A Generic Model For Proxy-Protected Anonymous Proxy Cryptography

Guoyan Zhang

Key Lab of Cryptographic Technology and Information Security
Ministry of Education, School of Mathematics, Shandong University, P.R.China

2008.05.14

Proxy Signature

- In 1996, Mambo et al. first invented the notion of proxy signature [1], in which the original signer can delegate his signature power to any reliable person called proxy signer when he is on vacation.
- Lee et al. [2] drew a conclusion in their paper that a secure proxy signature scheme must satisfy the following properties:Strong Unforgeability, Verifiability, Strong Identifiability, Strong Undeniability, Prevention of Misuse.
- Boldyreva, A. Palacio and B. Warinschi gave the formal notion and security model of proxy signature in [4], and the proxy signature schemes have provable-security guarantees;

Proxy Cryptographsystem

In 1997 Mambo et al. introduced the proxy cryptosystem (proxy decryption). In proxy cryptography, the original signer(decryptor) can delegate his cryptographic operation power to any reliable person called proxy signer(decryptor) when he is on vacation.

- Proxy cryptosystem with transformation;
- In 1998, Blaze, Beamer, and Strauss [3] proposed the notion of atomic proxy cryptography;
- Transformation-free proxy cryptosystems;

A Generic Model

Definition 1. A proxy-protected anonymous proxy decryption (PPAPD) scheme includes the following five algorithms:
-Setup: Given the security parameter 1^k, the delegatee runs the probability generation algorithm IBE_{Gen} to get the master secret key SK_O as his secret key and the master public parameter PK_O as his public key.
-Delegation Algorithm: The delegation algorithm includes two phases: user secret key generation and the partial proxy private key derivation.

- **Secret Key Generation:** Taken the public parameter PK_O as input, the delegater randomly picks a secret key SK_P and computes the corresponding public key PK_P. Following, he chooses an existential unforgeable signature scheme $S = (Gen_{sign}, Sign, Verify)$,

and computes the signature δ for the public key PK_P. Finally, he sends (δ, PK_P) with the verifying public key to the delegatee.

- **Partial Proxy Private Key Derivation:** Assuming the proxy time is t. Given the tuple (δ, PK_P), the public key PK_O, the secret key SK_O and the proxy time t, the delegatee first checks the validity of the public key PK_P and of the signature δ, if either invalid, he aborts, otherwise, he runs the private key extraction algorithm $IBE_{Extract}$ of IBE, and gets the partial proxy private key SK_{pp}. Then he sends SK_{pp} to the delegater.

-Encrypt: Taken the public keys PK_O and PK_P, the proxy time t and the message M, the probabilistic encryption algorithm Enc returns a ciphertext C on message M.

-Decrypt: On receiving the ciphertext C, the delegater gets the plaintext m using the secret key SK_P and the partial proxy private key SK_{pp}, or outputs the special symbol $\perp$ indicating invalid.

Definition 2. A proxy-protected anonymous proxy signature (PPAPS) scheme also includes five algorithms: The first three algorithms are same to the ones of the above proxy decryption scheme, and the last two algorithms are as follows:

-Sign: Taken the public key PK_O and PK_P, the proxy time t, the secret key SK_P, the partial proxy private key SK_{pp} and the message M, the delegater runs the probabilistic scheme to return the signature C on message M.

-Verify: Receiving the signature C, the verifier can verify the validity of the signature C. If $verify(C, PK_O, PK_P, M, t) = 1$, he outputs 1, otherwise outputs $invalid$.

Remark 1. From the above two models, we can see the delegatee cannot run the cryptographic operations imitating the delegater, because he doesn't know the secret key corresponding to the public key of the delegater. Of course, the delegatee can choose different public key whose corresponding secret key is knew to him instead of the delegater, but the delegater can publish his partial proxy private key to delate him.
Remark 2. In our model, any third party can not identify the identity of the delegater only from the public information, but the delegater can not deny he has run the cryptographic operation, because the signature for the public key is sent to the delegatee, and if the signature scheme used is existential unforgeable, the delegatee can revoke the identity of the delegater in dispute.

Remark 3. In fact, the public key is not combined with the identity and any third party cannot identify the accurate identity of the delegater. In other words, the model ensures the anonymity of the delegation. Furthermore, if the delegatee chooses a key pair (PK_P, SK_P), then he can run any cryptographic operations using the public key which also protects the privacy of the delegatee.

Attack Model

In our notion, we consider all potential actions of the adversary. There are two types of adversaries: the outside adversaries who aren't delegated or the malicious delegaters who want to imitate other delegaters, the malicious delegatee.

Definition 3. A proxy-protected anonymous proxy decryption $(PPAPD)$ scheme is secure against adaptive chosen ciphertext attack (IND-CCA) if no probabilistic polynomial time bound adversary has non-negligible advantage in either Game 1 or Game 2.

Game 1.
This game for the outside adversary. Taken a security parameter 1^k, the challenger runs the Setup algorithm to get the delegatee's secret key SK_O and the delegatee's public key PK_O, and he gives PK_O to the adversary, keeping SK_O secret.
Phase 1. The adversary can request two oracles: Partial-Proxy-Private-Key-Oracle and Decryption Oracle.
-Partial-Proxy-Private-Key-Oracle: On receiving the oracle$< PK_P, SK_P, t_i, \delta = Sign(PK_P) >$:

- The challenge checks the validity of the public key and the signature δ, if either invalid, he aborts. Otherwise, he searches the PartialProxyPrivateKeyList for a tuple $< PK_P, SK_{PP}, t_i >$, if exists, he sends SK_{PP} to the adversary.

- Otherwise, the challenge runs the Partial-Proxy-Private-Key-Derivation algorithm to get SK_{PP}, and adds the tuple $< PK_P, SK_{PP}, t_i >$ to the PartialProxyPrivateKeyList. He sends SK_{PP} to the adversary.

-Decryption-Oracles: On receiving the oracle $< PK_P, SK_P, C_i, t_i >$:

- The challenge checks the validity of the public key, if invalid, he aborts. Otherwise, he searches the PartialProxyPrivateKeyList for a tuple $< PK_P, SK_{PP}, t_i >$, if exists, he decrypts the ciphertext C_i using SK_{PP} and SK_P. And he sends the plaintext M to the adversary.

- Otherwise, the challenge runs the Partial-Proxy-Private-Key-Derivation algorithm to get SK_{PP}, and adds the tuple $< PK_P, SK_{PP}, t_i >$ to the PartialProxyPrivateKeyList. He decrypts the ciphertext C_i using SK_{PP} and SK_P to get M and sends the plaintext M to the adversary.

Challenge: The adversary generates a request challenge $< PK_{P^*}, SK_{P^*}, t_i^*, M_0, M_1 >$, t_i^* is the proxy time, and M_0, M_1 are equal length plaintext. If the public key PK_{P^*} is valid, the challenger picks a random bit $b \in \{0, 1\}$, sets $C^* = Enc(M_b, t_i^*, PK_{P^*}, PK_O)$. It sends C^* to the adversary.

Phase 2. The adversary can make polynomial queries, and the challenge responds as Phase 1.
At the end of the game, the adversary outputs $b' \in \{0, 1\}$ and wins the game if $b = b'$, furthermore, there are also two restrictions that the adversary has never request the partial proxy private key oracle on a tuple $< PK_{P^*}, SK_{P^*}, t_i^* >$ and the decryption oracle on a tuple $< C^*, t_i^*, PK_{P^*}, SK_{p^*} >$.

Game 2.
This game for the malicious delegatee adversary. Taken a security parameter 1^k, the challenger runs the Setup algorithm to get the delegatee's secret key SK_O and the delegatee's public key PK_O, and he gives (PK_O, SK_O) to the adversary.
Phase 1. The adversary can request one oracle: decryption oracle.
-Decryption-Oracles: On receiving the oracle $< PK_P, SK_P, C_i, t_i >$:

- The challenge checks the validity of the public key, if invalid, he aborts. Otherwise, he searches the PartialProxyPrivateKeyList for a tuple $< PK_P, SK_{PP}, t_i >$, if exists, he decrypts the ciphertext C_i using SK_{PP} and SK_P. And he sends the plaintext M to the adversary.

- Otherwise, the challenge runs the Partial-Proxy-Private-Key-Derivation algorithm to get SK_{PP}, and adds the tuple $< PK_P, SK_{PP}, t_i >$ to the PartialProxyPrivateKeyList. He decrypts the ciphertext C_i using SK_{PP} and SK_P to get M and sends the plaintext M to the adversary.

Challenge: The adversary generates a request challenge $< PK_{P^*}, SK_{P^*}, t_i^*, M_0, M_1 >$, t_i^* is the proxy time, and M_0, M_1 are equal length plaintext. If the public key PK_{P^*} is valid, the challenger picks a random bit $b \in \{0, 1\}$, sets $C^* = Enc(M_b, t_i^*, PK_{P^*}, PK_O)$. It sends C^* to the adversary.
Phase 2. The adversary can make polynomial queries, and the challenge responds as Phase 1.

At the end of the game, the adversary outputs $b' \in \{0, 1\}$ and wins the game if $b = b'$, furthermore, there are also one restriction that the adversary has never queried the decryption oracle on a tuple $< C^*, t_i^*, PK_{P^*} >$.

Definition 4. A proxy-protected anonymous proxy signature$(PPAPS)$ scheme is existential unforgeable against adaptive chosen message attack if no probabilistic polynomial time bound adversary has non-negligible advantage in either Game 1 or Game 2.

Game 1.

This game for the outside adversary. Taken a security parameter 1^k, the challenger runs the Setup algorithm to get the delegatee's secret key SK_O and the delegatee's public key PK_O, and he gives PK_O to the adversary, keeping SK_O secret.

The adversary can request two oracles: Partial-Proxy-Private-Key-Oracle and Signature-Oracle.
-Partial-Proxy-Private-Key-Oracle: On receiving the oracle$< PK_P, SK_P, t_i, \delta = Sign(PK_P) >$:

- The challenge checks the validity of the public key and the signature δ, if either invalid, he aborts. Otherwise, he searches the PartialProxyPrivateKeyList for a tuple $< PK_P, SK_{PP}, t_i >$, if exists, he sends SK_{PP} to the adversary.
- Otherwise, the challenge runs the Partial-Proxy-Private-Key-Derivation algorithm to get SK_{PP}, and adds the tuple $< PK_P, SK_{PP}, t_i >$ to the PartialProxyPrivateKeyList. He sends SK_{PP} to the adversary.

-Signature-Oracle: On receiving the oracle $< PK_P, SK_P, m_i, t_i >$:

- The challenge checks the validity of the public key, if invalid, he aborts. Otherwise, he searches the PartialProxyPrivateKeyList for a tuple $< PK_P, SK_{PP}, t_i >$, if exists, he signs m_i using SK_{PP} and SK_P in the normal way. And he sends the signature σ_i to the adversary.
- Otherwise, the challenge runs the Partial-Proxy-Private-Key-Derivation algorithm to get SK_{PP}, and adds the tuple $< PK_P, SK_{PP}, t_i >$ to the PartialProxyPrivateKeyList. He signs m_i using SK_{PP} and SK_P as the above step. And he sends the signature σ_i to the adversary.

At the end of the game, the adversary outputs $< PK_{p^*}, C^*, m^*, t^* >$, where PK_{p^*} is the challenge public key, C^* is the signature on message m^*, t^* is proxy time.
The adversary wins the game if he has never request the partial proxy private key oracle on a tuple $< PK_{P^*}, SK_{P^*}, t_i^* >$ and the signature oracle $< PK_p^*, SK_p^*, m^*, t^* >$.

Game 2.
This game for the malicious delegatee. Taken a security parameter 1^k, the challenger runs the Setup algorithm to get the delegatee's secret key SK_O and the delegatee's public key PK_O, and he gives $< PK_O, SK_O >$ to the adversary.
Phase 2. The adversary can request one oracle: signature oracle.
-Signature-Oracle: On receiving the oracle $< PK_P, SK_P, m_i, t_i >$:

- The challenge checks the validity of the public key, if invalid, he aborts. Otherwise, he searches the PartialProxyPrivateKeyList for a tuple $< PK_P, SK_{PP}, t_i >$, if exists, he signs the message m_i using SK_{PP} and SK_P. And he sends the signature σ_i to the adversary.

- Otherwise, the challenge runs the Partial-Proxy-Private-Key-Derivation algorithm to get SK_{PP}, and adds the tuple $< PK_P, SK_{PP}, t_i >$ to the PartialProxyPrivateKeyList. He signs the message m_i using SK_{PP} and SK_P. And he sends the signature σ_i to the adversary.

At the end of the game, the adversary outputs $(PK_{p^*}, C^*, m^*, t^*)$, where PK_{p^*} is the challenge public key, C^* is the signature on message m^*, t^* is proxy time. The adversary wins the game if he has never asked the signature oracle on $< PK_{p^*}, SK_{p^*}, m^*, t^* >$.

New Proxy-Protected Anonymous Proxy Decryption Scheme

Let (G, G_T) be bilinear map groups of order $p > 2^k$ and let $e : G \times G \to G_T$ denote a bilinear map. $H_1 : \{0,1\}^* \to \{0,1\}^n$ and $H_2 : \{0,1\}^* \to \{0,1\}^n$ are two collision-resistant hash functions.

Setup$(1^k, n)$**:** The original decryptor chooses g as a generator for G. Set $g_1 = g^\alpha$ for random $\alpha \in Z_p^*$, and pick a group element $g_2 \in G$ and vectors $(u', u_1, \ldots, u_n), (v', v_1, \ldots, v_n) \in G^{n+1}$. These vectors define the following hash functions:

$$F_u(W_1) = u' \prod_{i=1}^{n} (u_j^{i_j}), F_v(W_2) = v' \prod_{i=1}^{n} (v_j^{w_j}).$$

where $W_1 = i_1 i_2 \ldots i_n$ and $W_2 = w_1 w_2 \ldots w_n$. The original decryptor's public key is

$$PK_O = (G, G_T, e, g, g_1, g_2, u', u_1, u_2, \ldots, u_n, v', v_1, v_2, \ldots, v_n),$$

and the original decryptor's secret key is

$$SK_O = \alpha.$$

Delegation algorithm:

- **Secret key generation:** The proxy decryptor P randomly picks $x \in Z_p^*$, and computes the public key $PK_P = (X, Y) = (g_1^x, g_2^x)$. He runs Gen_{sign} of a secure signature scheme $S = (Gen_{sign}, Sign, Verify)$ to get the signature key pair (sk, vk), and he runs $Sign$ to get the signature δ on (X, Y). Finally he sends $((X, Y), \delta, vk)$ to the original decryptor O.

- **Partial proxy private key derivation:** Given $((X, Y), \delta, vk)$, the original decryptor O verifies: $e(X, g_2) = e(Y, g_1), Very(\delta, X, Y, vk) = 1$. If either fails, O outputs invalid. Otherwise, assuming the proxy time is t, he chooses random number $r \in Z_p^*$ and computes

$$W_1 = H_1(X, Y, t),$$

$$SK_{PP} = (d_1, d_2) = (g_2^{\alpha} \cdot F_u(W_1)^r, g^r).$$

  O sends $SK_{PP} = (d_1, d_2)$ to proxy decryptor P.

Encrypt: Assuming the proxy time is t. In order to encrypt message $m \in G_T$, parse $PK_P = (X, Y)$, and check the validity of the public key by the equation $e(X, g_2) = e(Y, g_1)$. If so, choose $s \in Z_p^*$ and compute the ciphertext as follows:

$$C = (C_0, C_1, C_2, C_3) = (m \cdot e(X, Y)^{-s}, g^s, F_u(W_1)^s, F_v(W_2)^s),$$

where $W_2 = H_2(C_0, C_1, C_2, PK_P)$.

Decrypt: The proxy decryptor P first computes $W_2 = H_2(C_0, C_1, C_2, PK_P)$ and checks the validity of the ciphertext by the following equation:

$$e(C_1, F_u(W_1)F_v(W_2)) = e(g, C_2C_3).$$

If the equation doesn't hold, he outputs invalid. Otherwise, he decrypts the ciphertext:

$$m = C_0(e(d_1, C_1)/e(C_2, d_2))^{x^2}.$$

Correctness:

$$C_0(e(d_1, C_1)/e(d_2, C_2))^{x^2} = C_0(e(g_2^{\alpha}F_u(W_1)^r, g^s)/e(g^r, F_u(W_1)^s))$$

$$= C_0(e(g_2, g_1)^s e(F_u(W_1)^r, g^s)/e(g^r, F_u(W_1)^s))^{x^2}$$

$$= me(X, Y)^{-s}e(g_1, g_2)^{x^2 s} = m.$$

Theorem 1. The proxy-protected anonymous proxy decryption scheme is IND-CCA secure against the outside adversary assuming the Decision Bilinear Diffi-Hellman problem is hard.

Theorem 2. The proxy-protected anonymous proxy decryption scheme is IND-CCA secure against the malicious deletee assuming the Decision Bilinear Diffi-Hellman problem is hard.

New Proxy-Protected Anonymous Proxy Signature Scheme

Setup: The original signer O chooses groups G, G_T of prime order q and a generator $P \in G$. There is also a bilinear map $e : G \times G \to G_T$. He chooses $H_1 : \{0,1\}^* \to G_T$ and $H_2 : \{0,1\}^* \to Z_q^*$. He randomly picks $s \in Z_q^*$ as his secret key, and computes $Q = sP$ as his public key. Then the public parameter and secret key are respectively :

$$PK_O = (G, G_T, q, P, H_1, H_2, e, Q), SK_O = s.$$

Delegation Algorithm:

- **Secret Key Generation:** The proxy signer picks $x \in Z_q^*$ and computes $PK_P = (X, Y) = (xP, xQ)$ as his public key. He runs Gen_{sign} of a secure signature scheme $S = (Gen_{sign}, Sign, Very)$ to get the signature key pair (sk, vk),

and he runs $Sign$ to get the signature δ on (X, Y). Finally he sends $((X, Y), \delta, vk)$ to the original signer O.

- **Partial Proxy Private Key Derivation:** On receiving (PK_P, δ, vk), the original signer verifies the validity of the public key and the signature for the public key by the two equations: $e(X, Q) = e(Y, P), Very(\delta, PK_P, vk) = 1$. If either equation doesn't hold, he outputs invalid; Otherwise, if the proxy time is t, he computes

$$W = H_1(X, Y, t), SK_{PP} = sW,$$

  and sends them to the proxy signer P.

Proxy Signature: The proxy signer P picks $k \in Z_q^*$ and computes

$$r = e(P, P)^k, c = H_2(m\|r), U = cxSK_{PP} + kP.$$

The proxy signature is (c, U, m).

Verification: Anyone can verify the validity of the proxy signature. The verifier computes

$$W = H_1(X, Y, t), r' = e(cW, Y)^{-1}e(U, P).$$

He checks whether $c = H_2(m\|r')$. If the condition holds, the signature is valid.

Security Analysis

Theorem 3. The proxy-protected anonymous proxy signature scheme is existential unforgeable against the outside adversary assuming the computational Diffi-Hellman(CDH) problem is hard in Gap Diffi-Hellman group.

Theorem 4. The proxy-protected anonymous proxy signature scheme is existential unforgeable against the malicious delegatee assuming the computational Diffi-Hellman(CDH) problem is hard in Gap Diffi-Hellman group.

Acknowledgement

The work is supported by 973 Project(No.2007CB807902) and the National Natural Science Foundation of China(NSFC Grant No.60525201)


M Mambo, K Usuda and E Okamoto. Proxy Signature: Delegation of the Power to Sign Messages. IEICE Trans.Fundations. 1996, E79-A(9):1338-1353.


B Lee, H Kim, and K Kim. Strong proxy signgture and its applications. In Proceedings of SCIS, 2001. pp. 603-608.


M Blaze, G Bleumer, M Strauss. Divertible protocol and atomic proxy cryptography. in: Advances in Cryptology -Eurocrypt '98, LNCS, vol. 1403, 1998. pp. 127-144.


A Boldyreva, A Palacio, B Warinschi. Secure Proxy Signature Schemes for Delegation of Signing Rights. At :http://eprint.iacr.org/2003/096.

Thank you!