Attribute-Based Signing Right Delegation

Weiwei Liu, Yi Mu, and Guomin Yang

School of Computer Science and Software Engineering,
University of Wollongong, Wollongong, NSW 2522, Australia
wl265@uowmail.edu.au, {ymu,gyang}@uow.edu.au

Abstract. Attribute-based signature and proxy signature are both very useful in many real-world applications. In this paper, we combine the special features of both signatures and propose an attribute-based proxy signature scheme, where the original signer, who possesses a set of attributes, can delegate his/her signing right to a designated proxy signer. By verifying the signature, a verifier can be convinced that the signature is generated by the proxy signer who has obtained the delegation from a legitimate signer whose attributes satisfy a predicate. However, the verifier cannot tell from the signature who is the original signer. We provide the formal definition and adversarial models for attribute-based proxy signature, and an efficient scheme that supports threshold predicates.

Keywords: proxy signature, attribute-based signature, threshold predicate.

1 Introduction

Proxy signature is a special type of digital signature and has been found useful in many real-world applications, for example, distributed computing [9] and grid computing [1]. Proxy signature allows an original signer to delegate his/her signing rights to a proxy signer who then can issue signatures on behalf of the original signer. The first proxy signature scheme was proposed by Mambo, Usuda and Okamoto in 1996 [8]. They discussed three different types of proxy signature, namely full delegation, partial delegation and delegation by warrant. Later, Kim et al. [2] proposed a new type of proxy signature combing partial delegation and warrant, and demonstrated that schemes combining partial delegation and warrant can provide a higher level of security than schemes based on partial delegation and warrant separately. Since then many proxy signature schemes based on partial delegation and warrant have been proposed (e.g.,[3,13,11,12,14,5]).

Attribute-based signature (ABS) is another special type of digital signature that has been proposed recently. It can be treated as an extension of identity-based signature (IBS) but has better fine-grained control over the signer's identification information. In an ABS, a signer with attribute set $\mathcal{A}$ will first obtain a secret key from the central authority (or key generation center), and then can use the obtained secret key to sign any messages. The signature can be verified with regards to an attribute predicate Υ and the verification will be successful if and

M.H. Au et al. (Eds.): NSS 2014, LNCS 8792, pp. 323–334, 2014.

only if the signer's attribute set $\mathcal{A}$ satisfies Υ. However, the verifier cannot gain any information about the signer's attributes except the fact that they satisfy the pre-claimed predicate. Several ABS schemes have been proposed recently to support different types of predicates. Li et al. [4] proposed two ABS constructions supporting flexible threshold predicates. In their schemes, the predicate is a set of n attributes, and the signer must possess at least k ($k \leq n$) of them in order to generate a valid signature. The verifier can be convinced that the signer is really holding k out of n attributes, but cannot find out which k attributes are possessed by the signer. Later, Maji et al. [7] proposed another ABS scheme where the attribute predicates can be expressed as monotone-span programs. Then in [10], Okamoto and Takashima proposed the first ABS scheme that can support more general non-monotone predicates. We noticed the paper regarding attribute-based signature has recently been proposed in [6]; However, the adversarial models in [6] are not properly defined. In addition, the application scenario is different from our work.

In this paper, we are interested in signing right delegation under the attribute-based setting environment. The proposed scheme can be regarded as a variant of attribute-based proxy signature schemes(ABPS). ABPS has many potential applications, for example, attribute-based authentication [7]. Consider a database whose access control is described in a policy such that only users who hold authorised attribute keys can access it. An authorised user can delegate his/her signing rights to another user so that the latter can also access the database and collect information when the former is not available. The delegated signer is called a proxy of the original authorised signer. In our proposed scheme, the verifier can be convinced that a valid proxy signer holds the right delegation from an original signer and therefore can access the database. The attributed based proxy signature can be regarded as a certificate for accessing the database.

The rest of the paper is organized as follows. We introduce some preliminaries in Section 2. The formal definition and security model of ABPS is presented in Section 3. We then present our ABPS scheme in Section 4 and prove its security in Section 5. The paper is concluded in Section 6.

2 Preliminaries

2.1 Bilinear Map

Let $\mathbb{G}_1$, $\mathbb{G}_2$ be two cyclic groups of prime order p and g a generator of $\mathbb{G}_1$. The $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ is said to be an admissible bilinear map if the following conditions hold:

- Bilinearity: $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$ for all $g_1, g_2 \in \mathbb{G}_1$ and $a, b \in_R \mathbb{Z}_p$.
- Non-degeneracy: There exists $g_1, g_2 \in \mathbb{G}_1$ such that $e(g_1, g_2) \neq 1_{\mathbb{G}_2}$.
- Computability: There is an efficient algorithm to compute $e(g_1, g_2)$ for all $g_1, g_2 \in \mathbb{G}_1$.

2.2 Complexity Assumption

Definition 1 ***Computational Diffie-Hellman (CDH) Problem****: Given* g, $g^a, g^b \in \mathbb{G}_1$ *for some random* $a, b \in \mathbb{Z}_p$*, compute* $g^{ab} \in \mathbb{G}_1$*. Define the success probability of a polynomial algorithm* $\mathcal{A}$ *in solving the CDH problem as:*

$$Succ^{CDH}_{\mathcal{A},\mathbb{G}_1}(\kappa) = \Pr[\mathcal{A}(g, g^a, g^b) = g^{ab} : a, b \in_R \mathbb{Z}_p]$$

where $\kappa = \mathsf{log}(p)$ *is the security parameter.*

Definition 2 ***Computational Diffie-Hellman (CDH) Assumption****:* $Succ^{CDH}_{\mathcal{A},\mathbb{G}_1}(\kappa)$ *is negligible in* κ*.*

2.3 Lagrange Interpolation

Given t points $q(1), q(2), ..., q(t)$ on a $t-1$ polynomial q, one could use Lagrange interpolation to compute $q(i)$ for any $i \in \mathbb{Z}_p$ through

$$q(i) = \sum_{j=1}^{t} q(j)\Delta_{j,s}(i).$$

The Lagrange coefficient $\Delta_{j,S}(i)$ of $q(j)$ in the computation of $q(i)$ can be computed as

$$\Delta_{j,S}(i) = \prod_{1 \le \pi \le t, \pi \ne j} \frac{i - \pi}{j - \pi}.$$

3 Attribute-Based Proxy Signature Definition and Security Model

3.1 Definition

An attribute-based proxy signature scheme is parameterized by a universe of possible attributes $\mathbb{A}$, a warrant space $\mathbb{M}_\omega$, and a message space $\mathbb{M}$. It consists of the following algorithms.

- **ABPS.Setup**: takes a security parameter 1^κ as input and outputs the public parameters $params$ and a master secret key MSK for the central authority.
- **ABPS.KeyGen**: takes $params$ as input and outputs a proxy key pair (pk, sk).
- **ABPS.AttrKeyGen**: takes $(MSK, params, \omega)$ as input where $\omega \subseteq \mathbb{A}$ is the attribute set of a user and outputs an attribute key sk_ω.
- **ABPS.DskGen**: takes $(sk_\omega, m_w \in \mathbb{M}_w, \Upsilon)$ as input, where m_w is a warrant specified by the original signer, Υ is a predicate such that there exists $\omega' \subseteq \omega$ which satisfies $\Upsilon(\omega') = 1$, and outputs a delegation key dsk.
- **ABPS.ProSig**: takes $(dsk, sk, m \in \mathbb{M})$ as input, and outputs a proxy signature σ.

– **ABPS.ProVer**: takes $(\Upsilon, pk, m_w, m, \sigma)$ as input, and outputs 1 ('accept') or 0 ('reject').

Correctness: We require that for any warrant and message spaces $\mathbb{M}_w, \mathbb{M} \subseteq \{0,1\}^*$ and any security parameter $\kappa \in \mathbb{N}$, if

$$(params, MSK) \leftarrow \textbf{ABPS.Setup}(1^\kappa),$$

$$(pk, sk) \leftarrow \textbf{ABPS.KeyGen}(params),$$

$$sk_\omega \leftarrow \textbf{ABPS.AttrenKenGen}(MSK, params, \omega),$$

$$dsk \leftarrow \textbf{ABPS.DskGen}(sk_\omega, m_w, \Upsilon),$$

then

$$\textbf{ABPS.ProVer}(\Upsilon, pk, m_w, m, \textbf{ABPS.ProSig}(dsk, sk, m)) = 1.$$

3.2 Security Model for ABPS

In an attribute-based proxy signature scheme, the security consideration is different from that for a traditional proxy signature or attribute-based signature. According to the definition of attribute-based proxy signature, we consider three different types of adversaries:

1. $\mathcal{A}_I$: an outsider attacker who only has the universe of attributes $\mathbb{A}$ and the public key pk_p of the proxy signer and tries to forge a valid proxy signature σ.
2. $\mathcal{A}_{II}$: a malicious proxy signer that possesses the private key sk_p and a valid warrant m_w from the original signer, and tries to forge a valid proxy signature σ for another warrant m_w^*.
3. $\mathcal{A}_{III}$: a malicious original signer that possesses the attribute key sk_ω and the public key pk_p of the proxy signer, and tries to forge a valid proxy signature σ without knowing the private key sk_p of the proxy signer.

It is obvious that if an attribute-based proxy signature scheme is secure under $\mathcal{A}_{II}$ or $\mathcal{A}_{III}$, it is also secure against $\mathcal{A}_I$. Thus we will only focus on the adversarial models with regards to $\mathcal{A}_{II}$ and $\mathcal{A}_{III}$ in the rest of this paper. Before we formally define each adversarial model, we first introduce three types of oracle queries that will appear in the models:

– **Attribute Key Generation Query**: $\mathcal{A}$ can query the attribute key for an attribute set $\omega \subseteq \mathbb{A}$ of his choice to the attribute key generation oracle $\mathcal{O}_{AKG}(\cdot)$. The corresponding attribute key sk_ω is then generated and returned to $\mathcal{A}$.
– **Delegation Query:** $\mathcal{A}$ can query the delegation oracle $\mathcal{O}_{DKG}(sk_\omega, \cdot, \cdot)$ with any warrant m_w and access structure Υ of his choice. The corresponding delegation key dsk is generated and returned to $\mathcal{A}$.
– **Proxy Signing Query:** $\mathcal{A}$ can query the proxy signing oracle $\mathcal{O}_{PS}(dsk, sk_p, \cdot)$ with any message m of his choice. A valid proxy signature on m is generated and returned to $\mathcal{A}$.

We define the selective adversarial game between a malicious proxy signer $\mathcal{A}_{II}$ and a simulator $\mathcal{S}$ as follows:

- **Initial Phase**: $\mathcal{A}_{II}$ chooses and outputs a challenge predicate Υ^* that will be used in forging a proxy signature.
- **ABPS.Setup Phase**: The simulator $\mathcal{S}$ runs **ABPS.Setup** to generate the $params$ and MSK, and sends $params$ to $\mathcal{A}_{II}$.
- **ABPS.KeyGen Phase**: The simulator $\mathcal{S}$ also runs the **ABPS.KeyGen** to generate the key pairs (pk_p, sk_p) of the proxy signer, and sends (pk_p, sk_p) to $\mathcal{A}_{II}$.
- **Attribute Key Generation Queries**: $\mathcal{A}_{II}$ selects an attribute set $\omega \in \mathbb{A}$, the simulator $\mathcal{S}$ runs $sk_\omega \leftarrow$ **ABPS.AttrKeyGen**$(MSK, params, \omega)$ and returns sk_ω to $\mathcal{A}_{II}$.
- **Delegation Queries Phase**: $\mathcal{A}_{II}$ chooses any predicate Υ such that $\Upsilon \neq \Upsilon^*$ and any warrant m_w of his choice and queries the delegation oracle $\mathcal{O}_{DKG}$. $\mathcal{S}$ generates the delegation key $dsk \leftarrow$ **ABPS.DskGen**$(sk_\omega, \Upsilon, m_w)$ and sends dsk to $\mathcal{A}$.
- **Proxy Signing Queries Phase**: $\mathcal{A}_{II}$ chooses a warrant $m_w \in \mathbb{M}_W$ and a message $m \in \mathbb{M}$ and queries the proxy signing oracle $\mathcal{O}_{PS}$. If m_w has appeared in a Delegation Query, a special symbol '$\perp$' is returned to $\mathcal{A}_{II}$. Otherwise, $\mathcal{S}$ generates

$$dsk \leftarrow \textbf{ABPS.DskGen}(sk_\omega, \Upsilon, m_w),$$

$$\sigma \leftarrow \textbf{ABPS.ProSign}(dsk, sk_p, m_w, m)$$

  and returns σ to $\mathcal{A}_{II}$.
- **Forgery Phase**: Finally, $\mathcal{A}$ outputs a proxy signature σ^* on message m^* for a warrant m_w^* and the predicate Υ^*.

We say $\mathcal{A}_{II}$ wins the game if

- **ABPS.ProVer**$(\Upsilon^*, pk_p, m_w^*, m^*, \sigma^*) = 1$;
- (m_w^*, Υ^*) has not been queried to $\mathcal{O}_{DSK}$;
- Attribute sets ω^* satisfying $\Upsilon^*(\omega^*) = 1$ have not been submitted to the attribute key generation oracle $\mathcal{O}_{AKG}$.

Define the advantage of a malicious adversary $\mathcal{A}_{II}$ in winning the game as

$$Adv_{\mathcal{A}_{II}}^{spcwcma}(\kappa) = \Pr[\mathcal{A}_{II} \text{ Wins the game}].$$

Definition 1. *We say an attribute-based proxy signature scheme is secure against the $\mathcal{A}_{II}$ under the selective-predicate and chosen warrant and message attacks if for any probabilistic polynomial time $\mathcal{A}_{II}$, $Adv_{\mathcal{A}_{II}}^{spcwcma}(\kappa)$ is negligible in κ.*

The adversarial game between a malicious original signer $\mathcal{A}_{III}$ and a simulator $\mathcal{S}$ is defined as follows:

- **ABPS.Setup Phase**: The simulator $\mathcal{S}$ runs the **ABPS.Setup** to generate the $params$ and MSK, and sends $params$ and MSK to $\mathcal{A}_{III}$.
- **ABPS.KeyGen Phase**: The simulator generates

$$(pk_p, sk_p) \leftarrow \textbf{ABPS.KeyGen}$$

  and sends pk_p to $\mathcal{A}_{III}$.
- **Proxy Signing Queries Phase**: $\mathcal{A}_{III}$ queries the proxy signing oracle $\mathcal{O}_{PS}$ by providing a warrant m_w, a valid delegation key dsk for m_w, and a message m of his choice. The simulator $\mathcal{S}$ generates the proxy signature $\sigma \leftarrow \textbf{ABPS.ProSign}(dsk, sk_p, m_w, m)$ and returns σ to $\mathcal{A}_{III}$.
- **Forgery Phase**: Finally, $\mathcal{A}_{III}$ outputs a proxy signature σ^* on message m^* for a warrant m_w^* and predicate Υ^*.

We say $\mathcal{A}_{III}$ wins the game if

- $\textbf{ABPS.PorVer}(\Upsilon^*, pk_p, m_w^*, m^*, \sigma^*) = 1$;
- (m_w^*, m^*) has not been queried to $\mathcal{O}_{PS}$;

Define the advantage of a malicious adversary $\mathcal{A}_{III}$ in winning the game as

$$Adv_{\mathcal{A}_{III}}^{cma}(\kappa) = \Pr[\mathcal{A}_{III} \text{ Wins the game}].$$

Definition 2. *We say an attribute-based proxy signature scheme is secure against the $\mathcal{A}_{III}$ under chosen message attacks if for any probabilistic polynomial time $\mathcal{A}_{III}$, $Adv_{\mathcal{A}_{III}}^{cma}(\kappa)$ is negligible in κ.*

4 Attribute-Based Proxy Signature Scheme

In our system, the original signer holds a set of attributes and delegates his signing rights to a proxy signer with a normal public/private key pair.

1. **ABPS.Setup**: First, define the universe of attributes U as elements in $\mathbb{Z}_p$. Let the $d-1$ default set of attributes from $\mathbb{Z}_p$ which has no intersection with U be $\Omega = \{\Omega_1, \Omega_2, \ldots, \Omega_{d-1}\}$ and let ω^* be another default attribute set with $\omega^* \subseteq U$. Select a random generator $g \in_R \mathbb{G}_1$ and a random number $x \in \mathbb{Z}_p^*$, set $g_1 = g^x$. Pick random elements g_2 and compute $Z = e(g_1, g_2)$. Select a collision resistant hash function $H : \{0,1\}^* \rightarrow \mathbb{G}_1$. The public parameters are $params = (g, g_1, g_2, Z, H)$. The master secret key is $MSK = x$.
2. **ABPS.KeyGen**: The user selects one random number $x_p \in_R \mathbb{Z}_p^*$ and set the private and public key pair as $(sk_p, pk_p) = (x_p, g^{x_p})$.
3. **ABPS.AttrKeyGen**: To generate a private key for an attribute set ω, proceed as follows:
   - Choose a $d-1$ polynomial q such that $q(0) = x$;
   - Generate a new set of attribute $\hat{\omega} = \omega \cup \Omega$. For each $i \in \hat{\omega}$, choose $r_i \in_R \mathbb{Z}_p$ and compute $d_{i0} = g_2^{q(i)} \cdot H(attr_i)^{r_i}$ and $d_{i1} = g^{r_i}$;
   - The private key $D_i = \{(d_{i0}, d_{i1})\}$, $i \in \hat{\omega}$.

4. **ABPS.DskGen**: Given a warrant m_w, the original signer selects a k-element subset $\omega' \subseteq \omega \cap \omega^*$ and the delegation signing key is generated as follows:
   - The original signer selects a default attribute subset $\Omega' \subseteq \Omega$ with $|\Omega'| = d - k$, chooses $n + d - k$ random values $r'_i \in \mathbb{Z}_p$, where $i \in \omega^* \cup \Omega'$;
   - The original signer chooses a random value $s \in \mathbb{Z}_p$ and computes $\sigma_0 = \prod_{i \in \omega \prime \cup \Omega'} d_{i0}^{\Delta_{i,S}(0)} \prod_{i \in \omega^* \cup \Omega'} H(attr_i)^{r'_i} H(m_w)^s$, $\{\sigma_i = d_{i1}^{\Delta_{i,S}(0)} g^{r'_i}\}_{i \in \omega' \cup \Omega'}$, $\{\sigma_i = g^{r'_i}\}_{\omega^*/\omega'}$, $\sigma'_0 = g^s$;
   - The delegation signing key is $dsk = (\sigma_0, \{\sigma_i\}_{i \in \omega^* \cup \Omega'}, \sigma'_0)$.
5. **ABPS.ProSign**: Given dsk, sk_p and a message $m \in \{0,1\}^*$. The proxy signature $\sigma_M = (\sigma_{M_1}, \sigma_{M_2}, \sigma_{M_3}, \sigma_{M_4})$ is generated as follows:
   - Compute $\sigma_{M_1} = \sigma_0 \cdot H(m)^{sk_p}$, $\sigma_{M_2} = \{\sigma_i\}_{i \in \omega^* \cup \Omega'}$, $\sigma_{M_3} = \sigma'_0$.
6. **ABPS.Verification**: Given $\sigma_M = (\sigma_{M_1}, \sigma_{M_2}, \sigma_{M_3}, \sigma_{M_4})$, m_w and Υ_{k,ω^*}, the verifier first check whether the proxy signer follow the rules specified in the warrant. If no, output reject, otherwise, the verifier checks the following equation:

$$\frac{e(g, \sigma_{M_1})}{\prod_{i \in \omega^* \cup \Omega'} e(H(attr_i), \sigma_i) e(H(m_w), \sigma_{M_3}) e(pk_p, H(m))} \stackrel{?}{=} Z.$$

   If the equation holds, output accept, where it can be assured that the signature is generated form some user possessing k attributes among ω^*, otherwise, output reject.

- **Correctness:** The correctness of the verification is justified by the following equations:

$$\begin{aligned}
&\frac{e(g, \sigma_{M_1})}{\prod_{i \in \omega^* \cup \Omega'} e(H(attr_i), \sigma_i) e(H(m_w), \sigma_{M_3}) e(pk_p, H(m))} \\
&= \frac{e(g, \sigma_0 \cdot H(m)^{sk_p})}{\prod_{i \in \omega^* \cup \Omega'} e(H(attr_i), \sigma_i) e(H(m_w), \sigma_{M_3}) e(pk_p, H(m))} \\
&= \frac{e(g, \prod_{i \in \omega \prime \cup \Omega'} d_{i0}^{\Delta_{i,S}(0)} \prod_{i \in \omega^* \cup \Omega'} H(attr_i)^{r'_i} H(m_w)^s \cdot H(m)^{sk_p})}{\prod_{i \in \omega^* \cup \Omega'} e(H(attr_i), \sigma_i) e(H(m_w), \sigma_{M_3}) e(pk_p, H(m))} \\
&= \frac{e(g, \prod_{i \in \omega \prime \cup \Omega'} d_{i0}^{\Delta_{i,S}(0)}) e(g, \prod_{i \in \omega^* \cup \Omega'} H(attr_i)^{r'_i}) e(H(m_w), g^s) e(pk_p, H(m))}{\prod_{i \in \omega^* \cup \Omega'} e(H(attr_i), \sigma_i) e(H(m_w), \sigma_{M_3}) e(pk_p, H(m))} \\
&= \frac{e(g, \prod_{i \in \omega \prime \cup \Omega'} d_{i0}^{\Delta_{i,S}(0)}) e(g, \prod_{i \in \omega^* \cup \Omega'} H(attr_i)^{r'_i})}{\prod_{i \in \omega^* \cup \Omega'} e(H(attr_i), \sigma_i)} \\
&= \frac{e(g, \prod_{i \in \omega \prime \cup \Omega'} g_2^{q(i) \cdot \Delta_{i,S}(0)}) e(g, \prod_{i \in \omega \prime \cup \Omega'} H(attr_i)^{r_i \cdot \Delta_{i,S}(0) + r'_i}) e(g, \prod_{i \in \omega^*/\omega'} H(attr_i)^{r'_i})}{e(g, \prod_{i \in \omega \prime \cup \Omega'} H(attr_i)^{r_i \cdot \Delta_{i,S}(0) + r'_i}) e(g, \prod_{i \in \omega^*/\omega'} H(attr_i)^{r'_i})} \\
&= e(g, g_2^{q(0)}) \\
&= e(g, g_2^x) \\
&= e(g^x, g_2) \\
&= Z.
\end{aligned}$$

5 Security Analysis

In this section we analyse the security of the above attribute-based proxy signature scheme against $\mathcal{A}_{II}$ and $\mathcal{A}_{III}$ adversaries.

Theorem 1. *Our attribute-based proxy signature scheme is secure against the $\mathcal{A}_{II}$ chosen warrant and chosen message attacks if the CDH Problem is hard.*

Proof. The proof is by contradiction in the selective predicate security model. Suppose that an adversary $\mathcal{A}_{II}$ has an advantage ϵ in attacking the proposed scheme, then we can build an algorithm $\mathcal{B}$ that use $\mathcal{A}_{II}$ to solve the CDH problem. Let $\mathbb{G}_1$ be a bilinear pairing group of prime order p, $\mathcal{B}$ is given $g, g^{\alpha}, g^{\beta} \in \mathbb{G}_1$ which is a random instance of the CDH problem. Its goal is to compute $g^{\alpha\beta}$. Algorithm $\mathcal{B}$ will simulate the challenger and interact with the forger $\mathcal{A}_{II}$ as described below, let's recall the definition of $\mathcal{A}_{II}$, $\mathcal{A}_{II}$ is a malicious proxy signer possessing the private key of the proxy signer. With this in mind, the simulation is as follows:

1. **Initial Phase**: $\mathcal{A}_{II}$ chooses a predicate Υ^*_{k,ω^*} as the challenge predicate.
2. **Setup**: Let the default attribute set Ω be $\{\Omega_1, \ldots, \Omega_{d-1}\}$. $\mathcal{B}$ sets $g_1 = g^{\alpha}, g_2 = g^{\beta}$, where g^{α}, g^{β} are inputs of the CDH problem. $\mathcal{B}$ sets the public parameters as:
   - $\mathcal{B}$ chooses random $x_p \in_R \mathbb{Z}_p^*$ and sets $sk_p = x_p, pk_p = g^{x_p}$.
   - $\mathcal{B}$ sends $(G_1, G_2, e, p, g, g_1, g_2, H)$ and (sk_p, pk_p) to $\mathcal{A}_{II}$.
3. **Hash queries**: In order to make the simulation easy to follow, we regard the attribute, warrant and message queries as H_1,H_2 and H_3 queries respectively. Assume $\mathcal{B}$ keeps hash tables T_1, T_2 and T_3 for the queries.
   (a) H_1 **Query**: Assume $\mathcal{A}_{II}$ makes q_{H1} attribute queries, for each query on attribute $attr_i$, $\mathcal{B}$ simulates as follows:
      - If $attr_i$ have existed in T_1, a same value $H(attr_i)$ is returned to $\mathcal{A}_{II}$.
      - Otherwise,
        - If $attr_i \in \omega^* \cup \Omega^*$, $\mathcal{B}$ chooses random $a_i \in \mathbb{Z}_p$ and returns $H(attr_i) = g^{a_i}$ to $\mathcal{A}_{II}$. $\mathcal{B}$ adds $(attr_i, H(attr_i))$ to T_1.
        - If $attr_i \notin \omega^* \cup \Omega^*$, $\mathcal{B}$ chooses random $a_i, b_i \in \mathbb{Z}_p$ and returns $H(attr_i) = g^{-a_i} g^{b_i}$ to $\mathcal{A}_{II}$. $\mathcal{B}$ adds $(attr_i, H(attr_i))$ to T_1.

   (b) H_2 **Query**: Assume $\mathcal{A}_{II}$ makes q_{H2} warrant queries, $\mathcal{B}$ selects a random number $\delta \in (0, q_{H2})$, for each query on warrant m_{w_i}, $\mathcal{B}$ simulates as follows:
      - If m_{wi} have existed in T_2, a same value $H(m_{w_i})$ is returned to $\mathcal{A}_{II}$.
      - Otherwise,
        - If $i \neq \delta$, $\mathcal{B}$ chooses random $a'_i, b'_i \in_r \mathbb{Z}_p$ and returns $H(m_{w_i}) = g_1^{b'_i} g^{a'_i}$. $\mathcal{B}$ adds $(m_{w_i}, H(m_{w_i}))$ to T_2.
        - If $i = \delta$, $\mathcal{B}$ chooses random b'_i and returns $H(w_i) = g^{a'_i}$. $\mathcal{B}$ adds $(m_{w_i}, H(m_{w_i}))$ to T_2

   (c) H_3 **Query**: Assume $\mathcal{A}_{II}$ makes q_{H3} message queries, for each query on message m_i, $\mathcal{B}$ simulates as follows:
      - If m_i has existed in T_3, a same value $H(m_i)$ is returned to $\mathcal{A}_{II}$.
      - Otherwise, $\mathcal{B}$ chooses random $r_i \in_R \mathbb{Z}_p$ and returns $H(m_i) = g^{r_i}$. $\mathcal{B}$ adds $(m_i, H(m_i))$ to T_3.

4. **Attribute key extraction queries**: Assume $\mathcal{A}_{II}$ issues an attribute key extraction query on an attribute set ω such that $|\omega^* \cap \omega| < k$. Following the analysis in [4], we first define three sets Γ, Γ', S in the following manner: $\Gamma = (\omega \cap \omega^*) \cup \Omega^*$ and $\Gamma \subseteq \Gamma' \subseteq S$ with $|\Gamma'| = d-1$. Let $S = \Gamma' \cup \{0\}$. The simulation on the attribute key D_i is as follows:
   - For $i \in \Gamma'$: $D_i = (g_2^{\tau_i} H(attr_i)^{r_i}, g^{r_i})$, where $\tau_i, r_i \in_R \mathbb{Z}_p$.
   - For $i \notin \Gamma'$, D_i could be simulated as:

$$D_i = (g_2^{\frac{\Delta_{0,S}(i)b_i}{a_i} + \sum_{j \in \Gamma'} \Delta_{j,S}(i)q(j)})(g_1^{-a_i} g^{b_i})^{r'_i}, g_2^{\frac{\Delta_{0,S}(i)}{a_i}} g^{r'_i}),$$

   where $r'_i \in_R \mathbb{Z}_p$. It is a correct key because it implicitly sets

$$r_i = \frac{\Delta_{j,S}(i)q(j)}{a_i}\beta + r'_i.$$

   As we know,

$$q(i) = \sum_{j \in \Gamma'} \Delta_{j,S}(i)q(j) + \Delta_{0,S}(i)q(0),$$

   thus we have,

$$g_2^{q(i)} H(attr_i)^{r_i} = g_2^{\frac{\Delta_{0,S}(i)b_i}{a_i} + \sum_{j \in \Gamma'} \Delta_{j,S}(i)q(j)} H(attr_i)^{r'_i}$$

   and

$$g^{r_i} = g_2^{\frac{\Delta_{0,S}(i)}{a_i}} g^{r'_i}.$$

5. **Delegation signing key queries**: $\mathcal{A}_{II}$ can also issue a query for a warrant W for an attribute set ω with k' values out of an n'-value attribute set ω. The delegation signing key query could be simulated as follows:
   - If $|\omega \cup \omega^*| < k$, $\mathcal{B}$ can generate a simulated private key for ω as in the attribute key simulation and get a signature for ω on W normally.
   - If $|\omega \cap \omega^*| > k$, $\mathcal{B}$ selects a random $(d-k')$-element subset Ω' from Ω. If $H(W) \neq g^{a_i}$, in order to simulate $(g_2^x \prod_{i \in \omega \cup \Omega'} H(attr_i)^{r_i} H(w)^{r_a}, \{g^{r_i}\}_{i \in \omega \cup \Omega'}, g^{r_a})$
     - Choose $r'_a \in \mathbb{Z}_p$ and set $r'_a = \frac{1}{c}\beta + r_a$. Then

$$g_2^x \prod_{i \in \omega \cup \Omega'} H(attr_i)^{r_i} H(w)^{r_a} = (g_1^c g^{a_i})^{r'_a} \prod_{i \in \omega \cup \Omega'} H(attr_i)^{r_i} g_2^{-\frac{a_i}{c}},$$

$$g^{r_a} = g_2^{\frac{-1}{c}} g^{r'_a}$$

     when $H(W) = g_1^c g^{a_i}$.
6. **Proxy signing queries**: Assume $\mathcal{A}_{II}$ makes q_{ps} proxy signing queries. If $\mathcal{A}_{II}$ issues a proxy signature queries for a message $m \in \{0,1\}^*$ under a warrant W for a predicate Υ, in order to simulate $\sigma = \sigma_0 \cdot H(m)^{sk_p}$, $\mathcal{B}$ generates the delegation signing key σ_0 as in the **delegation signing queries** and answers the proxy signing queries as follows:

- If m_i has existed in T_3, then return $\sigma = \sigma_0 \cdot pk_p^{r_i}$ as the proxy signature to $\mathcal{A}_{II}$, where $H(m_i) = g^{r_i}$ exists in T_3.
- If m_i does not appear in T_3, then choose random $r_i \in \mathbb{Z}_p$ and return $\sigma = \sigma_0 \cdot pk_p^{r_i}$ as the proxy signature to $\mathcal{A}_{II}$. $\mathcal{B}$ adds $(m_i, H(m_i))$ to T_3.

7. **Forgery**: Assume $\mathcal{A}_{II}$ outputs a valid proxy signature

$$\sigma^* = (\sigma_0^*, \{\sigma_i^*\}_{i \in \omega^* \cup \Omega^*}, \sigma_0')$$

for predicate Υ_{k,ω^*}^*. If $H(m_w) \neq g^{a_\delta'}$ or $\overline{\Omega^*} \neq \Omega^*$ where $\overline{\Omega^*}$ are the dummy attributes, $\mathcal{B}$ will abort. Therefore

$$\begin{aligned}\sigma^* &= (\sigma_0^*, \{\sigma_i^*\}_{i \in \omega^* \cup \Omega^*}, \sigma_0') \\ &= (g_2^\alpha \prod_{i \in \omega^* \cup \Omega^*} H(attr_i)^{r_i} H(m_w)^{r_a} H(m)^{sk_p}, \{g^{r_i}\}_{i \in \omega^* \cup \Omega^*}, g^{r_a}).\end{aligned}$$

Thus $\mathcal{B}$ can compute

$$g^{\alpha\beta} = \frac{\sigma_0^*}{\prod_{i \in \omega^* \cup \Omega^*} (\sigma_i^*)^{a_i} (\sigma_0')^{a_\delta'} (pk_p)^{r_i}}$$

because $H(attr_i) = g^{a_i}$, $H(m_w) = g^{a_\delta'}$.

Next, we analysis the success probability of $\mathcal{B}$, $\mathcal{B}$ will not abort if the following conditions holds:

- $H(m_w) = g^{a_\delta}$.
- Correct guess of $d - k$ elements Ω^* from Ω.

Therefore the success probability of $\mathcal{B}$ in solving CDH problem is:

$$Succ_{\mathcal{B}}^{CDH} = \frac{\epsilon}{q_{H2} \mathrm{C}_{d-1}^{d-k}}.$$

Theorem 2. *Our attribute-based proxy signature scheme is secure against the $\mathcal{A}_{III}$ chosen message attacks if the CDH Problem is hard.*

Proof. Let $\mathbb{G}_1$ be a bilinear pairing group of prime order p. Algorithm $\mathcal{B}$ is given $g, g^\alpha, g^\beta \in \mathbb{G}_1$ which is a random instance of the CDH problem. Its goal is to compute $g^{\alpha\beta}$. Algorithm $\mathcal{B}$ will simulate the challenger and interact with the adversary $\mathcal{A}_{III}$ as described below.

Let's recall the definition of the adversary $\mathcal{A}_{III}$. $\mathcal{A}_{III}$ has the attribute key of the original signer as well as the public of the proxy signer, thus the attribute key extraction and delegation queries are not needed here. The simulation is performed as follows:

1. **Setup**: $\mathcal{B}$ sets the public keys of the users and the common parameter as :
   - $\mathcal{B}$ selects a random generator $g \in_R \mathbb{G}_1$ and two random number $x, g_2 \in_R \mathbb{Z}_p^*$, then $\mathcal{B}$ chooses a $d-1$ degree polynomial q with $q(0) = x$ and computes $g_1 = g^x$. $\mathcal{B}$ sets $sk_p = \alpha, pk_p = g^\alpha$, where g^α, g^β are inputs of the CDH problem.
   - $\mathcal{B}$ then sends $(\mathbb{G}_1, \mathbb{G}_2, e, p, g, x, g_1, g_2, H)$ and pk_p to $\mathcal{A}_{III}$.
2. **Hash queries**:Assume $\mathcal{A}_{III}$ makes q_{H1}, q_{H2}, q_{H3} times for attribute, warrant and message queries, respectively. $\mathcal{B}$ maintains hash tables T_1, T_2, T_3 for attribute, warrant and message queries. For the hash queries for the attribute and warrant, $\mathcal{B}$ performs the same as in Theorem 1. For the message query on any m of $\mathcal{A}_{III}$'s choice, $\mathcal{B}$ chooses a random number $I \in (1, q_{H3})$, for each query on message m_i, if $(m_i, H(m_i))$ exits in hash table, $\mathcal{B}$ just returns $H(m_i)$ to $\mathcal{A}_{III}$, otherwise, the simulation is performed as follows:
   - If $m_i \neq m_I$, $\mathcal{B}$ chooses random $r_i \in_R \mathbb{Z}_p$, returns $H(m_i) = g^{r_i}$ and adds $(m_i, H(m_i))$ to T.
   - If $m_i = m_I$, $\mathcal{B}$ chooses random $r_I \in_R \mathbb{Z}_p$, return $H(m_I) = (g^\beta)^{r_I}$.
3. **Proxy Signing Queries**: Suppose $\mathcal{A}_{III}$ issues a proxy signing query for a message $M \in \{0,1\}^*$ under a warrant W with predicate Υ_{k,ω^*}. $\mathcal{B}$ first generates the attribute key sk_ω using the same method as the **attribute key extraction queries** in Theorem 1. Then $\mathcal{B}$ generates the delegation key $dsk = (\sigma_0, \{\sigma_i\}_{i \in \omega^* \cup \Omega'}, \sigma_0')$ using the method same as **Delegation signing key queries** in Theorem 1. Then $\mathcal{B}$ simulates the proxy signature queries as follows:
   - If $M \in T_3$, assume $H(M) = g^{r_M}$, $\mathcal{B}$ simulates the proxy signature $\sigma = (\sigma_1, \sigma_2, \sigma_3)$ where $\sigma_1 = \sigma_0 \cdot pk_p^{r_M}$, $\sigma_2 = \{\sigma_i\}_{i \in \omega^* \cup \Omega'}$ and $\sigma_3 = \sigma_0'$.
   - If $M \notin T_3$, $\mathcal{B}$ chooses random $r^* \in_R \mathbb{Z}_p$ and simulates the proxy signature as $\sigma_1 = \sigma_0 \cdot pk_p^{r^*}$, $\sigma_2 = \{\sigma_i\}_{i \in \omega^* \cup \Omega'}$ and $\sigma_3 = \sigma_0'$. $\mathcal{B}$ adds $(M, H(M))$ to hash table T.
4. **Forgery**: Assume that the adversary $\mathcal{A}_{III}$ can output a proxy signature $\sigma^* = (\sigma_1^*, \sigma_2^*, \sigma_3^*,)$ of the message M^* under the warrant W^* for predicate Υ^* such that:
   - (M^*, W^*) has not been submitted as one of the proxy signing queries.
   - $\sigma^* = (\sigma_1^*, \sigma_2^*, \sigma_3^*)$ is a valid proxy signature.

   In this case, if $M^* = m_I$, $\mathcal{B}$ can compute:

$$g^{\alpha\beta} = \left(\frac{\sigma_1^*}{g_2^x \prod_{i \in \omega \cup \Omega^*} (\sigma_2^*)^{a_i} (\sigma_3^*)^{a'_\delta}}\right)^{\frac{1}{r_I}}$$

Next, we analyse the success probability of $\mathcal{B}$. $\mathcal{B}$ will not abort if the following conditions hold:

- $H(m_w) = g^{a_\delta}$.
- $H(M) = (g^\beta)^{r_I}$.
- Correct guess of $d-k$ elements Ω^* from Ω.

Therefore,

$$Succ_{\mathcal{B}}^{CDH} = \frac{\epsilon}{q_{H2}(q_{H3} + q_{ps})\mathrm{C}_{d-1}^{d-k}}.$$

6 Conclusion

In this paper, we studied attribute-based proxy signature (ABPS) for threshold predicates. We presented a formal security model and a concrete construction of ABPS scheme. Our model has considered different types of potential adversaries against an ABPS scheme. An interesting feature of our scheme is that it offers original signer privacy, that is even the proxy signer cannot find out who is the original signer except that the original signer's attributes satisfy a pre-claimed predicate. We leave the problem of building ABPS for other types of predicates as our future work.

References

1. Foster, I., Kesselman, C., Tsudik, G., Tuecke, S.: A security architecture for computational grids. In: Proceedings of the 5th ACM Conference on Computer and Communications Security, pp. 83–92. ACM (1998)
2. Kim, S., Park, S., Won, D.: Proxy signatures, revisited. In: Han, Y., Quing, S. (eds.) ICICS 1997. LNCS, vol. 1334, pp. 223–232. Springer, Heidelberg (1997)
3. Lee, B., Kim, H., Kim, K.: Strong proxy signature and its applications. In: Proc of SCIS, vol. 1, pp. 603–608 (2001)
4. Li, J., Au, M.H., Susilo, W., Xie, D., Ren, K.: Attribute-based signature and its applications. In: ASIACCS, pp. 60–69 (2010)
5. Liu, W., Yang, G., Mu, Y., Wei, J.: k-time proxy signature: Formal definition and efficient construction. In: Susilo, W., Reyhanitabar, R. (eds.) ProvSec 2013. LNCS, vol. 8209, pp. 154–164. Springer, Heidelberg (2013)
6. Liu, X., Ma, J., Xiong, J., Zhang, T., Li, Q.: Personal health records integrity verification using attribute based proxy signature in cloud computing. In: Pathan, M., Wei, G., Fortino, G. (eds.) IDCS 2013. LNCS, vol. 8223, pp. 238–251. Springer, Heidelberg (2013)
7. Maji, H.K., Prabhakaran, M., Rosulek, M.: Attribute-based signatures. In: Kiayias, A. (ed.) CT-RSA 2011. LNCS, vol. 6558, pp. 376–392. Springer, Heidelberg (2011)
8. Mambo, M., Usuda, K., Okamoto, E.: Proxy signatures for delegating signing operation. In: ACM Conference on Computer and Communications Security, pp. 48–57 (1996)
9. Clifford Neuman, B.: Proxy-based authorization and accounting for distributed systems. In: Proceedings of the13th International Conference on Distributed Computing Systems, pp. 283–291. IEEE (1993)
10. Okamoto, T., Takashima, K.: Efficient attribute-based signatures for non-monotone predicates in the standard model. IACR Cryptology ePrint Archive, 2011, 700 (2011)
11. Wang, G.: Designated-verifier proxy signature schemes. In: Sasaki, R., Qing, S., Okamoto, E., Yoshiura, H. (eds.) Security and Privacy in the Age of Ubiquitous Computing. IFIP, vol. 181, pp. 409–424. Springer, Boston (2005)
12. Wu, W., Mu, Y., Susilo, W., Seberry, J., Huang, X.: Identity-based proxy signature from pairings. In: Xiao, B., Yang, L.T., Ma, J., Muller-Schloer, C., Hua, Y. (eds.) ATC 2007. LNCS, vol. 4610, pp. 22–31. Springer, Heidelberg (2007)
13. Xu, J., Zhang, Z., Feng, D.: ID-based proxy signature using bilinear pairings. In: Chen, G., Pan, Y., Guo, M., Lu, J. (eds.) ISPA-WS 2005. LNCS, vol. 3759, pp. 359–367. Springer, Heidelberg (2005)
14. Yu, Y., Mu, Y., Susilo, W., Sun, Y., Ji, Y.: Provably secure proxy signature scheme from factorization. Mathematical and Computer Modelling 55(3-4), 1160–1168 (2012)