Revocable Attribute-based Key Agreement Protocol without Random Oracles

Hao Wang
School of computer science and technology, Shandong University, Jinan, China
Email: whatsd@gmail.com

Qiu-Liang Xu and Xiu Fu
School of computer science and technology, Shandong University, Jinan, China
Email: xuqiuliang@sdu.edu.cn, fux2006@126.com

Abstract—In this paper, we present a two-party attribute-based key agreement protocol, which is secure in the standard model (without random oracles), under the truncated decision $q$-ABDHE assumption. Then we give a modified version of this scheme, in which the users can be revoked efficiently. The attribute-based key agreement protocol is based on the attribute-based encryption scheme, which is a generalization of identity-based cryptosystems, incorporates attributes as inputs to its cryptographic primitives. These kinds of schemes not only preserve the advantages of traditional identity-based key agreement protocol, but also provide some new properties, such as hiding the identity information of the individual, increasing the flexibility of key management, and providing efficient means to revoke users from the system.

Index Terms—attribute-based key agreement, without random oracles, revocation

I. INTRODUCTION

Key establishment is a process whereby two (or more) entities can establish a shared secret key (session key). There are two different approaches to key establishment between two entities. In one scenario, one entity generates a session key and securely transmits it to the other entity. This is known as enveloping or key transport. More commonly, both entities contribute information from which a joint secret key is derived. This is known as key agreement. All the protocols discussed in this paper are of this latter form.

The first key agreement protocol based on asymmetric cryptography was proposed by Diffie and Hellman in 1976 [4]. In 1984, Shamir [11] introduced the concept of Id-based cryptosystem where a user’s private key is generated by a trusted private key generator (PKG) and any party could easily derive the user’s public key from his identity. An authenticated key agreement protocol is called identity-based if in the protocol, users use an identity-based asymmetric key pair instead of a traditional public/private key pair for authentication and determination of the established key.

In 2005, Sahai and Waters first presented the concept of attribute-based encryption (ABE) [10]. In an ABE system, user’s keys and ciphertexts are labeled with sets of descriptive attributes and a particular key can decrypt a particular ciphertext only if there is a match between the attributes of the ciphertext and the user’s key. Then, a few attribute-based cryptosystems were developed [1, 8, etc.].

Based on above works, Wang et al. presented the first two-party attribute-based key agreement protocol in [12]. In this protocol, two parties use attribute information of the other party instead of the traditional public key to generate the session key shared between them. For example, Alice is an employee of the department $D_a$, and Bob is an employee of group $G_b$ in the department $D_b$, company $C_b$. The attribute of Alice is $\omega_a = (D_a, C_a)$, and the attribute of Bob is $\omega_b = (G_b, D_b, C_b)$. They can use their attribute information as public key to generate a symmetric key for a secure session.

Moreover, the attribute-based key agreement has a merit which an identity-based key agreement does not have: hiding the identity information of the individual. In the above example, Alice only knows the other party is one of employees of group $G_b$ in department $D_b$, company $C_b$, but she doesn’t know who he/she is, vice versa. This property is meaningful in the practical application, because in many scenes, hiding the identity information often is necessary.

Though Wang’s scheme is practical, its security relies heavily on random oracle model. Many researchers have expressed doubts about the wisdom of relying on the ROM. In particular, Canetti, Goldreich and Halevi [15] proved ten years ago that there are signature and encryption schemes which are secure in the ROM, but insecure for any instantiation of the random oracle. Furthermore, RSA-FDH and many other schemes provably secure in the Random-Oracle Model (ROM) require a cryptographic hash function whose output size does not match any of the standard hash functions. Recently, Leurent and Nguyen [16] show that the random-oracle instantiations proposed in the literature for such general cases are insecure.

In this paper, we present a new two-party attribute-
based key agreement protocol, which is secure in the standard model (without random oracle). Furthermore, we describe a revocable attribute-based key agreement scheme, and proof that the security of Revocable ABKA scheme in our modified security model and the security of ABKA in the original security model are equivalent.

The rest of the paper is organization as follows. In Section 2 we introduce the background knowledge and the definition of this paper. In Section 3 we propose our new two-party attribute-based key agreement protocol. We formally prove its security by reduction to the hardness of the truncated decision $q$-ABDHE assumption in Section 4. In Section 5, we present a modified attribute-based key agreement scheme, in which we can revoke users form the system efficiently. The conclusion of our work is given in Section 6.

II. PRELIMINARIES

A. Pairings

Let $G_1$ and $G_2$ denote two groups of prime order $p$, where $G_1$, with an additive notation, denotes a subgroup of the group of points on an elliptic curve; and $G_2$, with a multiplicative notation, denotes a subgroup of the multiplicative group of a finite field.

A pairing is a computable bilinear map between these two groups. For the purposes of this paper, we let $e$ denote a general bilinear map, i.e., $e: G_1 \times G_2 \rightarrow G_1$, which can be either a modified Weil pairing or a Tate pairing, and which has the following three properties:

Bilinear: If $P, Q \in G_1$ and $a, b \in Z_p^*$, then $e(P^a, Q^b) = e(P, Q)^{ab}$

Non-degenerate: There exists a $P \in G_1$ such that $e(P, P) \neq 1$.

Computable: If $P, Q \in G_1$, one can compute $e(P, Q)$ in polynomial time.

B. Intractability Assumptions

The security of our Attribute-Based key agreement scheme is based on the truncated decision $q$-ABDHE assumption [7].

The truncated decision $q$-ABDHE assumption:

Let $e: G_1 \times G_2 \rightarrow G_1$ be a bilinear map, we define the advantage function $Adv_{G_1, G_2}^{q$-ABDHE}$(k)$ of an adversary B as

$\text{Adv}_{G_1, G_2}^{q$-ABDHE}$(k)$ = $\Pr[B(G, ..., G, e(g, g)^{x+rx}) = 1] - \Pr[B(G, ..., G, e(g, g)^{y+ry}) = 1]$ where $x, y, z, r \in Z_p$.

We say that the truncated decision $q$-ABDHE assumption holds if $Adv_{G_1, G_2}^{q$-ABDHE}$(k) is negligible of all B.

C. Security Model of Authenticated Key Agreement

In this paper, we used a modified Bellare-Rogaway key agreement model [2] which is proposed by Chen et al. in [9]. In this model, each party involved in a session is treated as an oracle, and an adversary can access the oracle by issuing some specified queries (defined below).

An oracle $\Pi_{ij}^s$ denotes the $s$-th instance of party $i$ involved with a partner party $j$ in a session.

The security of a protocol is defined by a game with two phases. In the first phase, an adversary $E$ is allowed to issue the following queries in any order.

$Send(\Pi_{ij}^s, x)$: Upon receiving the message $x$, oracle $\Pi_{ij}^s$ execute the protocol and responds with an outgoing message $m$ or a decision to indicate accepting or rejecting the session. If the oracle $\Pi_{ij}^s$ does not exist, it will be created as initiator if $x = \lambda$, or as a responder otherwise. In this work, we require $i \neq j$, i.e., a party will not run a session with itself. Such restriction is not unusual in practice.

$Reveal(\Pi_{ij}^s)$. If the oracle has not accepted, it return $\bot$; otherwise, it reveals the session key.

$Corrupt(i)$. The party $i$ responds with its private key.

Once the adversary decides that the first phase is over, it starts the second phase by choosing a fresh oracle $\Pi_{ij}^s$ and issuing a $Test(\Pi_{ij}^s)$ query, where the fresh oracle $\Pi_{ij}^s$ and the $Test(\Pi_{ij}^s)$ query are defined as follow.

Definition 1 (fresh oracle) An oracle $\Pi_{ij}^s$ is fresh if (a) $\Pi_{ij}^s$ has accepted; (b) $\Pi_{ij}^s$ is unopened (not being issued the Reveal query); (c) party $j \neq i$ is not corrupted (not being issued the Corrupt query); (d) there is no opened oracle $\Pi_{ij}^s$, which has had a matching conversation to $\Pi_{ij}^s$.

$Test(\Pi_{ij}^s)$. Oracle $\Pi_{ij}^s$, which is fresh, as a challenger, randomly chooses $b \in \{0, 1\}$ and responds with the session key, if $b = 0$, or a random sample from the distribution of the session key otherwise.

After this point the adversary can continue querying the oracles except that it cannot reveal the test oracle $\Pi_{ij}^s$ or its partner $\Pi_{ij}^s$ (if it exists), and it cannot corrupt party $j$. Finally the adversary outputs a guess $b'$ for $b$. If $b' = b$, we say that the adversary wins. The adversary’s advantage is defined as:

$Adv^s(k) = \frac{Pr[E \text{ wins}] - \frac{1}{2}}{2}$

We use the session ID which can be the concatenation of the messages in a session to define matching conversations, i.e., two oracles $\Pi_{ij}^s$ and $\Pi_{ij}^t$ have matching conversations to each other if they have the same session ID.

A secure authenticated key (AK) agreement protocol is defined as follow.

Definition 2 Protocol $\Pi$ is a secure AK if:

1. In the presence of a benign adversary, which faithfully conveys messages, on $\Pi_{ij}^s$ and $\Pi_{ij}^t$, both oracles always accept holding the same session key, and this key is distributed uniformly on $\{0, 1\}^*$; and if for every adversary $E$:
2. If uncorrupted oracles $\Pi'_i$ and $\Pi'_j$ have matching conversations then both oracles accept and hold the same session key;
3. $Adv^s(k)$ is negligible.

### III. NEW TWO-PARTY ATTRIBUTE-BASED KEY AGREEMENT PROTOCOL

Use the encryption scheme which introduced by Fang et al. in [5], we construct an attribute-based two party key agreement protocol, that is secure in the standard model, under the truncated decision q-ABDHE assumption. Our attribute-based key agreement protocol (ABKA) consists of the following algorithms: Setup, Key-Gen, and Key-Agreement.

Let $G_1, G_2$ be groups of prime order $p$ and $g$ be a generator of $G_2$. We let $e$ denote an admissible bilinear map, i.e., $e: G_1 \times G_1 \rightarrow G_2$. We define the Lagrange interpolation coefficient $\Delta_{i,j}$ for $i \in Z_p$ and a set $S$, as element in $Z_p^*$. User's attribute set $\omega$ will be a subset of the universe $U$, where $U \subseteq Z_p^*$.

Suppose that Alice and Bob want to use the ABKA protocol to share a session key. The symbols $\omega_A$ and $\omega_B$ represent their attribute set separately. A description of this scheme is as follows:

**Setup:** Define the universe $U$ as $\{\mu_1, \ldots, \mu_m\}$, which is a set of all possible attributes, and every attribute maps to an element in $Z_p^*$. Pick $s_{\omega_A} = (s_{\omega_Aj})_{i\in \omega_A}, s_{\omega_B} \in Z_p^*$ at random $(x \neq i)$ and compute $X_{0} = g^{\mu_1}, \{X_j\}_{j \in \omega_A} = \{g^{\mu_j}\}_{j \in \omega_A}, h = g^\omega, V = e(g, h)$, Output a public parameter $\nu = (X_0, \{X_j\}_{j \in \omega_A}, V)$, and master key $mk = (s_{\omega_A}, \omega_A), \omega_B, \nu)$. The ABKA scheme is a secure authenticated key agreement protocol in the standard model under the truncated decision q-ABDHE assumption.

**Key-Gen:** To generate a private key for a user $M$ with attributes $\omega_M$ the following steps are taken. Two random polynomials $P(x)$ and $P(x)$ of degree $|\omega_M| - 1$ are randomly chosen. The private key of a set of attributes $\omega_M$ is $sk_{\omega_M} = (s_{\mu_{i,j}})_{i \in + \omega} = (s_{\mu_{i,j}})_{i \omega} = (\gamma_{i,j})_{i \omega}, \omega_M, s_{\mu_{i,j}} = P(i), d_{\mu_{i,j}} = g^{(s_{\mu_{i,j}}(i))_{i \omega}},$ for $i \in \omega, \omega_M, s_{\mu_{i,j}} \in Z_p^*, d_{\mu_{i,j}} = g^{(s_{\mu_{i,j}}(i))_{i \omega}}$.

**Key-Agreement:** Alice and Bob each randomly choose an ephemeral private key, $a, b \in Z_p$, and compute the values of the corresponding ephemeral public keys separately:

- $E_A = X_A^a, E_A = (X_A g^\omega)^{i \omega}, E_A = e(g, g)^a$ and $E_B = X_B^a, E_B = (X_B g^\omega)^{i \omega}, E_B = e(g, g)^a$.

Then they exchange the ephemeral public keys as follows:

- **Alice**→**Bob:** $E_A = E_A \| E_A \| E_A$
- **Bob**→**Alice:** $E_B = E_B \| E_B \| E_B$

Alice then computes secret $K_{ab}$ as follow:

$$K_{ab} = \prod_{i \omega} (e(E_A, d_{\mu_{i,j}}) \cdot E_A^a \cdot E_B^a)^{i \omega} \cdot e(E_B, d_{\mu_{i,j}}) \cdot E_B^a \cdot V^b,$$

Bob computes shared secret $K_{ab}$ as follow:

$$K_{ab} = \prod_{i \omega} (e(E_A, d_{\mu_{i,j}}) \cdot E_A^a \cdot E_B^a)^{i \omega} \cdot e(E_B, d_{\mu_{i,j}}) \cdot E_B^a \cdot V^b.$$

If Alice and Bob follow the protocol, they will compute the same shared secret:

$$K = K_{ab} = K_{aba} = e(g, h)^a.$$

This shared secret value $K$ is suitable to be used to derive a shared session key $s_k$ where we then use a key derivation function $H : [0, 1]^i \rightarrow [0, 1]^i$ to generate the shared session key: $sk = H(\omega_A \| \omega_B \| E_A \| E_B \| K)$, where $k = sk$.© 2009 ACADEMY PUBLISHER

### IV. SECURITY OF OUR ABKA SCHEME

**Theorem 2** The ABKA scheme is a secure authenticated key agreement protocol in the standard model under the truncated decision q-ABDHE assumption.

**Proof.** We will present our scheme satisfied the Definition 2. First, we explained that the condition 1 is satisfied: If two participants of the agreement follow the protocol specifications, and adversary $E$ is benign, then two participants both can receive the protocol message which correctly opposite party sends. Also according to the agreement accuracy analysis, we have $K_{ab} = K_{aba}$, and their conversation is matching. Therefore, they will obtain the same session key. And, this key distributed uniformly at random in session key space.

Next, the condition 2 also satisfy: If two participants of the agreement have not been corrupted, then they impossible to pretend to be by the adversary; If their conversation is matching, then means that they have received the protocol message which correctly opposite party sends, therefore they can obtain same session key. Below, we prove that the condition 3 is also satisfied.

We use the reduction to absurdity. Assume that there exists an adversary $E$ who can win the game (in section 2) with non-negligible advantage $\eta$ in time $t$. We can construct from $E$ a simulator $S$ which solves the truncated decision q-ABDHE problem with non-negligible probability.

Given input of, as described in Section 2, the two groups $G_1, G_2$, the bilinear map $e$, a generator of $g$ of $G_1$, and a truncated decision q-ABDHE instance $(g, g^\mu, \ldots, g^\mu, g^{\mu}, g^{\mu})$, $S$’s task is to distinguish $R = e(g, g^{\mu})^{\mu}$ from a random element in $G_2$.

**Setup:** $S$ picks a random polynomial $g_i(\mu)$ of degree $q$ and compute $X_0 = g^\mu, X_i = X_i^\mu, g^\mu, \mu = e(g, g)^{\mu}$, where $i \in U, e \in Z_p^*$, using the values $g^{\mu}, g^{\mu}, \ldots, g^{\mu}$.

Note that this does not change the distribution of the public parameter $(X_0, \{X_i\}_{i \omega}, V)$, and the simulator $S$ does not need to know the master key.

In order to guarantee that the secret of the polynomial $g_i(\mu)$, we limit the number of Corrupt queries made by adversary $E$ is at most $q - 1$. Moreover, we suppose that $E$ at most initiates $q$ sessions, namely for
any Oracle $\Pi'_{\omega_i}$, there is $i \in \{1, 2, \ldots, q_i\}$. $S$ randomly chooses 3 integer, $i, j \in \{1, 2, \ldots, q_i\}$, $n \in \{1, 2, \ldots, q_i\}$, and guesses that the adversary $E$ will make a Test query to Oracle $\Pi'_{\omega_i}$, where $\omega_i$ is the attributes set of the $i$th participant.

**Corrupt queries:** $S$ simulates the corrupt queries made by $E$ on the participant whose attribute is $\omega_i$, as follows:

1. If $\omega_i = \omega_j$, $S$ gives up.
2. Else, let $\Gamma = \omega_k \cap \omega_j$; Let $\Gamma$ be any set such that $\Gamma \subseteq \omega_k$ and $|\Gamma| = |\omega_k| - 1$; Let $T = \Gamma \cup \{0\}$.
3. Firstly, $S$ chooses a random element $s_{\omega_i} \in Z_p$, and picks a random polynomial $g_i(x)$ of degree $q$, such that $g_i(0) = s_{\omega_i}$. Then it defines a polynomial $g(x) = g_i(x) - g_i(0) = s_{\omega_i}$ of degree $q-1$, and sets $d_{k_i} = g_i(x)$.
4. Next, $S$ defines the private key components $(s_{k_i}, d_{k_i}, \omega_i)$, as follows:
   - For every $i \in \Gamma$ as, $S$ chooses a random element $s_{\omega_i} \in Z_p$, and picks a random polynomial $g_i(x)$ of degree $q$, such that $g_i(0) = s_{\omega_i}$.
   - Then it defines a polynomial $G_i(x) = g_i(x) - g_i(0) = s_{\omega_i}$.
   - And, sets $d_{k_i} = g_i(x)$.
5. The intuition behind these assignments is that we implicitly choose two random polynomials $g_i(x)$ and $g_j(x)$ for oracle $\Pi$.

The simulator also needs to calculate the decryption key values for all $i \in \omega_k - \Gamma$. We calculate these points to be consistent with our implicit choice of $P_i(x)$ and $P_2(x)$.

We define a polynomial of degree $q-1$:

$$G_i(x) = \sum_{i \in \Gamma} (g_i(x) - g_i(0)) \Delta_{i,j}(i) + \sum_{i \in \Gamma} (g_i(x) - g_i(0)) \Delta_{i,j}(i')$$

The key components for $i \in \omega_k - \Gamma$ are calculated as:

$$s_{k_i} = g_i(0) \Delta_{i,j}(i) + \sum_{i \in \Gamma} g_i(0) \Delta_{i,j}(i')$$

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

$$d_{k_i} = (g(x) - g_i(0))_{\omega_i}$$

Correctness:

$$d_{k_i} = (g(x) - g_i(0))_{\omega_i}$$

**Send queries:** $S$ answers all the Send queries as specified for a normal oracle, i.e., for the first Send query to an oracle, $S$ takes a random value in $Z_p$ to form its contribution, except that if $E$ asks $\Pi_{\omega_i}$, for any $t$, when adversary $E$ makes a Send query to an oracle $\Pi_{\omega_i}$, $S$ computes the message $E_t = E_{i_t} \oplus E_{i_t}$ for oracle $\Pi_{\omega_i}$, as follows:

$$S$$

defines a $q$ degree polynomial $G'(x) = x^{q-1}$, sets $a = G'(x)$, $E_{i_t} = E_{i_t} \oplus E_{i_t}$, $E_{i_t} = (G'(x))_t$, $E_{i_t} = (G'(x))_t$, $E_{i_t} = (G'(x))_t$, $E_{i_t} = (G'(x))_t$, $E_{i_t} = (G'(x))_t$ and sends the message $E_{i_t} = E_{i_t} \oplus E_{i_t}$ to adversary $E$. Then, generates secret key of $\omega_i$, $s_{k_{i_t}} = (s_{i_t}, d_{j_t}, (s_{i_t}, d_{j_t}))$. $S$ computes the session key $K_{i_t}$ as follow:

$$K_{i_t} = K_{i_t} = \prod_{i \in \omega_k} (E_{i_t}, d_{j_t}) \cdot E_{i_t} \oplus E_{i_t} \oplus E_{i_t} \oplus V'$$

where $r$ is randomly picked in $Z_p$. $S$ computes the session key $s_{i_t}$ for oracle $\Pi_{\omega_i}$ as follow:

$$s_{i_t} = H(a_t \oplus \omega_i) \otimes (E_{i_t} \oplus E_{i_t} \otimes K_{i_t})$$

Correctness:

$$E_{i_t} = X_{i_t}^{G'(x)} = (g(x))^{i_t} \oplus E_{i_t} = (g(x))^{i_t} \oplus E_{i_t} = (g(x))^{i_t} \oplus E_{i_t} = (g(x))^{i_t} \oplus E_{i_t} = (g(x))^{i_t} \oplus E_{i_t} = (g(x))^{i_t} \oplus E_{i_t} = (g(x))^{i_t} \oplus E_{i_t} = (g(x))^{i_t} \oplus E_{i_t} = (g(x))^{i_t} \oplus E_{i_t} = (g(x))^{i_t} \oplus E_{i_t}$$

Hence the session key $s_{i_t}$ created above is distributed the same as the one in the real attack.

**Reveal queries:** $S$ aborts and issues the warning, when an adversary $E$ makes Reveal query to oracle $\Pi_{\omega_i}$ or its matching oracle $\Pi'_{\omega_i}$. Otherwise, $S$ will give the matching oracle $\Pi'_{\omega_i}$ to ask the Test query, then $S$ aborts. However if $E$ does not choose $\Pi_{\omega_i}$, the session key of the oracle which was asked Reveal query to $E$.

**Test query:** At some point in the simulation, $E$ will ask a Test query of some oracle. If $E$ does not choose $\Pi'_{\omega_i}$ to ask the Test query, then $S$ aborts. However if $E$ aborts and issues the warning, when an adversary $E$ makes Reveal query to oracle $\Pi_{\omega_i}$ or its matching oracle $\Pi'_{\omega_i}$. Otherwise, $S$ will give the matching oracle $\Pi'_{\omega_i}$ to ask the Test query, then $S$ aborts. However if $E$ does not choose $\Pi'_{\omega_i}$, the session key of the oracle which was asked Reveal query to $E$. Therefore, the simulator is able to construct a private key for the attribute $\omega_k$. Furthermore, the distribution of the private key for $\omega_k$ is identical to that of original scheme since our choice of $P_i(x)$ and $P_2(x)$ induce two random $|\omega_k| \cdot 1$ degree polynomials $P_i(x)$, $P_2(x)$ and our construction of the private key components $s_{k_i} = (s_{i_t}, d_{j_t}, (s_{i_t}, d_{j_t}))$.
\( b \in \{0,1\} \), and return the session key held by \( \Pi_{\omega, i} \) if \( b = 0 \), or else a random key sampled from \( \{0,1\}^t \) if \( b = 1 \).

After making Test query, adversary \( E \) can make Corrupt queries to all the participant oracles except \( J \), and make Reveal queries to all the participant oracles except \( I \) and \( J \). Finally, \( E \) outputs its guess \( b' \in \{0,1\} \) for \( b \).

\( S \) does not abort at some point during the attack, namely, the view of adversary \( E \) in this game is completely same as in the real world. According to supposition \( E \) is a successful adversary, so \( |\Pr(b' = b) − 1/2| > \eta \).

To solve the truncated decision q-ABDHE problem, \( S \) directly gives the output of \( E \) (\( b' \)) as the reply return to its challenger of the truncated decision q-ABDHE problem.

We have proved that, when \( R = e(g, g)^{\omega t} \), the simulation is perfect, and we suppose that the adversary \( E \) can guess \( b \) correctly with the probability \( \eta + \sqrt{\frac{1}{2}} \).

Therefore, with the help of \( E \), \( S \) can solve the truncated decision q-ABDHE problem with a non-negligible probability.

\( \square \)

V. REVOCAIBLE ABKA AND ITS SECURITY

In this section, we describe a revocable attribute-based key agreement scheme endowed with a similar and equally efficient revocation mechanism as in the BGK system [3, 13]. In this modified attribute-based key agreement scheme, we can revoke users efficiently. Our construction uses the same binary tree structure as [3] and applies the same revocation technique.

A. Syntax of Revocable ABKA

**Definition 3** (Revocable ABKA) An attribute-based key agreement scheme endowed with efficient revocation and simply Revocable ABKA scheme \( RABKA = (\text{Setup}, \text{Key}−\text{Gen}, \uparrow \text{Key}−\text{Update}, \text{Key}−\text{Agreement}, \text{Revocation}) \) is defined by six algorithms and has associated session key space \( S \), attribute space \( U \) and time space \( \text{TIME} \). We assume that the size of \( \text{TIME} \) is polynomial in the security parameter. Each algorithm is run by either one of two types of parties - key authority or user. Key authority maintains a revocation list RL and state ST. Revocation list RL can be part of state ST, but we keep it explicit for clarity. In what follows, we call an algorithm stateful only if it updates RL or ST. We treat time as discrete as opposed to continuous.

**Setup:** The stateful Setup algorithm (run by key authority) takes input security parameter \( k \) and number of users \( n \), and returns parameters (system parameters) \( pk \), master key \( mk \), revocation list \( RL \) (initially empty) and state \( ST \).

**Key-Gen:** The stateful Key-Gen algorithm (run by key authority) takes as input parameters \( pk \), master key \( mk \), and an attribute set \( \omega \subset U \) as the identity, and returns a set of private keys \( sk_\omega \) and update state \( ST \).

**Upkey-Gen:** The Upkey-Gen algorithm (run by key authority) takes input public parameters \( pk \), master key \( mk \), update time \( t \in T \), revocation list \( RL \) and state \( ST \), and output update key \( uk_t \).

**Key-Update:** The deterministic Key-Update algorithm (run by user) take input private keys \( sk_\omega \), and update key \( uk_t \), and output state key \( SK_{\omega,t} \) or a special symbol \( \perp \) indicating that \( \omega \) was revoked. (We say an attribute set \( \omega \) was revoked at time \( t \) if Revoke algorithm was run by key authority on input \( (\omega_t, RL, ST) \) for any RL, ST.)

**Key-Agreement:** The Key-agreement algorithm is run by two users, who want to generate a symmetric key for a secure session. Each of them chooses an ephemeral private key, computes the values of the corresponding ephemeral public key \( epk \) (using the attribute \( \omega \) of the opposite party and time \( t \)), takes the ephemeral public key of the other party, the state key of his/her own, the public parameters as input, and outputs the session key \( K \).

**Revolve:** The stateful Revolve algorithm (run by key authority) takes input attribute to be revoked \( \omega \subset U \), revocation time \( t \in T \), revocation list \( RL \) and state \( ST \), and outputs an updated revocation list RL.

Correctness requires that, for any outputs \( (pk, mk) \) of Setup, any \( \omega \in U \) and \( t \in T \), all possible states \( ST \) and revocation list \( RL \), if \( \omega \) is not revoked by time \( t \), then for \( (sk_\omega, t) \leftarrow \text{Key} - \text{Gen}(pk, mk, \omega, ST) \), \( uk_t \leftarrow \text{Upkey} - \text{Gen}(pk, sk_t, RL, ST) \), \( SK_{\omega,t} \leftarrow \text{Key} - \text{Update}(sk_\omega, uk_t) \), then \( K = K_{\omega,t} = \text{Key} - \text{Agreement}(epk_t, SK_{\omega,t}, pk) \) = \( \text{Key} - \text{Agreement}(epk_t, SK_{\omega,t}, pk) = K'_{\omega,t} \).

**Remarks:** Note that we differentiate between the terms “private key” and “state key”.

B. Security Model of Revocable ABKA.

We modify the security model introduced in section 2, enables it to adapt the Revocable ABKA scheme. In our modified security model, an adversary \( E \) is allowed to issue two addition queries in any order at any time.

**Revoke(\( \omega,t \))**. The key authority (simulator) revoke the attribute set \( \omega \), and updated revocation list RL.

**Update-key(t)**. The key authority (simulator) responds with the update key \( uk_t \).

In addition, we modify the definition of fresh oracle.

**Definition 4** (Fresh oracle) An oracle \( \Pi'_{i,j} \) is fresh if (a) \( \Pi'_{i,j} \) has accepted; (b) \( \Pi'_{i,j} \) is unopened (not being issued the Reveal query); (c) party \( j \neq i \) is not corrupted (not being issued the Corrupt query) except it was revoked prior to the time when Test query was issued; (d) there is no opened oracle \( \Pi'_{i,j} \), which has had a matching conversation to \( \Pi_{i,j} \).

**Remarks.** Note that we differentiate between the term (c) of Definition 1 and Definition 4.

C. The BGK Construction

The idea of the scheme described by Boldyreva, Goyal and Kumar in [3] consists in assigning users to the leaves of a complete binary tree. Upon registration, the key
authority provides them with a set of distinct private keys (all corresponding to their identity) for each node on the path from their associated leaf to the root of the tree. During period \( t \), a given user’s decryption key can be obtained by suitably combining any one of its node private keys with a key update for period \( t \) and associated with the same node of the tree.

\[
\text{Path}(v) = \{\text{all corresponding to non-revoked users} \}
\]

The root node is denoted by \( \text{root} \). If \( v \) is a leaf node then \( \text{Path}(v) \) denotes the set of nodes on the path from \( v \) to \( \text{root} \) (both \( v \) and \( \text{root} \) inclusive). If \( v \) is a non-leaf node then \( v_L \) and \( v_R \) denote left and right child of \( v \). We assume that nodes in the tree are uniquely encoded as strings, and the tree is defined by all of its nodes descriptions.

Define a function \( \text{KUNodes} \) which is used to compute the minimal set of nodes for which key update needs to be published so that only non-revoked users at time \( t \) are able to decrypt ciphertexts. The function takes input a binary tree \( T \), revocation list \( RL \) and time \( t \) and outputs a set of nodes, which is the minimal set of nodes in \( T \) such that none of the nodes in \( RL \) with corresponding time \( \leq t \) (users revoked on or before \( t \)) have any ancestor (or, themselves) in the set, and all other leaf nodes (corresponding to non-revoked users) have exactly one ancestor (or, themselves) in the set. The function operates as follows. First mark all the ancestors of revoked nodes as revoked, then output all the non-revoked children of revoked nodes. Refer to Figure 2 for a pictorial depiction.

Here is a formal specification.

\[
\text{KUNodes}(T, RL, t)
\]

\[
X, Y \leftarrow \emptyset
\]

\[
\forall (v, t) \in RL
\]

\[
\text{if } t < t \text{ then add Path}(v) \text{ to } X
\]

\[
\forall x \in X
\]

\[
\text{if } x \notin X \text{ then add } x \text{ to } Y
\]

\[
\text{if } Y = \emptyset \text{ then add root to } Y
\]

\[
\text{Return } Y
\]

\[
\gamma, \delta \in \mathbb{Z}_p
\]

D. Main Construction of Revocable ABKA Scheme

At the beginning of the description, we restrict that every attribute set is defined to have the same size \( L \). (This is a reasonable restriction. Every attribute set can extend to have the same size. Furthermore, if the size of the attribute set is 1, the scheme is a traditional id-based key agreement scheme with revocation.)

**Setup:** Define the attribute space \( U \) and time space \( TIME \), where \( U = \{\mu_1, \ldots, \mu_U\} \subset \mathbb{Z}_p \) is a set of all possible attribute (every possible attribute maps into an element in \( Z_\gamma \)), and \( TIME = \{t_1, \ldots, t_{\text{TIME}}\} \subset \mathbb{Z}_p \) is a set of all promissory time (every promissory time maps into an element in \( Z_\gamma \)). Pick \( x_0, \{x_i\}_{i \in \text{TIME}}, y_0 \in Z_\gamma \) at random \((x_i \neq i)\) and compute \( X_0 = g^{x_0}, \{X_i\}_{i \in \text{TIME}} = \{g^{x_i}\}_{i \in \text{TIME}}, h = g^{y_0}, V = e(g, h) \). Let \( RL \) be an empty set and \( T \) be a binary tree with at least \( n \) leaf nodes. Output a public parameter \( \text{params} = (X_0, \{X_i\}_{i \in \text{TIME}}, V) \), master key \( mk = (x_0, \{x_i\}_{i \in \text{TIME}}, y_0) \), \( RL, ST = T \).

**Key-Gen:** To generate a private key for a user with attributes \( \omega \) the following steps are taken.

Pick an unassigned (If no such attributes are stored at leaf node we say that leaf node is unassigned) leaf node \( v \) from \( T \) and store \( \omega \) in that node.

For every node \( N \in \text{Path}(v) \), if \( N \) is undefined, chose two random polynomials \( P_{x_1}(x) \) and \( P_{x_2}(x) \) of degree \( L \), and stored them in node \( N \). (If no such polynomials are stored at node \( N \), we say that node \( N \) is undefined, otherwise, defined.) The private key of a set of attributes
\( \omega \) is \( \{sk_{\omega}, \}_{x \in \mathbb{N}_ts} = \{(s_{i,j}, d_{i,j}, s_{i,j}(x), d_{i,j})\}_{x \in \mathbb{N}_ts} \),
where \( s_{i,j} = P_{sk_i}(t), d_{i,j} = g^{P_{sk_i}(t)-s_{i,j}(x)} \), for \( i \in \omega \), and
\( s_{i,j} \in \mathbb{Z} \).

**Upkey-Gen:** For every Node \( N \in \text{KUNodes}(T, RL, t) \),
\( uk_i = \{uk_{i,j}\}_{x \in \mathbb{N}_ts} = \{(uk_{i,j}(x), uk_{i,j}(x))\}_{x \in \mathbb{N}_ts} \),
where \( uk_{i,j} = P_{sk_i}(t), uk_{i,j} = g^{P_{sk_i}(t)-uk_{i,j}(x)} \).

**Key-Update:**
For every \( (N, sk_{\omega}) \in sk_{\omega} \), \( (N', uk_{\omega}) \in uk \).
If \( \exists (N, N') \in \text{Test} \), \( N = N' \) then \( SK_{\omega} = (sk_{\omega}, uk_{\omega}) \).
Else (if \( sk_{\omega} \) and \( uk_{\omega} \) don't have any node in common) then \( SK_{\omega} \leftarrow ^- \).  

**Key-Agreement:** Alice and Bob each randomly choose an ephemeral private key, \( a,b \in \mathbb{Z} \) and compute the values of the corresponding ephemeral public keys separately:
\( E_{a} = X_{a}^{t}, E_{a} = (X_{a}^{t})^{s_{i,j}(0)} \cdot E_{a} = g^{a} \cdot g^{b} \) and
\( E_{b} = X_{b}^{t}, E_{b} = (X_{b}^{t})^{s_{i,j}(0)} \cdot E_{b} = g^{a} \cdot g^{b} \), where
\( t \) is an element of set \( \text{TIME} \), denotes the time of key-agreement execution.
Then they exchange the ephemeral public keys as follows:
Alice \( \rightarrow \) Bob: \( E_{a} = E_{a} \parallel E_{a} \parallel E_{a} \)
Bob \( \rightarrow \) Alice: \( E_{b} = E_{b} \parallel E_{b} \parallel E_{b} \).
For the time of key-agreement execution, Alice chooses the corresponding node \( N \), and uses \( \{sk_{\omega}, uk_{\omega}\} \) to compute the shared secret \( K_{ab} \) as follow:
\[ K_{ab} = \prod_{i,j \in \mathbb{N}_ts} (e(E_{a,j}, d_{a,j}) \cdot E_{b}^{s_{i,j}(0)} \cdot e(E_{a,j}, d_{a,j}) \cdot E_{b}^{s_{i,j}} \cdot V)^{a} \]
where \( s_{i,j} = uk_{i,j}, d_{i,j} = uk_{i,j} \), for \( i \in \mathbb{N}_ts \).
Bob computes shared secret \( K_{ab} \) as follow:
\[ K_{ab} = \prod_{i,j \in \mathbb{N}_ts} (e(E_{a,j}, d_{a,j}) \cdot E_{b}^{s_{i,j}(0)} \cdot e(E_{a,j}, d_{a,j}) \cdot E_{b}^{s_{i,j}} \cdot V)^{b} \]
where \( s_{i,j} = uk_{i,j}, d_{i,j} = uk_{i,j} \), for \( i \in \mathbb{N}_ts \).
If Alice and Bob follow the protocol, they will compute the same shared secret:
\[ K = K_{ab} = K_{ab} = g^{e(g, h)^{s_{i,j}}} \cdot \]

**Revoke:** To revoke an attribute set \( \omega \) at time \( t \) with revocation list RL and state ST, add \( (v, t) \) to RL for all node \( v \) associated with the attribute set \( \omega \). Return RL.

**E. Security of Revocable ABKA Scheme**

**Theorem 3** Let ABKA scheme is a secure authenticated key agreement protocol in original security model. Then Revocable ABKA scheme is a secure authenticated key agreement protocol in the modified security model.

**Proof.** There are three dissimilarities between our modified security model and the original model.

1. The adversary can make Revoke queries in any order at any time to revoke the attribute set.
2. The adversary can make Update-key queries in any order at any time to obtain the corresponding \( uk \).
3. The adversary can make Test query to the user, who has been made Corrupt query, iff this user has been revoked.

We will proof that the security of Revocable ABKA scheme in the modified security model and the security of ABKA in the original security model are equivalent.

In both security models, the security of the scheme is defined as the advantage of adversary win the game after make Test query. We will show that the advantage of adversary win the game after make Test query in the modified security model is same as that in the original model.

We classify the oracles which are used to Test query (\( \text{Test}(\Pi_{ab}) \)):

1. Neither of \( \Pi_{ab} \) and \( \Pi'_{ab} \) is revoked. In this case, the adversary can’t make Corrupt query to them, so the advantage of adversary is same as that in the original model.
2. Either of \( \Pi_{ab} \) or \( \Pi'_{ab} \) is revoked. Without loss of generality, we suppose that \( \Pi_{ab} \) is revoked.

Follow the Definition 4, \( \Pi'_{ab} \) can be corrupted by adversary, so the adversary can obtain \( sk_{\omega} \).
Furthermore, the adversary can make Update-key queries to obtain \( uk \), but there are no matching nodes \( (N, N') \in \text{Test} \), \( N = N' \), for every \( (N, sk_{\omega}) \in sk_{\omega}, (N', uk_{\omega}) \in uk \).
As a result, the adversary can’t obtain the state key \( SK_{\omega} \).
Finally, according to the property of ABKA, collusion among different users is not useful. In this case, the advantage of adversary is same as that in the original model.
3. Both of \( \Pi_{ab} \) and \( \Pi'_{ab} \) are revoked. This condition is same as condition 2.

In summary, the advantage of adversary win the game in the modified security model is same as that in the original model. Thus, if the ABKA scheme is a secure authenticated key agreement protocol in original security model, the Revocable ABKA scheme is a secure authenticated key agreement protocol in the modified security model.

VI. Conclusions

We proposed a new two-party attribute-based key agreement protocol. Comparing Wang’s scheme [12], which is provably secure in random oracle model, our scheme is secure in the standard model (without random oracles).

Furthermore, we presented a Revocable ABKA scheme, in which we can revoke users efficiently, and simply proved that the security of Revocable ABKA scheme in our modified security model and the security of ABKA in the original security model are equivalent.
ACKNOWLEDGMENT

The authors wish to thank anonymous reviewers for giving helpful suggestions. This work is supported by the National Nature Science Foundation of China under Grant No. 60873232, the National Nature Science Foundation of Shandong Province under Grant No. Y2007G37, and Shandong Postdoctoral Special Fund for Innovative Research under Grant No. 200803051.

REFERENCES


Hao Wang was born in Shandong, China in 1984. He has completed his Bachelor of Science in Computational Mathematics in Qufu Normal University, China in 2007. He is a Ph.D. candidate of Shandong University, interested in the research on public key cryptography including encryption, key agreement etc.

Qiu-Liang Xu was born in Shandong, China in 1960. He has a Ph.D. in Applied Mathematics (1999) and an M.S. in Computational Mathematics (1985) from Shandong University, Jinan, China.

He is a Professor of Computer Science in Shandong University, where he has been since 1985. His main interest is public key cryptography including encryption, digital signature, cryptographic protocol etc.

Xiu Fu was born in Shandong, China in 1985. She has completed her Bachelor of Engineering in computer science and technology in Jinan University of Jinan, China in 2007. Now she is an M.E. candidate of Shandong University, interested in the research on information security.