2014 IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications

Efficient Traceable Attribute-Based Signature

Shenglong Ding
Software School, Fudan University
Shanghai, China
dingshlong@gmail.com

Yiming Zhao
Software School, Fudan University
Shanghai, China
zhym@fudan.edu.cn

Yuyang Liu
Software School, Fudan University
Shanghai, China
liuyuyang13@fudan.edu.cn

***Abstract*—In an Attribute-Based Signature (ABS) scheme, only users whose attributes satisfy the signing predicate can create valid signatures. Typically, ABS schemes require two properties: unforgeability and privacy. In this paper, we propose a new structure and syntax for Traceable Attribute-Based Signature (TABS), which has a good balance between privacy and traceability. In a TABS scheme, the privacy of signers is also protected, each single user cannot link a signature to the identity (the set of attributes) of the signer. Meanwhile, given a malicious signature, the two trusted parties in our system can work together to trace the identity of the signer. Besides, we give a construction of a TABS scheme, and prove its existential unforgeability in the selective predicate security model under the q-Augmented Diffie-Hellman Exponent assumption. Compared to other ABS schemes, our scheme generates shorter signatures and is much more efficient.**

***Keywords*—Attribute-Based Signature, Traceability, Privacy, Existential Unforgeability, Selective Predicate Model, Access Structure**

I. Introduction

Attribute-Based Encryption (ABE), first proposed by Sahai and Waters [1] as a generalization of Identity Based Encryption (IBE), is considered to have enormous potential for providing data security in distributed environments. Because of the lack of expressiveness of the primitive in [1], Goyal et al. [2] further proposed two complementary forms of ABE. In Key Policy ABE, attributes are used to annotate the ciphertexts and access structures over these attributes are ascribed to users' private keys. While in Ciphertext Policy ABE, attributes are used to describe the users' identities, and the ciphertexts are associated with access structures. In recent years, extensive research [3], [4], [5] have been conducted on attribute-based encryption.

With the development of ABE, Attribute-Based Signature (ABS) is also widely studied. The formal definition of ABS was first introduced by Maji et al. [6]. In ABS, each user has a set of attributes and receives his private key from the attribute issuing authority. To generate a signature, the signee provides the message to be signed as well as a signing predicate, and only those signers whose attributes satisfy the predicate can generate a valid signature. Since then, several ABS schemes have been proposed [7], [8], [9], [10], [11], mainly to provide schemes secure in the standard model or supporting general access structures.

Traceability has always been one of the most important properties for signature schemes. In traditional certificate-based or identity-based signature schemes, the signature contains the information of the signer, either the public certificate or the identity. However, for attribute-based signature schemes, *privacy* is the key property for its wide use in distributed and cloud environments. That is, "a valid ABS signature only attests to the fact that a single user, whose attributes satisfy the predicate, endorsed the signature" [6], the signature cannot be linked to the set of attributes (and corresponding private keys) the signer used to generate it. And for some ABS schemes [11], [6], they further satisfy *perfect privacy*, i.e. given the same message, the signatures generated by different signers, whose attributes all satisfy the predicate, are indistinguishable. This property makes ABS improper in some applications, especapilly those need to protect the privacy of users and support traceability at the same time. For example, a large datahouse uses ABS for anonymous authentication to protect the privacy of its users, but there should also be a method to detect the users who leak confidential data.

A. Related Work

To our best knowledge, Escala et al. [10] was the first to propose a similar cryptographic primitive named *Revokable Attribute-Based Signature* in which the *judge* can "*revoke* the anonymity of an attribute-based signature". However, (1) non-interactive witness indistinguishable (NIWI) proofs [12], [13] are used in this scheme in order to make it non-linkable and support general signing policies, thus the scheme is not quite efficient; (2) The special user *judge* stores a secret table, with which anyone can easily break the anonymity of any signature, this setting is not very reasonable in a distributed or cloud environment, as once the judge is broken through or the judge himself leaks the secrets for private benefits, the whole system won't be safe.

More recently, to overcome the second drawback, El Kaafarani et al. [14] proposed a new security model and two generic constructions for decentralized traceable attribute-based signatures (DTABS). DTABS protects against a fully corrupt tracing authority, and also enhances security by avoiding a possible attacking senario of [10]. However, DTABS also uses non-interactive zero-knowledge (NIZK) proofs [12], [13] and therefore is not quite efficient as well.

B. Our Contribution

In this paper, we propose the concept of Traceable Attribute-Based Signature (TABS) as the name of this primitive. We give the formal syntax and definition of TABS. And in our TABS structure, there are four basic roles: attribute issuing authority, private key generator (PKG), signer and verifier. This structure is similar to Maji et al.'s [6] structure, but the two trusted third parties play somewhat different roles from theirs. Then *Traceability* is defined as given a valid signature, the attribute issuing authority and PKG can work together to trace the signer. It is worth noticing that our TABS schemes also satisfy the *privacy* property, that is, each *single* user, including the issuing authority and PKG, cannot link a signature to the identity (the set of attributes) of the signer.

Next, we give the construction of an efficient TABS scheme without NIZK or NIWI proofs based on the recent work of [11], [6], [5]. In this construction, we use the general access structure LSSS (Linear Secret Sharing Scheme [15]), instead of the Z_n-monotone span programs in [10] as the signing predicate and the scheme can be proved existentially unforgeable in the selective predicate security model under the q-Augmented Diffie-Hellman Exponent (q-ADHE) assumption in the random oracle. q-ADHE problem satisfies the general form[16] of hard problems in the bilinear group and can be easily shown to be hard. We also give the method to modify our scheme to be secure in the standard model. Meanwhile, we prove that our scheme satisfies traceability and privacy.

Besides, our scheme generates short-size signatures (much shorter than the scheme in [6], [10], [14]), supports arbitrary-length messages, and is more efficient than the traceable scheme in [10], [14] and the deniable scheme in [6]. These properties all make our scheme more practical.

C. Organization

The rest of the paper is organized as follows. In section II, we describe some preliminaries, including some background definitions, the syntax and security of TABS. In section III we give the construction of our TABS scheme followed by the security proof and analysis of the scheme in section IV. Finally, we conclude in section V.

II. Preliminaries

We first briefly review the definitions for access structures and relevant background knowledge of Linear Secret Sharing Schemes (LSSS). Then we give the definitions and security model of traceable attribute-based signature. Finally, we give basic information of bilinear maps and introduce the Diffie-Hellman Exponent assumption.

A. Access Structure

Definition 1: (**Access Structure** [15]) Let $\{P_1, P_2, \ldots, P_n\}$ be a set of parties. A collection $\mathbb{A} \subseteq 2^{\{P_1,\ldots,P_n\}}$ is monotone if for all B, C, if $B \in \mathbb{A}$ and $B \subseteq C$ then $C \in \mathbb{A}$. An *access structure* (respectively, *monotone access structure*) is a collection (respectively, monotone collection) $\mathbb{A}$ of non-empty subsets of $\{P_1, P_2, \ldots, P_n\}$. The sets in $\mathbb{A}$ are called the *authorized sets*, and the sets not in $\mathbb{A}$ are called the *unauthorized sets*.

In our context, *parties* should be considered as *attributes*. Thus, the access structure is a collection of sets of attributes. And we will only use monotone access structures in this paper. Besides, in ABS, we will use the term *signing predicate* instead of access structure. Each predicate Γ is an access structure $\mathbb{A}$, given a set S of attributes, we say that S *satisfies* Γ if and only if $S \in \mathbb{A}$, and we denote this as $\Gamma(S) = 1$.

For access structures, there are two typical kinds of implementation, *access tree* and *linear secret sharing scheme* (LSSS). Using the standard techniques in [17], anyone can convert an access tree to an LSSS matrix. In our scheme, we will use LSSS to represent the access structure.

Definition 2: (**LSSS** [15]) A secret-sharing scheme Π over a set of parties $\mathcal{P}$ is called *linear* (over Z_p) if

1) The shares for each party form a vector over Z_p.
2) There exists a matrix $\mathbf{M}$ with ℓ rows and n columns called the share-generating matrix for Π. For all $i = 1, \ldots, \ell$, we let the function ρ define the party labeling row i as $\rho(i)$. When we consider the column vector $\vec{v} = (s, r_2, \ldots, r_n)$, where $s \in Z_p$ is the secret to be shared, and $r_2, \ldots, r_n \in Z_p$ are randomly chosen, then $\mathbf{Mv}$ is the vector of ℓ shares of the secret s according to Π. The share $(\mathbf{Mv})_i$ belongs to party $\rho(i)$.

Every LSSS has the *linear reconstruction* property [15], i.e. suppose Π is an LSSS for the access structure $\mathbb{A}$, let $S \in \mathbb{A}$ be an authorized set, $I \subseteq \{1, 2, \ldots, \ell\}$ be defined as $I = \{i : \rho(i) \in S\}$, then there exist constants $\{w_i \in Z_p\}_{i \in I}$ such that, if $\{\lambda_i\}_{i \in I}$ are valid shares of any secret s according to Π, then $\sum_{i \in I} w_i \lambda_i = s$. And these constants $\{w_i\}_{i \in I}$ can be found in time polynomial in the size of $\mathbf{M}$.

However, for any unauthorized set I of rows, the target vector $(1, 0, \ldots, 0)$ is not in the span of the rows in the set I. Moreover, there exists a vector $\mathbf{w}$ such that $\mathbf{w} \cdot (1, 0, \ldots, 0) = -1$ and $\mathbf{w} \cdot \mathbf{M}_i = 0$ for all $i \in I$.

B. Syntax and Security

To introduce the traceability property, we slightly modify the syntax of ABS schemes in [6]. In a Traceable Attribute-Based Signature scheme, there are basically four roles:

- *Attribute Issuing Authority*. This authority is responsible for authenticating users' attributes from their private information and issuing attributes to users.
- *Private Key Generator*. PKG is responsible for generating private keys for users, given his attributes.
- *Signer*. Signer first receives his attributes from the attribute issuing authority, next he can get his private key from PKG. Using the private key, signers can create signatures.
- *Verifier*. Verifier uses the public parameters to verify signatures generated by the signers.

A TABS scheme is made up of the following 7 algorithms.

1) **ASetup**(λ). The *ASetup* algorithm is run by the attribute issuing authority. It takes a security parameter λ as input and outputs the public parameters and the secret key for the issuing authority. Besides, it also defines the universe of attributes.
2) **Issue**(ID, w). The *Issue* algorithm is run by the issuing authority. It takes as input a user's identity ID and the materials w to prove his attributes , after verifying w it will use the issuing authority's secret key to generate a signature of the attributes.
3) **Setup**(λ, U). The *Setup* algorithm is run by the private key generator. It takes the security parameter λ and the number of attributes U in the system as input,

and returns the public parameters PK and master secret key mk.

4) **Extract**(S, mk). The *Extract* algorithm is run by PKG. It receives the user's set of attributes S (along with the issuing authority's signature of S), and computes the private key SK_S.
Note: when PKG computes private keys for users, it won't know the identity of the user, it only knows that the user has a set S of attributes.
5) **Sign**(m, SK_S, Γ). The *Sign* algorithm is run by the signer who has attributes S. It receives a message m and a signing predicate Γ. When S satisfies Γ, the algorithm will output a signature σ; otherwise, the algorithm will abort.
6) **Verify**(m, Γ, σ, PK). The *Verify* algorithm is run by the verifier. It takes a signature σ of message m under predicate Γ, and will output "valid" or "invalid".
7) **Trace**(m, Γ, σ). The *Trace* algorithm is run by the attribute issuing authority and PKG together. It takes as input the signature σ of message m under predicate Γ, and outputs the signer's identity.

Definition 3: (**Correctness**) A TABS scheme is *correct* if for any message m, any signing predicate Γ, and any signer with attributes S s.t. $\Gamma(S) = 1$, then

$$\text{Verify}(m, \Gamma, Sign(m, SK_S, \Gamma), PK) = \text{"valid"}.$$

Besides correctness, a good TABS scheme should also satisfy *unforgeability*. And the definition of unforgeability is given in the following selective predicate game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$:

- **Init Phase**. The adversary $\mathcal{A}$ declares a challenge predicate Γ^*.
- **Setup Phase**. The challenger $\mathcal{C}$ runs the *Setup* algorithm and publish its public parameters PK.
- **Query Phase**. $\mathcal{A}$ is allowed to adaptively issue queries to 2 oracles:
    1) *Extract* oracle. $\mathcal{A}$ can query for the private key for any set S of attributes under the restriction $\Gamma^*(S) \neq 1$.
    2) *Sign* oracle. $\mathcal{A}$ can query for signature of any message m under any predicate Γ.
- **Forge Phase**. Finally, $\mathcal{A}$ outputs a signature σ^* on a message m^* with the challenge predicate Γ^*, and (m^*, Γ^*) was not queried to the *Sign* oracle.

The advantage of $\mathcal{A}$ in this game if defined as

$$Adv_{\mathcal{A}} = \Pr[Verify(m^*, \Gamma^*, PK, \sigma^*) = \text{"}valid\text{"}].$$

Definition 4: (**Unforgeability**) A TABS scheme is existentially unforgeable in the selective predicate model if all polynomial-time adversaries have at most a negligible advantage in the above game.

Next, based on our TABS structure, we give the definition of traceability.

Definition 5: (**Traceability**) A TABS scheme is traceable if given any valid signature, the issuing authority and PKG can work together to find out the signer's identity.

This definition requires that not only the *Trace* algorithm works, but also nobody, except the issuing authority and PKG, can pretend to be another valid or nonexistent user and create signatures.

Finally, compared to the perfect privacy property, which requires that given a message and a predicate all valid signatures have the same distribution, we give a "weaker" definition for privacy.

Definition 6: (**Privacy**) A TABS scheme satisfies privacy if given any message m, any predicate Γ, any two signers ID$_1$, ID$_2$, whose attributes S_1 and S_2 both satisfies Γ, and a valid signature generated by one of them, any polynomial-time adversary (except that he breaks the two trusted parties at the same time) cannot find out the signer's id.

But notice that the adversary should have no knowledge about signatures created by ID$_1$ or ID$_2$ before.

C. Bilinear Maps

Let G and G_T be two cyclic groups of prime order p, g is a generator of G and $e : G \times G \rightarrow G_T$ is a bilinear map. e has the following properties:

1) Bilinearity: for all $u, v \in G$ and $a, b \in Z_p$, we have $e(u^a, v^b) = e(u, v)^{ab}$.
2) Non-degeneracy: $e(g, g) \neq 1$.

We say that G is a bilinear group if the group operation in G and the bilinear map e are both efficiently computable. We refer the reader to [18] for more details of bilinear groups.

The unforgeability of our TABS scheme is based on the computational q-Augmented Diffie-Hellman Exponent (q-ADHE) assumption, which is modified from the q-BDHE assumption [16].

Definition 7: (q-**ADHE**) Choose $a \in_R Z_p^*$, the q-ADHE assumption is that given $(g, y_1, y_2, \ldots, y_q, y_{q+2}, \ldots, y_{2q}, y_{2q+1})$ where $y_i = g^{a^i}$, no polynomial-time adversary is able to compute y_{q+1} with more than a negligible advantage.

By simple observation, q-ADHE assumption is a little stronger than the q-DHE assumption[11], but it also satisfies the general form[16] of hard problems in the bilinear group, thus we can use it to construct our scheme.

And the traceability of our TABS scheme is based on the computational Inverse Diffie-Hellman (IDH) assumption.

Definition 8: (**IDH**) Choose $a, b \in_R Z_p^*$, the IDH assumption is that given (g, g^a, g^b), no polynomial-time adversary is able to compute $g^{a/b}$ with more than a negligible advantage.

It is easy to prove that the IDH problem is as hard as the CDH assumption.

III. Traceable ABS Scheme

In this section, we provide the construction of our traceable attribute-based signature scheme. We begin by describing the techniques we use in the scheme and then give the detailed scheme.

A. Our Techniques

To construct an traceable attribute-based signature scheme, a big challenge is to remove the *private key blinding* property. In many attribute-based schemes like [5], [6], [11], each user gets his private key from the trusted third (PKG in our system), but given the private key, every user can *fake* a new but valid private key. This is called private key blinding, and is the core of the perfect privacy property of schemes in [6], [11]. Thus, in our construction, the *Extract* algorithm should work in a different way and generate private keys that cannot be blinded.

As for *Trace* algorithm, we propose the new four-role system to enchance system robustness. In a traditional three-role structure (PKG, signer, verifier), PKG (also plays as the traceation judge in [10]) will maintain the mapping between the identity (name, address etc.) of each user and his private key. Thus if PKG is attacked and controlled, the whole system will break up. In our four-role system, as described in section II-B, the issuing authority knows each user's identity but not his private key, while PKG knows private keys but not corresponding identities. And they must work together to trace a malicious signature.

Perfect privacy is obviously contradictory with traceability. Hence we will only achieve privacy, i.e. given a valid signature, only the issuing authority and PKG working together can find out the signer. To achieve this, we will first use random group elements to blind the used attributes in the signatures as in [11], and then cut the mapping from signatures to signers into two parts, mantained by the issuing authority and PKG respectively.

Finally, to improve efficiency and support arbitrary messages, we will use a hash function modeled as a random oracle. This makes our scheme more suitable for practical applications.

B. Construction

According to the definition of traceable attribute-based signature in section II-B, our scheme follows:

- **ASetup**(λ): The algorithm takes as input the security parameter λ, and generates the public parameters and secret key for the issuing authority. Besides, it also publishes the universe of attributes $\mathbb{U}$ in the system.
- **Issue**(ID, w): The algorithm takes as input a user's ID and his materials w as the evidence of the user's attributes S, it verifies w and then creates a signature Sig(S) with the issuing authority's secret key. Besides, the issuing authority also maintains an *attribute signature table* to record the mapping from the user's signature of attributes Sig(S) to his ID.
  Note: The issuing authority can choose *any* secure and efficient signature scheme to setup and generate signatures, e.g. the Boneh-Boyen signature scheme [19] or the Waters scheme [20]. And each user should regard his attributes' signature as a personal secret.
- **Setup**(λ, U): The algorithm takes as input the security parameter λ and the number of attributes $U = |\mathbb{U}|$. It first chooses a bilinear group G with the bilinear map $e : G \times G \to G_T$, both G and G_T are cyclic groups of prime order p, and g is a random generator of G. Next, it chooses U random group elements $h_1, \dots, h_U \in G$ that are associated with the U attributes in the system. In addition, it chooses random exponents $\alpha, a \in Z_p$, sets $Y = e(g,g)^\alpha$, $Z = g^a$, and chooses a collision-resistant hash function $H : \{0,1\}^* \to G$. Then the public parameters are

$$PK = \{G, G_T, e, p, g, Y, Z, h_1, \dots, h_U, H\}.$$

The master secret key is

$$mk = \{g^\alpha\}.$$

- **Extract**(S, Sig(S), mk): The algorithm takes as input the master secret key mk and a set S of attributes along with the issuing authority's signature Sig(S). After verifying the signature, this algorithm first chooses $t \in_R Z_p^*$ and computes $K = g^\alpha g^{a(t+t^2)}$, $L = g^t$, $T = g^{at^2}$, then for each attribute $x \in S$, it computes $K_x = h_x^t$. Thus, the private key for S is

$$SK_S = \{K, L, T, \{K_x\}_{x \in S}\}.$$

Besides, PKG maintains a *private key table* to record the mapping from $e(g, L)$ to the user's signature of attributes Sig(S).
Note: In our *Extract* algorithm, owing to $T = g^{at^2}$, it is hard to fake a new secret key. Refer to section IV-C for more details.

- **Sign**(m, SK_S, Γ): The algorithm takes as input the message $m \in \{0,1\}^*$ to be signed, the private key for the attribute set S and the signing predicate Γ expressed as an LSSS access structure $(\mathbf{M}, \rho)$, where $\mathbf{M}$ is an $\ell \times k$ matrix, and ρ is an *injective* function that maps rows of $\mathbf{M}$ to attributes.
  The algorithm first checks if S satisfies Γ. If $\Gamma(S) \neq 1$, signer S cannot generate the signature and aborts. Otherwise, the algorithm finds a vector $\vec{\alpha} = (\alpha_1, \dots, \alpha_\ell)$ such that $\vec{\alpha}\mathbf{M} = (1, 0, \dots, 0)$. Besides, it also chooses a random vector $\vec{\beta} = (\beta_1, \dots, \beta_\ell)$ such that $\vec{\beta}\mathbf{M} = (0, 0, \dots, 0)$. Next, for each $i \in [1, \ell]$, it computes $s_i = L^{\alpha_i} g^{\beta_i}$ and sets $y = \prod_{i=1}^{\ell} (K_{\rho(i)}^{\alpha_i} h_{\rho(i)}^{\beta_i})$. Then, it chooses two random exponents $r_1, r_2 \in Z_p$ and computes

$$A = yKH(m)^{r_1} g^{r_2},\ B = g^{r_1},\ C = Tg^{r_2}.$$

Finally, the signature is $\sigma = (s_1, \dots, s_\ell, A, B, C)$ along with the description of (m, Γ).
Note: the algorithm may not have $K_{\rho(i)}$ for each $i \in [1, \ell]$ to compute y, but when it computes $\vec{\alpha}$, it should ensure that if $\alpha_i \neq 0$, then $\rho(i) \in S$, in this way y can be computed easily. The random vector β here is to blind the used attributes. Refer to section IV-D for more details. The random factor g^{r_2} is used to blind C, and for signatures generated by different signers, C has the same distribution.

- **Verify**(m, Γ, σ, PK): The algorithm takes as input the public parameters PK and the signature $\sigma = (s_1, \dots, s_\ell, A, B, C)$ of a message $m \in \{0,1\}^*$ under the signing predicate $\Gamma = (\mathbf{M}_{\ell \times k}, \rho)$. First, the algorithm chooses a random vector $\vec{v} = (v_1 = 1, v_2, \ldots, v_k)$, and for each $i \in [1, \ell]$, it computes

$$\lambda_i = \sum_{j=1}^{k} (v_j \mathbf{M}_{i,j}).$$

Then, it checks if the following equation holds:

$$Ye(g, C)e(H(m), B) \prod_{i=1}^{\ell} e(Z^{\lambda_i} h_{\rho(i)}, s_i) =?e(g, A).$$

Only if the equation holds, this algorithm outputs "valid".

- **Trace**(m, Γ, σ): The algorithm takes as input a signature σ of a message m under the predicate Γ. Like the *Verify* algorithm, this algorithm chooses $\vec{v} = (1, v_2, \ldots, v_k)$ and computes λ_i similarly, then it computes

$$\prod_{i=1}^{\ell} e(g^{\lambda_i}, s_i) = e(g, L).$$

Then, PKG looks through his private key table and gets the signature of the signer's attributes, next the issuing authority will look through the attribute signature table and return the signer's ID.

IV. Analysis

In this section, we first show the correctness of our signature scheme. Next, we give formal security proof of the existential unforgeability, traceability and privacy. Then, we give a method to modify our scheme to be secure in the standard model. Finally, we analyze the efficency of our scheme and compare with some other ABS schemes.

A. Correctness

According to the definition of correctness, an attribute-based signature scheme is correct if valid signatures can pass the *Verify* algorithm.

Theorem 1: The traceable attribute-based signature scheme in section III is correct.

Proof: Suppose $\sigma = (s_1, \ldots, s_\ell, A, B, C)$ is a valid signature of a message $m \in \{0,1\}^*$ under the predicate $\Gamma = (\mathbf{M}_{\ell \times k}, \rho)$, and the signature is created by some *unknown* signer who has the attribute set S with $SK_S = \{K, L, \{K_x\}_{x \in S}, g^t\}$. Then we know

$$\begin{aligned} A &= yKH(m)^{r_1} g^{r_2} \\ &= yg^{\alpha} g^{a(t+t^2)} H(m)^{r_1} g^{r_2} \\ &= g^{\alpha} H(m)^{r_1} T g^{r_2} g^{at} y. \end{aligned}$$

Here $y = \prod_{i=1}^{\ell} (K_{\rho(i)}^{\alpha_i} h_{\rho(i)}^{\beta_i})$, in which $(\alpha_1, \ldots, \alpha_\ell) = \vec{\alpha}$ is a vector satisfying $\vec{\alpha}\mathbf{M} = (1, 0, \ldots, 0)$, $(\beta_1, \ldots, \beta_\ell) = \vec{\beta}$ is a vector satisfying $\vec{\beta}\mathbf{M} = (0, 0, \ldots, 0)$. Next, after we choose $\vec{v} = (1, v_2, \ldots, v_k)$ and compute λ_i in the Verify algorithm, we have

$$\sum_{i=1}^{\ell} \alpha_i \lambda_i = \sum_{i=1}^{\ell} \alpha_i \sum_{j=1}^{k} (v_j \mathbf{M}_{i,j}) = \sum_{j=1}^{k} v_j \sum_{i=1}^{\ell} (\alpha_i \mathbf{M}_{i,j}) = 1.$$

Similarly, $\sum_{i=1}^{\ell} \beta_i \lambda_i = 0$.
Thus,

$$e(g, g^{at}) = e(g, g^{\sum_{i=1}^{\ell} a\lambda_i (t\alpha_i + \beta_i)}) = \prod_{i=1}^{\ell} e(Z^{\lambda_i}, s_i).$$

Finally, we have

$$\begin{aligned} e(g, A) &= e(g, g^{\alpha}) e(g, H(m)^{r_1}) e(g, T g^{r_2}) e(g, g^{at}) e(g, y) \\ &= Ye(H(m), B) e(g, C) \prod_{i=1}^{\ell} e(Z^{\lambda_i}, s_i) e(g, \prod_{i=1}^{\ell} (K_{\rho(i)}^{\alpha_i} h_{\rho(i)}^{\beta_i})) \\ &= Ye(g, C) e(H(m), B) \prod_{i=1}^{\ell} e(Z^{\lambda_i} h_{\rho(i)}, s_i). \end{aligned}$$

So, our scheme is correct. ■

B. Unforgeability

Next, we will show existential unforgeability of our scheme in the selective predicate model. Some other ABS schemes [11], [8] also gave security proof in the same model. But in our scheme, we use a hash function modeled as a random oracle to improve efficiency and support arbitrary messages, our proof would consider how the challenger answers the random queries from the adversary.

Theorem 2: The traceable attribute-based signature scheme in section III is existentially unforgeable in the selective predicate model under the q-ADHE assumption in the random oracle model.

More specifically, if the adversary breaks our ABS scheme with non-negligible probability ϵ in time t, and makes q_R random oracle queries, q_S *Sign* oracle queries ($q_R > q_S$), then we can build a simulator to solve the q-ADHE problem in time t with probability $\epsilon' \geq (1 - \frac{q_S}{q_R}) \cdot \epsilon$.

Proof: Suppose there is a polynomial adversary $\mathcal{A}$, that can break our scheme in the selective predicate model, then we can build a simulator $\mathcal{S}$, which can solve the q-ADHE problem.

Sketch: According to the selective predicate security model in section II-A, the simulator $\mathcal{S}$ receives the challenge predicate in **Init** phase, and publishs public parameters in **Setup** phase. Next, $\mathcal{S}$ will answer the adversary $\mathcal{A}$'s queries to the *Extract* oracle and *Sign* oracle in **Query** phase. Finally, $\mathcal{S}$ will use the forged signature $\mathcal{A}$ outputs in **Forge** phase to solve the q-ADHE problem. And as we use a hash function modeled as a random oracle, $\mathcal{S}$ should control the oracle and respond to the random oracle queries.

The detailed simulation proceeds as follows:

First, the simulator $\mathcal{S}$ takes in the q-ADHE challenge

$$(g, y_1, \ldots, y_q, y_{q+2}, \ldots, y_{2q}, y_{2q+1})$$

where $y_i = g^{a^i}$, and is asked to compute y_{q+1}.

- **Init**. The adversary $\mathcal{A}$ declares the challenge signing predicate $\Gamma^* = (\mathbf{M}^*, \rho^*)$, where $\mathbf{M}^*$ is a $\ell^* \times k^*$ $(2k^* < q)$ matrix and ρ^* is an *injective* function that maps rows of $\mathbf{M}$ to attributes.

- **Setup**. The simulator $\mathcal{S}$ chooses random $\alpha' \in Z_p$ and implicitly set $\alpha = \alpha' + a^{q+1}$ by letting $Y = e(g,g)^{\alpha} = e(y_1, y_q)e(g,g)^{\alpha'}$. Next, $\mathcal{S}$ defines group elements $h_1, \ldots, h_U \in G$. For each attribute x, $1 \le x \le U$, choose random $z_x \in Z_p$. If x appears in the challenge predicate Γ^*, i.e. there exists an $i \in [1, \ell^*]$, such that $\rho^*(i) = x$, then let

$$h_x = g^{z_x} y_1^{-\mathbf{M}^*_{i,1}} y_2^{-\mathbf{M}^*_{i,2}} \cdots y_{k^*}^{-\mathbf{M}^*_{i,k^*}}.$$

Otherwise, let $h_x = g^{z_x}$.
Finally, $\mathcal{S}$ publishes the public parameters

$$PK = \{G, G_T, e, p, g, Y, Z = y_1, h_1, \ldots, h_U, H\}$$

where $H : \{0,1\}^* \to G$ is a random oracle controlled by $\mathcal{S}$.
We point it out here that the parameters are distributed randomly due to the g^{z_x} factor. Besides, by the restriction ρ^* is an injective function, our assignment is unambiguous.

- H**-queries**. At any time, $\mathcal{A}$ and $\mathcal{S}$ can query the random oracle $H : \{0,1\}^* \to G$. To respond to these queries, $\mathcal{S}$ maintains a list of tuples (m, Q_m, r_m, T_m) as explained below. We refer to this list as H^{list}. The list is initially empty.
When $\mathcal{A}$ queries for $H(m)$, $\mathcal{S}$ responds as follows:
  1) If the query m already appears on the H^{list} in a tuple (m, Q_m, r_m, T_m), then $\mathcal{S}$ responds with $H(m) = Q_m \in G$.
  2) Otherwise, $\mathcal{S}$ picks a random $r_m \in Z_p$ and computes $Q_m = g^{r_m}$, then it adds the tuple $(m, Q_m, r_m, 0)$ to H^{list} and responds to $\mathcal{A}$ with $H(m) = Q_m$.

  And when $\mathcal{S}$ queries for $H(m)$, $\mathcal{S}$ responds as follows:
  1) If the query m already appears on the H^{list} in a tuple (m, Q_m, r_m, T_m), then $\mathcal{S}$ responds with (Q_m, r_m, T_m).
  2) Otherwise, $\mathcal{S}$ picks a random $r_m \in Z_p$ and computes $Q_m = y_q g^{r_m}$, then it adds the tuple $(m, Q_m, r_m, 1)$ to H^{list} and responds with $(Q_m, r_m, 1)$.

  We point it out here that T_m in the tuple is to denote the message m is first queried by $\mathcal{A}$ ($T_m = 0$) or $\mathcal{S}$ ($T_m = 1$).

- **Query**. The adversary $\mathcal{A}$ can adaptively query the *Extract* Oracle and *Sign* Oracle, and $\mathcal{S}$ answers these queries in the following way:

Extract Oracle: Suppose $\mathcal{A}$ queries the private key for a set S where S does not satisfy $\mathbf{M}^*$. $\mathcal{S}$ first finds a vector $\vec{w} = (w_1, \ldots, w_{k^*}) \in Z_p^{k^*}$ such that $w_1 = -1$ and for all i where $\rho^*(i) \in S$ we have $\vec{w}\mathbf{M}^*_i = 0$.
Then, $\mathcal{S}$ chooses a random $r \in Z_p^*$, and if q is even, $\mathcal{S}$ would implicitly defines t as

$$t = \frac{r^2}{2} + w_1 a^q + w_2 a^{q-1} + \cdots + w_{k^*} a^{q-k^*+1} + r a^{\frac{q}{2}}.$$

As $2k^* < q$, we have $q - k^* + 1 > \frac{q}{2}$. Then $\mathcal{S}$ computes

$$\begin{aligned} L &= g^t = g^{\frac{r^2}{2}} y_{\frac{q}{2}}^{r} \prod_{i=1}^{k^*} y_{q+1-i}^{w_i}, \\ T &= g^{at^2} = y_1^{\frac{r^4}{4}} y_{\frac{q}{2}+1}^{r^3} \prod_{i=2}^{k^*} y_{q+2-i}^{r^2 w_i} \cdot \prod_{i=1}^{k^*} y_{\frac{3q}{2}+2-i}^{2rw_i} \cdot \\ &\quad \prod_{i=1}^{k^*} \prod_{j=1}^{k^*} y_{2q+3-i-j}^{w_i w_j}, \\ K &= g^{\alpha} g^{a(t+t^2)} = T \cdot g^{\alpha'} y_1^{\frac{r^2}{2}} y_{\frac{q}{2}+1}^{r} \cdot \prod_{i=2}^{k^*} y_{q+2-i}^{w_i}. \end{aligned}$$

Next, $\mathcal{S}$ will compute K_x for all $x \in S$. If x appears in the challenge predicate Γ^*, and $\rho^*(i) = x$ for some $i \in [1, \ell^*]$, then

$$K_x = L^{z_x} \prod_{j=1}^{k^*} \Big(y_j^{\frac{r^2}{2}} y_{\frac{q}{2}+j}^{r} \prod_{k=1,\ldots,k^*}^{k \neq j} y_{j+q+1-k}^{w_k}\Big)^{-\mathbf{M}^*_{i,j}}.$$

If x doesn't appear in Γ^*, $K_x = L^{z_x}$.
Considering $2k^* < q$ and the known y_i, we can easily compute the above $(L, T, K, \{K_x\}_{x \in S})$. And if q is odd, $\mathcal{S}$ would implicitly set

$$\begin{aligned} t &= r + w_1 a^q + w_2 a^{q-1} + \cdots + \\ &\quad w_{k^*} a^{q-k^*+1} + r a^{(q+1)/2} + a^{(q-1)/2} \end{aligned}$$

and compute in a similiar way.
Finally, $\mathcal{S}$ replies $\mathcal{A}$ with the private key

$$SK_S = \{K, L, T, \{K_x\}_{x \in S}\}.$$

Note: the way we construct K_x here is motivated by the security proof in [5].

Sign Oracle: Suppose $\mathcal{A}$ queries for a signature of (m, Γ), where $m \in \{0,1\}^*$ and $\Gamma = (\mathbf{M}_{\ell \times k}, \rho)$.
First, $\mathcal{S}$ checks if there exist a set S s.t. $\Gamma(S) = 1$ but $\Gamma^*(S) \neq 1$, if such $\mathcal{S}$ exists, $\mathcal{S}$ will simply generate the private key for S and **Sign**(m, Γ). Otherwise, $\mathcal{S}$ queries $H(m)$ and gets (Q_m, r_m, T_m), if $T_m = 0$, $\mathcal{S}$ aborts the simulation; if $T_m = 1$, $\mathcal{S}$ finds a vector $\vec{\alpha} = (\alpha_1, \ldots, \alpha_\ell)$ such that $\vec{\alpha}\mathbf{M} = (1, 0, \ldots, 0)$, besides, $\mathcal{S}$ also chooses a random vector $\vec{\beta} = (\beta_1, \ldots, \beta_\ell)$ such that $\vec{\beta}\mathbf{M} = (0, 0, \ldots, 0)$.
Then, $\mathcal{S}$ chooses random $t, r_0, r_2 \in Z_p$ and implicitly sets $r_1 = r_0 - a$, next it computes

$$\begin{aligned} s_i &= g^{t\alpha_i} g^{\beta_i} = L^{\alpha_i} g^{\beta_i}, (i = 1, \ldots, \ell) \\ B &= g^{r_0} g^{-a} = g^{r_1},\ C = g^{at^2} g^{r_2}, \\ y &= \prod_{i=1}^{\ell} h_{\rho(i)}^{t\alpha_i + \beta_i} = \prod_{i=1}^{\ell} (K_{\rho(i)}^{\alpha_i} h_{\rho(i)}^{\beta_i}), \\ A &= y g^{\alpha'} g^{a(t+t^2)} g^{r_2} y_q^{r_0} B^{r_m} = yKH(m)^{r_1} g^{r_2}. \end{aligned}$$

Finally, $\mathcal{S}$ returns $\sigma = (s_1, \ldots, s_\ell, A, B, C)$.

- **Forge**. After polynomially bounded number of queries, $\mathcal{A}$ will output a forged signature σ^* of a message $m^* \in \{0,1\}^*$ for the challenge predicate Γ^*, where $\sigma^* = (s_1^*, \ldots, s_\ell^*, A^*, B^*, C^*)$.

First, $\mathcal{S}$ queries $H(m^*)$ and gets $(Q_{m^*}, r_{m^*}, T_{m^*})$. If $T_{m^*} = 1$, $\mathcal{S}$ aborts the simulation; If $T_{m^*} = 0$, $\mathcal{S}$ implicitly sets vector $\vec{v} = (-1, -a, -a^2, \dots, -a^{k^*-1})$ and for $i \in [1, \ell^*]$, computes

$$\lambda_i = \vec{v}\mathbf{M}_i^* = -\sum_{j=1}^{k^*} a^{j-1}\mathbf{M}_{i,j}^*.$$

Thus, for each $i \in [1, \ell^*]$, $\rho^*(i) = x$,

$$h_x = g^{z_x} y_1^{-\mathbf{M}_{i,1}^*} y_2^{-\mathbf{M}_{i,2}^*} \dots y_{k^*}^{-\mathbf{M}_{i,k^*}^*} = g^{z_x} g^{a\lambda_i}.$$

And for any $\vec{\alpha} = (\alpha_1, \dots, \alpha_{\ell^*})$, $\vec{\beta} = (\beta_1, \dots, \beta_{\ell^*})$ such that $\vec{\alpha}\mathbf{M}^* = (1, 0, \dots, 0)$, $\vec{\beta}\mathbf{M}^* = (0, 0, \dots, 0)$, we have

$$\sum_{i=1}^{\ell^*} \lambda_i \alpha_i = -1, \; \sum_{i=1}^{\ell^*} \lambda_i \beta_i = 0.$$

Next, if σ^* is a valid signature, that is

$$\begin{aligned} A^* &= yKH(m^*)^{r_1} g^{r_2} = g^\alpha g^{at} H(m^*)^{r_1} g^{at^2 + r_2} y \\ &= g^{\alpha'} y_{q+1} (B^*)^{r_{m^*}} C^* \prod_{i=1}^{\ell^*} s_i^{z_{\rho^*(i)}}. \end{aligned}$$

So, $\mathcal{S}$ can compute

$$y_{q+1} = A^* / (g^{\alpha'} (B^*)^{r_{m^*}} C^* \prod_{i=1}^{\ell^*} s_i^{z_{\rho^*(i)}}).$$

as the solution to the q-ADHE problem, which is in contradiction to the q-ADHE assumption.

Finally, we should compute the probability that the simulation will abort:

1) $\mathcal{A}$ queries $H(m)$ before he queries for a signature of m under any predicate Γ for the first time.
2) For the forged signature of message m^*, $H(m^*)$ is first queried by $\mathcal{S}$.

By simple observation, the simulation will abort with probability q_S/q_R, where q_S and q_R are the number of queries $\mathcal{A}$ makes to the *Sign* oracle and the random oracle, respectively.

So, suppose $\mathcal{A}$ has probability ϵ to break our scheme and forge a valid signature, then $\mathcal{S}$ will solve the q-ADHE problem with probability $\epsilon' \geq (1 - \frac{q_S}{q_R}) \cdot \epsilon$. ■

C. Traceability

We prove traceability in two steps: First, the Trace algorithm is correct; second, every valid private key in our scheme cannot be faked, hence only the original signer has his own secret key.

Lemma 1: The Trace algorithm in the traceable attribute-based signature scheme in section III is correct.

This can be easily proven in a similar way as in the proof of correctness.

Lemma 2: Suppose the IDH problem is hard, then given one or multiple valid private key(s), every polynomial-time adversary $\mathcal{A}$ cannot fake a new valid private key.

Proof: First, we simplify the problem: given public params $PK = (Y = e(g,g)^\alpha, Z = g^a)$, one or multiple tuple(s) $D_t = (K_t, L_t, T_t)$ with $K_t = g^\alpha g^{at}, L_t = g^t, T_t = g^{at^2}$, and denote the set of these tuples as D^{list}, $\mathcal{A}$ is asked to compute $D_{t^*} \notin D^{list}$.

Next, we can easily find that given a tuple D_t, it is hard to compute g^{at} if the IDH problem is hard. And based on this, we will prove lemma 2 in 2 steps.

1) If D^{list} only contains 1 tuple D_t. Assume $\mathcal{A}$ can compute D_{t^*}. Then we have $t^* = it + j$, in which $i, j \in Z_p$ are known to $\mathcal{A}$. That is

$$\begin{aligned} L_{t^*} &= g^{t^*} = (L_t)^i \cdot g^j, \\ T_{t^*} &= (T_t)^{i^2} \cdot (g^{at})^{2ij} \cdot (Z)^{j^2}. \end{aligned}$$

Since $\mathcal{A}$ doesn't know g^{at}, he must set $ij = 0$. If $j = 0, t^* = it$, then

$$K_{t^*} = g^\alpha g^{ait} = K_t \cdot (g^{at})^{i-1},$$

so $i - 1 = 0, t^* = t, D_{t^*} = D_t$. If $i = 0, t^* = j$, then

$$K_{t^*} = g^\alpha g^j = K_t \cdot g^j \cdot g^{-at},$$

which $\mathcal{A}$ can't compute. That is, $\mathcal{A}$ can't compute $D_{t^*} \notin D^{list}$.
Note: Actually, we should set $t^* = it + j + kat^2 + la$, but easily we can find $k = l = 0$.

2) If D^{list} contains multiple tuples. Take $D^{list} = \{D_{t_1}, D_{t_2}\}$ as an example, and assume $\mathcal{A}$ can compute D^{t^*}. Then we have $t^* = it_1 + jt_2 + k$, in which $i, j, k \in Z_p$ are known to $\mathcal{A}$. That is

$$\begin{aligned} L_{t^*} &= g^{t^*} = L_{t_1}^i L_{t_2}^j g^k, \\ K_{t^*} &= (K_{t_1})^i (K_{t_2})^j g^k (g^\alpha)^{1-(i+j)}. \end{aligned}$$

Since $\mathcal{A}$ doesn't know g^α, he must set $i + j = 1$. Then

$$\begin{aligned} T_{t^*} &= g^{a(it_1 + jt_2 + k)^2} \\ &= (T_{t_1})^{i^2} (T_{t_2})^{j^2} (K_{t_1})^{2ki-2k} (K_{t_2})^{2kj} (g^{at_1})^{2ijt_2 + 2k}. \end{aligned}$$

And since $\mathcal{A}$ doesn't know g^{at_1}, he must set $2ijt_2 + 2k = 0$. But t_2 is a random value, thus $ij = 0, k = 0$, that is $t^* = t_1$ or $t^* = t_2$, i.e. $D_{t^*} \in D^{list}$.

Finally, lemma 2 is proved. ■

With lemma 1 and 2, we can get the following conclusion:

Theorem 3: The traceable attribute-based signature scheme in section III is traceable.

D. Privacy

Theorem 4: The traceable attribute-based signature scheme in section III satisfies privacy.

Proof: First, we prove that given a signature, nobody can find out the attributes the signer used to sign. This is achieved by the random vector $\vec{\beta}$. Given a signing predicate $\Gamma = (\mathbf{M}, \rho)$, $\mathbf{M}$ is a an $\ell \times k$ matrix, and Γ is not a (n, n)-threshold predicate (otherwise the signer needs all attributes in Γ and this analysis is meaningless), thus we have $rank(\mathbf{M}) < \ell$. Then the number

TABLE I. COMPARISON BETWEEN ABS SCHEMES

scheme	Maji et al.[6]	Okamoto et al.[9]	Escala et al.[10]	our scheme
Signature size	$\ell + k + 2$	$7\ell + 11$	$8\ell + k + 7$	$\ell + 3$
Computation	$k\ell + k + 3$	$7\ell + 15$	$-$	$\ell + 3$
Unforgeability	full	full	full	selective predicate
Model	generic group	standard	random oracle	random oracle
Privacy	Pefect Privacy	Perfect Privacy	Privacy	Privacy
Traceability	$\times$	$\times$	$\surd$	$\surd$

of vector $\vec{\beta}$ s.t. $\vec{\beta}\mathbf{M} = (0, \ldots, 0)$ is polynomial, so $\vec{\beta}$ can hide the attributes used to sign.

Next, given a signature, only the issuing authority and PKG working together can find out the identity of the signer. Notice that anyone can compute $e(g, L)$ as in the Trace algorithm, but the mapping from $e(g, L)$ to the signer's ID is stored in two tables, maintained by the issuing authority and PKG. ■

Remark: Our scheme doesn't satisfy perfect privacy: given any two signatures of any message, anyone can figure out that whether the two signatures are generated by the same signer.

E. Standard Model

TABS scheme uses a hash function modeled as a random oracle to improve efficiency. To be secure in the standard model, the technique in [20] can be used to modify our scheme. That is, in the Setup algorithm, let PKG define a public parameter $n \in Z_p$, restrict all the messages $m \in \{0,1\}^n$, and choose public $\mu_0, \ldots, \mu_n \in G$. Then in the Sign and Verify algorithm, replace $H(m)$ by $\mu_0 \prod_{i \in \gamma} \mu_i$, in which $\gamma = \{i \in \{1, \ldots, n\}, m_i = 1\}$.

F. Efficiency

Compared to the ABS schemes in [6], [9], [10], our scheme generates a constant-size, shorter signature and has a smaller computation cost. And the size of PKG public parameters and user private keys is also contant and shorter. The comparison between ABS schemes is listed in the Table I. In the table, ℓ, k is the size of matrix in the signing predicate, and we only count the number of pairing operations for computation analysis.

In Table I, all schemes support general access structures, especially Okamoto et al. [9] uses non-monotone access structures. Maji et al.'s scheme in the table is the efficient third instance in [6] without NIZK. [10] scheme uses NIWI, so it is hard to compare its computation. [14] is not listed in the table for the same reason.

V. CONCLUSIONS

In this paper, we give a new structure and definition of Traceable Attribute-Based Signature (TABS) based on the work in [10]. The TABS has a good balance between privacy and traceability: each single user cannot link a signature to identity (the set of attributes) of the signer, while there is a method to trace the signers of malicious signatures. This primitive will have great potential in distributed environments where users' privacy and traceability are both important. We give the formal syntax and security definition of TABS, and also construct an efficient TABS scheme without NIZK or NIWI proofs. Next, we prove the scheme's existential unforgeability in the selective predicate model under the Diffie-Hellman Exponent assumption in the random oracle and prove its traceability and privacy. We also give a method to make our scheme secure in the standard model. Compared with other ABS schemes, our scheme generates shorter signatures and is more efficient, thus is more suitable for practical applications.

REFERENCES

[1] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *EUROCRYPT*, 2005, pp. 457–473.

[2] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *ACM Conference on Computer and Communications Security*, 2006, pp. 89–98.

[3] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *IEEE Symposium on Security and Privacy*, 2007, pp. 321–334.

[4] A. B. Lewko, T. Okamoto, A. Sahai, K. Takashima, and B. Waters, "Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption," in *EUROCRYPT*, 2010, pp. 62–91.

[5] B. Waters, "Ciphertext-policy attribute-based encryption: An expressive, efficient and provably secure realization," in *Public Key Cryptography*, 2011, pp. 53–70.

[6] H. K. Maji, M. Prabhakaran, and M. Rosulek, "Attribute-based signatures," in *CT-RSA*, 2011, pp. 376–392.

[7] J. Li and K. Kim, "Attribute-based ring signatures," *IACR Cryptology ePrint Archive*, vol. 2008, p. 394, 2008.

[8] J. Li, M. H. Au, W. Susilo, D. Xie, and K. Ren, "Attribute-based signature and its applications," in *ASIACCS*, 2010, pp. 60–69.

[9] T. Okamoto and K. Takashima, "Efficient attribute-based signatures for non-monotone predicates in the standard model," in *Public Key Cryptography*, 2011, pp. 35–52.

[10] A. Escala, J. Herranz, and P. Morillo, "Revokable attribute-based signatures with adaptive security in the standard model," in *AFRICACRYPT*, 2011, pp. 224–241.

[11] A. Ge, C. Chen, C. Ma, and Z. Zhang, "Short and efficient expressive attribute-based signature in the standard model," *IACR Cryptology ePrint Archive*, vol. 2012, p. 125, 2012.

[12] J. Groth, R. Ostrovsky, and A. Sahai, "Perfect non-interactive zero knowledge for np," in *EUROCRYPT*, 2006, pp. 339–358.

[13] J. Groth and A. Sahai, "Efficient non-interactive proof systems for bilinear groups," in *EUROCRYPT*, 2008, pp. 415–432.

[14] A. El Kaafarani, E. Ghadafi, D. Khader, "Decentralized Traceable Attribute-Based Signatures," in *CT-RSA*, 2014, pp 327-348.

[15] A. Beimel, "Secure schemes for secret sharing and key distribution," Ph.D. dissertation, Israel Institute of Technology, Technion, Haifa, Israel, 1996.

[16] D. Boneh, X. Boyen, and E.-J. Goh, "Hierarchical identity based encryption with constant size ciphertext," in *EUROCRYPT*, 2005, pp. 440–456.

[17] A. B. Lewko and B. Waters, "Decentralizing attribute-based encryption," in *EUROCRYPT*, 2011, pp. 568–588.

[18] D. Boneh and M. K. Franklin, "Identity-based encryption from the weil pairing," in *CRYPTO*, 2001, pp. 213–229.

[19] D. Boneh and X. Boyen, "Secure identity based encryption without random oracles," in *CRYPTO*, 2004, pp. 443–459.

[20] B. Waters, "Efficient identity-based encryption without random oracles," in *EUROCRYPT*, 2005, pp. 114–127.