Attribute-Based Server-Aided Verfication Signature

Zhiwei Wang, Ruirui Xie, Wei Zhang, Liwen He, Guozi Sun, and Wei Chen

1.College of Computer, Nanjing University of Posts and Telecommunications,Nanjing, Jiangsu 210003, China
zhwwang@njupt.edu.cn

Abstract. Attribute based signature (ABS) is a novel cryptographic primitive, which enables a party can sign messages for any predicate satisfy by their attributes. However, heavy computational cost is required during the verification procedure in most existing ABS schemes, which may needs many pairing operations. Pairing are costly operation when compared to exponentiation in the base group. As a result, this presents a greatly challenge for resource-limited users, such as smart cards and wireless sensor. In other words, verification can hardly be done in these devices if attribute based signature is employed. We solve this problem by proposing a new notion called *Attribute-Based Server-Aided Verification Signature*. It is similar to normal ABS scheme, but it further enables the verifier to verify the signature with the assistance of an external server. In this paper, we provide the security definition of Attribute-Based Server-Aided Verification Signature, and design a concrete server-aided verification protocol for Li et al.'s attribute based signature. We also prove that our protocol is secure with random oracles.

Key words: attribute based signature; server-aided verification; pairing; resource-limited user

1 Introduction

Attribute based signature (ABS) is a novel cryptographic primitive, which extends the identity based signatures in which a signer is defined by a set of attributes instead of a single string representing the signer's identity. In ABS, a user obtains his attribute secret key for a set of attributes from an attribute authority, with which they can later sign message for any predicate satisfied by their attributes. If the signature is valid, then the verifier will be convinced that the signer's attributes satisfy the signing predicate while remaining completely ignorant of the identity of signer. ABS has found many important applications[1], such as private access control, anonymous credential, trust negotiations, distributed access control, attribute based messaging, etc.

However, one of the main efficiency drawbacks of ABS is that the verification procedure requires heavy computational cost. Some existing ABS schemes[2, 1, 3], a large number of pairings are needed in verification, which commonly grows linearly with the size of predicate formula. Pairings are costly operation when compared to the exponentiation in the base group - when pairings are used on an elliptic curve defined over a field of q elements, the last operation of the pairing is an exponentiation in a field of q^k elements, where k is the embedding degree of the elliptic curve, which computational cost is much more heavy than an exponentiation on the elliptic curve. Although some researchers [4] have dramatically reduced the computational cost of pairings, their technique requires the simultaneous computation of many pairings, which may not be suitable for the memory restrict devices. Recently, Herranz et al.[5] and Gagné et al.[6] propose the short pairing-efficient ABS schemes. However, both of their schemes are very inefficient in the signing algorithm.

In this paper, we introduce a new notion called *Attribute-Based Server-Aided Verification Signature* (ABSAVS). There is no difference between an ABS and ABSAVS in the signing process. On the other side, there is an additional server which helps the verifier to verify the signature. While the verifier received a ABS signature, he computes a transformed signature, which contains the information that should be initialized, and sends it to the server. It generates a token after executing most of pairing computations and sends the token back to the verifier. Finally the verifier verifies the signature from this token, with lightweight computational cost.

1.1 Our Contribution

In this paper, we firstly analysis Li et al.'s outsourced verification scheme[7], and show that it can be forged by collusion between untrusted server and outside attacker[8]. Then, we propose a security definition of *Attribute-Based Server-Aided Verification Signature* (ABSAVS), and design a concrete ABSAVS scheme from Li et al.'s ABS scheme[2]. We employ the Lagrange interpolation formula in this protocol. With the help of the server, the number of pairing involving in verification is greatly reduced from $O(|\Omega^*|)$ to 2 (These two pairings also can be pre-computed offline.), Ω^* is the attribute set in the threshold predicate included in the signature.

1.2 Related Work

Attribute based signature: The first normal definition of ABS was presented Maji et al. [9], but the security of their scheme is based on the generic group model. Li et al. [2] and Shahandashit et al. [1] proposed the ABS schemes that support threshold predicate in standard model. Nevertheless, both of their schemes require $O(|\Omega^*|)$ pairings in verification, where Ω^* is the attribute set in the threshold predicate included in the signature. In 2011, Escala et al.[3] presented an ABS scheme supporting flexible threshold predicate, which shares the similar efficiency with Li et al.'s work in verification [2]. In 2012, Herranz et al.[5] and Gagné et al.[6] proposed the threshold predicate ABS schemes with constant size signatures. But their schemes are both very inefficient in the signing algorithm. Recently, Li et al. [7] presented a outsourced verify protocol by using Wu et al.'s technique [8]. However, this protocol cannot resist the collusion of untrusted server and outside attacker. We specify that many existing work of ABS requires a large number of pairing computation in verification. The complexity of commonly grows linearly with the size of the predicate formula in threshold ABS.

Serever-Aided Verification Signature: Employing a powerful server to assist the low power device to carried out cryptographic operations is a promising solution to reduce the computational cost, which is known as "server-aided computation". However, in practice, users are more likely to face an untrusted server which could try to extract the secret of the user or respond with a false result. To resist the untrusted server, many schemes for server-aided verification signature have been proposed in the literature. The notion was introduced by Quisquater and De Soete [10] for speeding up RSA verification with a small exponent. Lim and Lee [11] introduced this idea into discrete-logarithm based schemes, by proposing efficient protocols for speeding up the verification of discrete-logarithm based identity proofs and signatures. The server-aided verification protocol introduced by Girault and Quisquater [12] is computational secure based on the hardness of a sub-problem of the underlying complexity problem in the original signature scheme. Girault and Lefrance [13] proposed a more generalized model of server-aided verification without the assumption of [11]. Wu et al. [8] formally define the security model for capturing collusion attacks, and propose concrete server-aided verification signature schemes that are secure against such attacks. Li et al. [7] presented a outsourced verify protocol by using Wu et al.'s technique. In this paper, we will show that their protocol is not secure.

1.3 organization

This paper is organized as follows. In Section 2, we describe some preliminaries, and review Li et al.'s ABS scheme [2]. In Section 3, we analysis Li et al.'s outsourced verify protocol[7]. In Section 4, we present the security definition of *Attribute-Based Server-Aided Verification Signature*. In section 5, we propose a concrete attribute-based server-aided verification protocol from Li et al.'s ABS scheme, and prove that it is secure under random oracles in Section 6. Finally, we draw conclusion in Section 7.

2 Preliminaries

2.1 Bilinear Mapping

Bilinear Mapping: Let $\mathbb{G}_1$ and $\mathbb{G}_T$ be two groups of prime order p and g be generator of $\mathbb{G}_1$. The map $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$ is said to be an admissible bilinear mapping if the following three conditions hold true:

- e is bilinear, i.e., $e(g^a, g^b) = e(g, g)^{ab}$ for all $a, b \in \mathbb{Z}_p$.
- e is non-degenerate, i.e., $e(g, g) \neq 1_{\mathbb{G}_T}$.

– e is efficiently computable.

We say that $(\mathbb{G}_1, \mathbb{G}_T)$ are bilinear groups if there exists the bilinear mapping $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$ as above, and e, and the group action of in $\mathbb{G}_1$ and $\mathbb{G}_T$ can be computed efficiently. Such groups can be built from Weil pairing or Tate pairing on elliptic curves.

2.2 Complexity Assumptions

Computational Diffie-Hellman Problem(CDH): Give (g, g^a, g^b) for some $a, bin\mathbb{Z}_p^*$, compute g^{ab}. An algorithm $\mathcal{A}$ has advantage ε in solving CDH on $\mathbb{G}_1$ if

$$Pr[\mathcal{A}(g, g^a, g^b) = g^{ab} : a, b \in_R \mathbb{Z}_p^*] \geq \varepsilon.$$

The probability is over the uniform random choice of a, b from $\mathbb{Z}_p^*$ and over the coin tosses of $\mathcal{A}$.

2.3 Attribute based Signature

An ABS scheme consists of four algorithms, namely, *Setup*, *Extract*, *Sign*, and *Verify*. Denote the universe of attributes as U. A predicate over U is a monotone boolean function, whose inputs are associated with attributes in U. We say that an attribute set Ω satisfies a predicate Υ if $\Upsilon(\Omega) = 1$. More precisely, all predicates $\Upsilon_{k,\Omega^*}(\cdot) :\rightarrow \{0, 1\}$ for Ω^* with threshold k from 1 to d are supported, where d is a system parameter and

$$\Upsilon_{k,\Omega^*}(\Omega) = \begin{cases} 1 \, , \, |\Omega \bigcap \Omega^*| \geq k \\ 0 \, , \quad otherwise \end{cases}$$

– **Setup** On input 1^λ, where λ is the security parameter, this algorithm outputs public parameters *params* and *sk* as a master secret key for attribute authority;
– **Extract** For each user's private key request on attribute set Ω, this algorithm takes as input the master secret key sk and the attribute set Ω, it outputs the user's private key sk_Ω.
– **Sign** Assume a user wants to sign a message m with a predicate Υ and a set of attributes Ω' satisfying $\Upsilon_{k,\Omega}(\Omega') = 1$, he takes as input his attribute private key sk_Ω for attributes Ω, outputs signature σ.
– **Verify** After receiving a signature σ on message m and attributes Ω' with respect to a predicate Υ, the signature is valid if $\Upsilon_{k,\Omega}(\Omega') = 1$ and the signature is valid.

Security Definition: The security definition of ABS can be divided into two items[2]. The first one is *Unforgeability*, which requires that the ABS scheme is existentially unforgeable against chosen predicate and message attack. The second is *Attribute Signer Privacy*, which means that the signature reveals noting about the identity or attributes of the signer beyond what is explicitly revealed by the claim being made.

2.4 Li et al.'s ABS scheme

In this section, we will review the ABS scheme proposed by Li et al. [2]. We define the attributes in universe U as elements in $\mathbb{Z}_p$, and a $d-1$-element dummy attribute set $\hat{\Omega}$. Then, we define the Lagrange coefficient $\Delta_{j,S}(i)$ of $q(j)$ in computation of $q(i)$ as:

$$\Delta_{j,S}(i) = \prod_{\eta \in S, \eta \neq j} \frac{i-\eta}{j-\eta}.$$

Setup Select a random generator $g \in \mathbb{G}_1$, a random $x \in \mathbb{Z}_p^*$, and set $g_1 = g^x$. Then, choose a random element $g_2 \in \mathbb{G}_1$ and compute $Z = e(g_1, g_2)$. Two hash function are also chosen such that $H_1, H_2 : \{0,1\}^* \to \mathbb{G}_1$. Finally, output the public key $PK = (g, g_1, g_2, Z, d, H_1, H_2)$ and the master key $MK = x$.

Extract For each user's private key request on the attribute set Ω, choose a $d-1$ degree polynomial $q(y)$ randomly such that $q(0) = x$. Then, for each $i \in \Omega \cup \hat{\Omega}$, choose $r_i \in_R \mathbb{Z}_p$ and compute $d_{i0} = g_2^{q(i)} \cdot H_1(i)^{r_i}$ and $d_{i1} = g^{r_i}$. The private key is $D_i = (d_{i0}, d_{i1})$ for $i \in \Omega \cup \hat{\Omega}$.

Sign To sign a message m with predicate $\Upsilon_{k,\Omega^*}(\cdot)$, namely, to prove owing at least k attributes among an $n-$element attribute set Ω^*. Select an arbitrary k-element subset $\Omega' = \Omega^* \cap \Omega$. Furthermore, selects a dummy attribute set $\hat{\Omega}' \subseteq \hat{\Omega}$ with $|\hat{\Omega}'| = d-k$ and choose $n+d-k$ random values $r_i' \in \mathbb{Z}_p$ for $i \in \Omega^* \cup \Omega'$. Finally, the signer computes $\sigma_0 = [\prod_{i \in \Omega' \cup \hat{\Omega}'} d_{i0}^{\Delta_{i,S}(0)}] \cdot [\prod_{i \in \Omega^* \cup \hat{\Omega}'} H_1(i)^{r_i'}] \cdot H_2(m)^s$, $\{\sigma_i = d_{i1}^{\Delta_{i,S}(0)} g^{r_i'}\}_{i \in \Omega' \cup \hat{\Omega}'}$, $\{\sigma_i = g^{r_i'}\}_{\Omega^*/\Omega'}$, and $\sigma_0' = g^s$, with a random $s \in$ ı. The signature is $\sigma = (\sigma_0, \{\sigma_i\}_{i \in \Omega^* \cup \hat{\Omega}'}, \sigma_0')$.

Verify Once received the signature $\sigma = (\sigma_0, \{\sigma_i\}_{i \in \Omega^* \cup \hat{\Omega}'}, \sigma_0')$ of the message m with threshold k for attributes $\Omega^* \cup \hat{\Omega}'$, check if the following equation holds:

$$\frac{e(g, \sigma_0)}{[\prod_{i \in \Omega^* \cup \hat{\Omega}'} e(H_1(i), \sigma_i)] e(H_2(m), \sigma_0')} = Z.$$

3 Analysis of Li et al.'s outsourced verification protocol

Recently, Li et al. proposed a outsourced verification protocol by using Wu et al.'s technique [8] for their ABS scheme, which provides a solution that verifier can verify the signature with the help of a outside server. This protocol is consisted with three algorithms: the transformation algorithm for outsourced verification **Transf**, the out sourced verify algorithm **Verify-out**, and the verify algorithm **Verify**, replaces the original verifying algorithm in the ABS scheme.

- **Transf**: Once received the signature $\sigma = (\sigma_0, \{\sigma_i\}_{i \in \Omega^* \cup \hat{\Omega}'}, \sigma_0')$, the verifier picks a random $t \in_R \mathbb{Z}_p$ and computes $\tilde{\sigma}_0 = g^t \cdot \sigma_0$. Then, it sends the transformed signature $\sigma_{trans} = (\tilde{\sigma}_0, \{\tilde{\sigma}_i\}_{i \in \Omega^* \cup \hat{\Omega}'}, \tilde{\sigma}_0')$ to the outside server, where $\tilde{\sigma}_i = \sigma_i$ and $\tilde{\sigma}_0' = \sigma_0'$.

- **Verify-out**: When the server received the σ_{trans} on the message m with predicate Υ_{k,Ω^*}, it computes and returns

$$\Lambda = \frac{e(g, \tilde{\sigma}_0)}{[\prod_{i\in\Omega^*\cup\hat{\Omega}'} e(H_1(i), \tilde{\sigma}_i)]e(H_2(m), \tilde{\sigma}'_0)}.$$

- **Verify**: The verifier checks whether the equation $\Lambda = e(g, g)^t \cdot Z$ holds. If it holds, output 1 which indicates the signature is indeed from some user with k attributes among Ω^*. Otherwise, output 0.

This outsourced verification protocol reduces the computation load at verifier side through delivering computation to the outside server. However, if the outside server collude with an attacker, the server can utilize **Verify-out** to convince the verifier that an invalid signature is valid with a negligible probability. The following show the procedure of collusion attack:

1. The attacker $\mathcal{A}$ first sends an invalid signature $\sigma^* = (\sigma_0^*, \{\sigma_i^*\}_{i\in\Omega^{**}\cup\hat{\Omega}'}, {\sigma_0}'*)$ to the verifier, where $\Upsilon_{k,\Omega^{**}}(\Omega) \neq 1$, and Ω is the attributes set from which the private key is generated.
2. The verifier picks a random $t \in_R \mathbb{Z}_p$ and computes $\tilde{\sigma}_0 = g^t \cdot \sigma_0^*$. Then, it sends the transformed signature to the server $\mathcal{S}$.
3. $\mathcal{A}$ and $\mathcal{S}$ are colluded, and $\mathcal{A}$ sends the original invalid signature $\sigma^* = (\sigma_0^*, \{\sigma_i^*\}_{i\in\Omega^{**}\cup\hat{\Omega}'}, {\sigma_0}'*)$ to $\mathcal{S}$.
4. $\mathcal{S}$ gets g^t from $g^t = \frac{\tilde{\sigma}_0}{\sigma_0^*}$, and computes the correct Λ from $\Lambda = e(g, g^t) \cdot Z$. Finally, he sends Λ to the verifier.
5. The verifier checks $\Lambda = e(g, g^t) \cdot Z$, and it always holds. Thus, $\mathcal{A}$ and $\mathcal{S}$ always win the game.

4 Security definition of ABSAVS

4.1 Syntax of ABSAVS

An attribute-based server-aided verification signature ABSAVS consists of t-wo parts: a normal ABS scheme and a attribute-based server-aided verification protocol **ABSA-Verify**. The **ABSA-Verify** protocol is an interactive proto-col between server and verifier, who only has a limited computational abili-ty and is not able to perform all computations in signature verification alone. The **ABSA-Verify** protocol consists of four algorithms: **ParamGen**, **Initially compute**,**Server-aided verify**, **Lightweight-verify**, which can be defined as follows:

ParamGen This algorithm outputs the secret parameters for the receiver.

Initially compute This algorithm takes as input the signature σ, and computes the transformed signature $\hat{\sigma}$.

Server-aided verify The server-aided verify algorithm takes as input - the transformed signature $\hat{\sigma}$ and the corresponding message m and the predicate Υ. It outputs Λ which is used by verifier to perform the lightweight verification.

Lightweight-verify The lightweight verification algorithm takes as input the server-aided verification information Λ. It outputs 1 if the original signature is deemed valid and 0 otherwise.

Obviously, Li et al.'s outsourced verification protocol [7] is exactly a **ABSA-Verify** protocol.

Completeness of ABSA-Verify protocol. An honest server can correctly convince the verifier about the validness (or invalidness) of an attribute-based signature. That is,

$$\textbf{ABSA-Verify}(Server, Verifier) = \textbf{Verify}(\cdot).$$

4.2 Security definition of ABSAVS

In this section, we will define the security of attribute-based server-aided verification signature (ABSAVS) against the collusion between the server and the attacker who impersonates a valid signer. If we allow the server and the attacker to collude, the server will have original signatures (valid or invalid) of any messages. Thus, it is impossible to give a unified security definition to capture both existentially unforgeable ABS and soundness-**ABSA-Verify** simultaneously. With this in mind, we now define the security of **ABSA-Verify** protocol against collusion and adaptive chosen predicate attacks.

Setup. The challenger $\mathcal{C}$ runs the algorithm **Setup** in ABS scheme to obtain the public parameters $params$ and the master secret key sk. The attacker is given $params$.

Queries. The attacker $\mathcal{A}$ can perform a polynomially bounded number of queries on attribute set Ω, and $\mathcal{C}$ answers the corresponding private key sk_{Ω} to $\mathcal{A}$ with the master secret key sk. Since $\mathcal{A}$ has sk_{Ω}, he doesn't need to make the signing queries. $\mathcal{A}$ only needs to make the **Attributed-Based Server-Aided Verification Queries** for a polynomially bounded times. For each query, the challenger $\mathcal{C}$ responds by executing **ABSA-Verify** protocol with $\mathcal{A}$, where $\mathcal{A}$ acts as **Server** and $\mathcal{C}$ acts as **Verifier**. At the end of each executing, the challenger returns the output of **ABSA-Verify** protocol to $\mathcal{A}$.

Output. The attacker $\mathcal{A}$ finally outputs a signature (m^*, σ^*) with a predicate Υ^*, where no attribute set Ω^* such that there exist $\Omega \subset \Omega^*$ satisfying $\Upsilon^*(\Omega) = 1$ has been submitted to the private key extraction queries. σ^* is considered to be a randomly invalid signature with respect to Υ^*. We say $\mathcal{A}$ wins the game if **ABSA-Verify**$(\mathcal{A}, \mathcal{C}) = Valid$.

We define **ABSA-Verify**$-Adv_{\mathcal{A}}$ to be the probability that $\mathcal{A}$ wins the above game, taken over the coin tosses made by $\mathcal{A}$ and challenger.

Definition 1. *An attacker $\mathcal{A}$ is said to (t, q_v, ε)-break the soundness of* ***ABSA-Verify*** *in a* ***ABSAVS*** *if $\mathcal{A}$ runs in time at most t, make at most q_v attributed-based server-aided verification queries and* ***ABSA-Verify*** $-$ *$Adv_{\mathcal{A}}$ is at least*

*ε. The **ABSA-Verify** in a **ABSAVS** is (t, q_v, ε)-sound against collusion and adaptive chosen predicate attacks if there no attacker that (t, q_v, ε) -breaks it.*

5 ABSAVS scheme based on Li et al.'s ABS scheme

In this section, we construct a ABSAVS scheme based on Li et al.'s ABS scheme [2], which is secure against collusion and adaptive chosen predicate attacks. The proposed ABSAVS scheme consists of Li et al.'s ABS scheme and a **ABSA-Verify** protocol. We define the **ABSA-Verify** protocol as follows:

ParamGen The verifier randomly chooses $a \in \mathbb{Z}_p^*$, where $p = |\mathbb{G}_1|$. Let $|\Omega^* \cup \hat{\Omega}'| = n$, the verifier randomly selects a $n-1$-degree polynomial $f(x)$ and $f(0) = a$. Then, verifier randomly chooses $r_1, \cdots, r_n \in \mathbb{Z}_p$. The verifier keeps these parameters secretly.

Initially compute when the verifier received the signature $\sigma = (\sigma_0, \{\sigma_i\}_{i \in \Omega^* \cup \hat{\Omega}'}, \sigma_0')$ of message m, he chooses a special element $\theta \in \Omega^* \cup \hat{\Omega}'$, and computes

$$\hat{\sigma}_0 = [\prod_{i \in \Omega^* \cup \hat{\Omega}', i \neq \theta} (g_2^{f(i)} \cdot H_1(i)^{r_i})^{\Delta_{i,\Omega^* \cup \hat{\Omega}'}(0)}] \cdot \sigma_0$$

$$\hat{\sigma}_i = (g^{r_i})^{\Delta_{i,\Omega^* \cup \hat{\Omega}'}(0)} \cdot \sigma_i, i \in \Omega^* \cup \hat{\Omega}'.$$

Finally, the verifier sends the transformed signature $\hat{\sigma} = (\hat{\sigma}_0, \{\hat{\sigma}_i\}_{i \in \Omega^* \cup \hat{\Omega}'}, \sigma_0')$ with message m to the server.

Server-aided verify The server computes

$$\frac{e(g, \hat{\sigma}_0)}{[\prod_{i \in \Omega^* \cup \hat{\Omega}'} e(\hat{\sigma}_i, H_1(i))] \cdot e(\sigma_0', H_2(m))} = \Lambda$$

, and returns it to the verifier.

Lightweight-verify The verifier computes $V = e(g, g_2)^a$ and

$$W = e((g_2^{f(\theta)} H_1(\theta)^{r_\theta})^{\Delta_{\theta,\Omega^* \cup \hat{\Omega}'}(0)}, g),$$

and checks whether $\Lambda \cdot W = V \cdot Z$ holds, where $Z = e(g_1, g_2)$ is defined in Li et al.'s ABS scheme. If holds, then the signature is valid, and invalid, otherwise.

Efficiency Analysis The proposed **ABSA-Verify** protocol reduces the computation load at verifier side through delivering computation to server but only two pairings locally. Moveover, V and W can be pre-computed offline by the verifier. In the original Li et al.'s ABS scheme, it needs $|\Omega^* \cup \hat{\Omega}'| + 2$ pairings in verification. Compared with the original scheme. the verifier's computational overhead is greatly decreased in our scheme.

5.1 Security Analysis

According to the Lagrange Polynomial $\prod_{i\in\Omega^*\cup\hat{\Omega}'}(g_2^{f(i)})^{\Delta_{i,\Omega^*\cup\hat{\Omega}'}(0)} = g_2^{f(0)} = g_2^a$, the correctness of verification is justified by the following equation:

$$\begin{aligned}\Lambda\cdot W &= \frac{e(g,\hat{\sigma}_0)}{[\prod_{i\in\Omega^*\cup\hat{\Omega}'} e(\hat{\sigma}_i,H_1(i))]\cdot e(\sigma_0',H_2(m))}\cdot e((g_2^{f(\theta)}H_1(\theta)^{r_\theta})^{\Delta_{\theta,\Omega^*\cup\hat{\Omega}'}(0)},g)\\ &= \frac{e(g,g_2^a)\cdot e(g,\sigma_0)}{[\prod_{i\in\Omega^*\cup\hat{\Omega}'} e(\sigma_i,H_1(i))]\cdot e(\sigma_0',H_2(m))}\\ &= e(g,g_2)^a\cdot Z\end{aligned}$$

If the server is untrustworthy, it want to utilize **Server-aided verify** to convince the verifier that an invalid signature is valid in a negligible probability. However, it cannot be successful in our protocol, since the verifier will choose a random a at first, and hide it in the signature by Lagrange Polynomial. That is, $\hat{\sigma}_0 = [\prod_{i\in\Omega^*\cup\hat{\Omega}', i\neq\theta}(g_2^{f(i)}\cdot H_1(i)^{r_i})^{\Delta_{i,\Omega^*\cup\hat{\Omega}'}(0)}]\cdot\sigma_0$ and $\hat{\sigma}_i = (g^{r_i})^{\Delta_{i,\Omega^*\cup\hat{\Omega}'}(0)}\cdot\sigma_i, i\in\Omega^*\cup\hat{\Omega}'$. Since the server has no knowledge of $r_1,\cdots,r_n\in Z_p$, it cannot get a or $e(g,g_2)^a$ from $(\hat{\sigma}_0,\{\hat{\sigma}_i\}_{i\in\Omega^*\cup\hat{\Omega}'},\sigma_0')$. Thus, the server cannot make an invalid signature to be valid.

If the server can collude with the attacker, who impersonates a signer, it can not only receive the transformed signature $\hat{\sigma} = (\hat{\sigma}_0,\{\hat{\sigma}_i\}_{i\in\Omega^*\cup\hat{\Omega}'},\sigma_0')$, but also obtain the original signature $\sigma = (\sigma_0,\{\sigma_i\}_{i\in\Omega^*\cup\hat{\Omega}'},\sigma_0')$ from the attacker. Thus, the server can get $\mu = \prod_{i\in\Omega^*\cup\hat{\Omega}', i\neq\theta}(g_2^{f(i)}\cdot H_1(i)^{r_i})^{\Delta_{i,\Omega^*\cup\hat{\Omega}'}(0)}$ and $\nu_i = (g^{r_i})^{\Delta_{i,\Omega^*\cup\hat{\Omega}'}(0)}, i\in\Omega^*\cup\hat{\Omega}'$. However, the server still cannot re-construct $e(g,g_2)^a$ from μ and ν_i by using Lagrange Polynomial, since it is lack of

$$(g_2^{f(\theta)}H_1(\theta)^{r_\theta})^{\Delta_{\theta,\Omega^*\cup\hat{\Omega}'}(0)}$$

in μ. Thus, the server also cannot make an invalid signature be valid.

6 Security Proof

In this section, we provide a security proof to our ABSAVS scheme in the random oracle by using Li et al.'s technique [7].

Theorem 1. *The proposed attribute-based server-aided verification signature (ABSAVS) scheme based on Li et al.'s ABS scheme is secure in the random oracle if the CDH assumption holds in* $\mathbb{G}_1$.

Proof. Assume that an attacker $\mathcal{A}$ has a non-negligible probability in breaking our ABSAVS scheme in sense of collusion and selective predicate, we attempt to build a challenger $\mathcal{C}$ that utilize $\mathcal{A}$ as sub-algorithm to solve the CDH problem with a non-negligible probability.

Suppose the challenger $\mathcal{C}$ is given an instance of CDH problem g, $X = g^x$ and $Y = g^y$ where $x,y\in_R\mathbb{Z}_p$ and asked to compute g^{xy}. We proceed the simulation as follows:

Init. $\mathcal{C}$ runs $\mathcal{A}$, and receives a challenge predicate Υ_{k,Ω^*}.

Setup. Let the dummy attribute denote as $\hat{\Omega}$. $\mathcal{C}$ selects a subset $\hat{\Omega}' \subseteq \hat{\Omega}$ with $|\hat{\Omega}'| = d - k$. $\mathcal{C}$ randomly chooses $x_1 \in_R \mathbb{Z}_p$ and sends $g_1 = X/(g^{x_1})$ and $g_2 = Y$ to $\mathcal{A}$.

Queries. $\mathcal{C}$ initialize an integer $j = 0$, two empty table L_1 and L_2, and an empty set U. $\mathcal{A}$ is allowed to issue queries as follows.

H_1 **query:** The attacker $\mathcal{A}$ can make at most q_{H1} queries to hash function H_1. $\mathcal{C}$ maintains a list L_1 to store the answers to hash oracle H_1. Then, upon receiving the query i, $\mathcal{C}$ checks L_1, and if an entry for i is exist, then the same answer will be returned. Otherwise, $\mathcal{C}$ computes:

$$H_1(i) = \begin{cases} g^{\beta_i} & i \in \Omega^* \cup \hat{\Omega}' \\ g_1^{\alpha_i} g^{\beta_i} & i \notin \Omega^* \cup \hat{\Omega}' \end{cases}$$

where $\alpha_i, \beta_i \in_R \mathbb{Z}_p$, and answers with $H_1(i)$. Then, $\mathcal{C}$ adds $(i, H_1(i))$ to L_1.

H_2 **query:** Suppose $\mathcal{A}$ can make at most q_{H2} queries to hash oracle H_2, and $\mathcal{C}$ maintains a list L_2 to store the answers. Then, upon receiving the j-th time H_2-query on message m_j for $1 \leq j \leq q_{H2}$. $\mathcal{C}$ checks on the list L_2. If an entry for the query is exist, then the same answer will be returned. Otherwise, $\mathcal{C}$ returns $H_2(m_j) = g^{\beta'_j}$, where $\beta'_j \in_R \mathbb{Z}_p$. Then, $\mathcal{C}$ adds $(m_j, H_2(m_j))$ to L_2.

Private key query We suppose that $\mathcal{A}$ can make at most q_k private key extract queries. When receiving an private key request on attribute set Ω, if $\Upsilon_{k,\Omega^*}(\Omega) = 1$, then $\mathcal{C}$ outputs failure, otherwise, $\mathcal{C}$ sets $j + 1$ and attempts to perform simulation as follows. $\mathcal{C}$ defines two sets with $\Gamma = (\Omega \cap \Omega^*) \cup \hat{\Omega}'$, $\Gamma \subseteq \Gamma' \subseteq \Omega \cup \hat{\Omega}'$ and $|\Gamma'| = d - 1$. Then, for any $i \in \Gamma'$, $(d_{i0}, d_{i1}) = (g_2^{\gamma_i} H_1(i)^{r_i}, g^{r_i})$, where $\gamma_i, r_i \in_R \mathbb{Z}_p$. For $i \in \Omega \cup \hat{\Omega}\ \Gamma'$, let $r_i = -\frac{y\Delta_{i,\Gamma' \cup 0}}{\alpha_i} + r'_i$ where $r_i \in_R \mathbb{Z}_p$ and simulate $(d_{i0}, d_{i1}) = (g_2^{\sum_{j\in\Gamma'} \gamma_j \Delta_{j,\Gamma'\cup 0}(i) - \frac{\beta_i}{\alpha_i}\Delta_{0,\Gamma'\cup 0}(i)} g_1^{\alpha_i r'_i} g^{\beta_i r'_i}, g_2^{-\frac{\Delta_{i,\Gamma'\cup 0}(i)}{\alpha_i}} g^{r'_i})$. The Lagrange Polynomial $q(i) = \sum_{j\in\Gamma'} q(j)\Delta_{j,\Gamma'\cup 0}(i) + q(0)\Delta_{0,\Gamma'\cup 0}(i)$ is behind the above assignments. The challenger $\mathcal{C}$ is implicitly selecting a random $d-1$-degree polynomial $q(x)$ by choosing its values for the $d - 1$ points as $q(i) = \gamma_i$, and $q(0) = x - x_1$. Finally, after updating the entry $j, \Omega, \cdot, PKey$ in L where the private key $PKey = \{d_{i0}, d_{i1}\}_{i\in\Omega\cup\hat{\Omega}}$, $\mathcal{C}$ returns $PKey$ to $\mathcal{A}$.

Server-aided verification query The attacker only needs to make q_v server-aided verification queries adaptively. For each queries (m, σ), the challenger $\mathcal{C}$ responds by executing **ABSA-Verify** protocol with the attacker $\mathcal{A}$, where $\mathcal{A}$ acts as $Server$ and $\mathcal{C}$ acts as $Verifier$. At the end of each execution, the challenger returns the output of **ABSA-Verify** protocol to $\mathcal{A}$.

Output. $\mathcal{A}$ outputs a forged signature σ^* on message m^* with Υ_{k,ω^*}, where no attribute set ω^* such that there exists $\omega \subseteq \omega^*$ satisfying $\Upsilon^*(\omega) = 1$ has been submitted to the private key extract oracle. If $H_2(m^*) \neq g^{\beta'_\delta}$, where

$\delta \in \{1, 2, \cdots, q_{H2}\}$, then $\mathcal{C}$ will aborts. Otherwise, if this forged signature can be verified by the **ABSA-Verify** protocol, then it can be simulated as follows.

1. In the first place, $\mathcal{A}$ submits the forged signature $\sigma^* = (\sigma_0^*, \{\sigma_i^*\}_{i\in\omega\cup\hat{\omega}'}, \sigma_0'^*)$ to $\mathcal{C}$.
2. Secondly, $\mathcal{C}$ who acts as $Verifier$ chooses a $n-1$-degree polynomial $f(x)$ and $f(0) = x_1$ and a special element $\theta \in \omega^* \cup \hat{\omega}'$. $\mathcal{C}$ computes the transformed signature $\hat{\sigma}^* = (\hat{\sigma}_0^*, \{\hat{\sigma}_i^*\}_{i\in\omega\cup\hat{\omega}'}, \sigma_0'^*)$, and returns it to $\mathcal{A}$ who acts as $Server$.
3. Finally, $\mathcal{A}$ sends Λ^* to $\mathcal{C}$. If Λ^* is valid, it means that

$$\begin{aligned}
&\Lambda^* \cdot e((g_2^{f(\theta)} H_1(\theta)^{r_\theta})^{\Delta_{\theta,\omega^*\cup\hat{\omega}'}(0)}, g)\\
&= e(g, g_2)^{x_1} \cdot e(g_1, g_2)\\
&= e(g^{x_1}, g_2) \cdot e(X/g^{x_1}, g_2)\\
&= e(X, Y) = e(g, g^{xy}).
\end{aligned}$$

That is,

$$\begin{aligned}
&\Lambda^* \cdot e((g_2^{f(\theta)} H_1(\theta)^{r_\theta})^{\Delta_{\theta,\omega^*\cup\hat{\omega}'}(0)}, g)\\
&= \frac{e(g, \hat{\sigma}_0^*)}{[\prod_{i\in\omega^*\cup\hat{\omega}'} e(\hat{\sigma}_i^*, H_1(i))] \cdot e(\sigma_0'^*, H_2(m))} \cdot e((g_2^{f(\theta)} H_1(\theta)^{r_\theta})^{\Delta_{\theta,\omega^*\cup\hat{\omega}'}(0)}, g)\\
&= \frac{e(g, [\prod_{i\in\omega^*\cup\hat{\omega}', i\neq\theta}(g_2^{f(i)} \cdot H_1(i)^{r_i})^{\Delta_{i,\omega^*\cup\hat{\omega}'}(0)}] \cdot \sigma_0^*)}{[\prod_{i\in\omega^*\cup\hat{\omega}'} e((g^{r_i})^{\Delta_{i,\omega^*\cup\hat{\omega}'}(0)} \cdot \sigma_i^*, H_1(i))] \cdot e(\sigma_0'^*, H_2(m))} \cdot e((g_2^{f(\theta)} H_1(\theta)^{r_\theta})^{\Delta_{\theta,\omega^*\cup\hat{\omega}'}(0)}, g)\\
&= \frac{e(g, [\prod_{i\in\omega^*\cup\hat{\omega}', i\neq\theta}(g_2^{f(i)} \cdot g^{\beta_i r_i})^{\Delta_{i,\omega^*\cup\hat{\omega}'}(0)}] \cdot \sigma_0^*)}{[\prod_{i\in\omega^*\cup\hat{\omega}'} e((g^{r_i})^{\Delta_{i,\omega^*\cup\hat{\omega}'}(0)} \cdot \sigma_i^*, g^{\beta_i})] \cdot e(\sigma_0'^*, g^{\beta'_\delta})} \cdot e((g_2^{f(\theta)} g^{\beta_\theta r_\theta})^{\Delta_{\theta,\omega^*\cup\hat{\omega}'}(0)}, g)\\
&= e(X, Y) = e(g, g^{xy})
\end{aligned}$$

Then, $\mathcal{C}$ can compute

$$g^{xy} = \frac{[\prod_{i\in\omega^*\cup\hat{\omega}', i\neq\theta}(g_2^{f(i)} \cdot g^{\beta_i r_i})^{\Delta_{i,\omega^*\cup\hat{\omega}'}(0)}] \cdot \sigma_0^* \cdot (g_2^{f(\theta)} g^{\beta_\theta r_\theta})^{\Delta_{\theta,\omega^*\cup\hat{\omega}'}(0)}}{[\prod_{i\in\omega^*\cup\hat{\omega}'}((g^{r_i})^{\Delta_{i,\omega^*\cup\hat{\omega}'}(0)} \cdot \sigma_i^*)^{\beta_i}](\sigma_0'^*)^{\beta'_\delta}}.$$

7 Conclusions

In this paper, we propose a new cryptographic notion of attribute-based server-aided verification signature (ABSAVS), in which the verifier can verify an attribute based signature with the assistance of an external server. This system may be very suitable for the resource limited devices. We analysis Li et al.'s outsourced verification protocol, and show that it can be forged by collusion between untrusted server and outside attacker. We propose a formal security definition of ABSAVS, and design a concrete ABSAVS scheme from Li et al.'s ABS scheme. Our scheme can resist the collusion attack, and we give a security proof in the random oracle. However, in addition to two pairings, it still needs one multi-exponentiation and one exponentiation for the verifier in our ABSAVS scheme, and how to further reduce the computational cost in the verifier's side is our future work.

Acknowledgments. This research is supported by the National Natural Science Foundation of China under Grant No.61272422, 61202353, the Natural science fund for colleges and universities in Jiangsu Province No. 11KJB520015, and the Natural Science Foundation of Nanjing University of Posts and Telecommunications No. NY212089.

References

1. Shahandashti, S., Safavi-Naini, R.: Threshold attribute-based signatures and their application to anonymous credential systems. In: Preneel, B. (ed.) Progress in Cryptology - AFRICACRYPT 2009, Lecture Notes in Computer Science, vol. 5580, pp. 198-216. (2009)
2. Li, J., Au, M.H., Susilo, W., Xie, D., Ren, K.: Attribute-based signature and its applications. In: Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security. pp. 60-69. ASIACCS'10, ACM, New York, NY, USA (2010)
3. Escala, A., Herranz, J., Morillo, P.: Revocable attribute-based signatures with adaptive security in the standard model. In: Nitaj, A., Pointcheval, D. (eds.) Progress in Cryptology - AFRICACRYPT 2011, Lecture Notes in Computer Science, vol. 6737, pp. 224-241. (2011)
4. Lauter, K., Montgomery, P.L., Naehrig, M.: An Analysis of Affine Coordinates for Pairing Computation. In: Joye, M., Miyaji, A., Otsuka, A. (eds.) Pairing 2010. LNCS, vol. 6487, pp. 1C20. Springer, Heidelberg (2010)
5. Herranz, J., Laguillaumie, F., Libert, B., Rfols, C.: Short attribute-based signatures for threshold predicates. In: Dunkelman, O. (ed.) Topics in Cryptology - CT-RSA 2012, Lecture Notes in Computer Science, vol. 7178, pp. 51-67. (2012)
6. Martin Gagné, Shivaramakrishnan Narayan, and Reihaneh Safavi-Naini. Short Pairing-Efficient Threshold-Attribute-Based Signature. M. Abdalla and T. Lange (Eds.): Pairing 2012, LNCS 7708, pp. 295C313, 2013.
7. Jin Li, Xiaofeng Chen, Jingwei Li, Chunfu Jia, Duncan S. Wong, Willy Susilo. Secure Outsourced Attribute-Based Signatures. Cryptology ePrint Archive: Report 2012/605, http://eprint.iacr.org/2012/605, 2012.
8. Wei Wu, Yi Mu, Willy Susilo, Xinyi Huang. Provably secure server-aided verification signatures. Computers and Mathematics with Applications, Volume 61, Issue 7, April 2011, Pages 1705-1723
9. Maji, H., Prabhakaran, M., Rosulek, M.: Attribute-based signatures: Achieving attribute-privacy and collusion-resistance. Cryptology ePrint Archive, Report 2008/328 (2008)
10. J-J. Quisquater, M. De Soete. Speeding up smart card RSA computation with insecure coprocessors, in: Proceedings of Smart Cards 2000, 1989, pp. 191C197.
11. C.H. Lim, P.J. Lee. Security and performance of server-aided RSA computation protocols, Advances in Cryptology CRYPTO95, Lecture Notes in Computer Science, vol. 963, Springer-Verlag (1995), pp. 70C83.

12. M. Girault, J.J. Quisquater. GQ+GPS = new ideas + new protocols. Eurocrypt02Rump Session, 2002.
13. M. Girault, D. Lefranc. Server-aided verification: theory and practice, ASIACRYPT05, Lecture Notes in Computer Science, vol. 3788, Springer-Verlag (2005), pp. 605C623.