberger. Improved Proxy Re-encryption Schemes with Applications to Secure Distributed Storage. In *NDSS*, 2006.

[BBS98] M. Blaze, G. Bleumer, and M. Strauss. Divertable protocols and atomic proxy cryptography. In *EUROCRYPT*, 1998.

[Bet] John Bethancourt. Intro to Bilinear Maps. `http://www.cs.cmu.edu/~bethenco/bilinear_maps.pdf`. [Online; accessed 8-April-2007].

[DI03] Y. Dodis and A. Ivan. Proxy Cryptography Revisited. In *NDSS*, 2003.

[MO97] M. Mambo and E. Okamoto. Proxy Cryptosystems: Delegation of the Power to Decrypt Ciphertexts. In *TFECCS*, 1997.