Received October 22, 2019, accepted November 7, 2019, date of publication November 13, 2019, date of current version November 22, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2953300

Ciphertext Outdate Attacks on the Revocable Attribute-Based Encryption Scheme With Time Encodings

KWANGSU LEE, (Member, IEEE)
Department of Computer and Information Security, Sejong University, Seoul 05006, South Korea
e-mail: kwangsu@sejong.ac.kr

This work was supported by the Institute for Information and Communications Technology Promotion (IITP) grant funded by the Korea Government, Ministry of Science and ICT (MSIT) (A Study on Functional Encryption: Construction, Security Analysis, and Implementation) under Grant 2016-6-00600.

ABSTRACT Cloud storage is a new computing paradigm that allows users to store their data in the cloud and access them anytime anywhere through the Internet. To address the various security issues that may arise in the cloud storage accessed by a large number of users, cryptographic encryption should be considered. Currently, researches on revocable attribute-based encryption (RABE) systems, which provide user revocation function and ciphertext update function by extending attribute-based encryption (ABE) systems that provide access control to ciphertexts, are actively being studied. Recently, Xu et al. proposed a new RABE scheme that combines ABE and identity-based encryption (IBE) schemes to efficiently handle ciphertext update and user revocation functionality. In this paper, we show that there is a serious security problem in Xu et al.'s RABE scheme such that a cloud server can obtain the plaintext information of stored ciphertexts by gathering invalidated credentials of revoked users. Additionally, we also show that the RABE scheme of Xu et al. can be secure in a weaker security model where the cloud server cannot obtain any invalidated credentials of revoked users.

INDEX TERMS Cloud storage, access control, attribute-based encryption, revocation, ciphertext update.

I. INTRODUCTION

Cloud storage is a computing paradigm that stores data in a centralized cloud and allows users to access these data anytime anywhere on the Internet using simple client devices. The main advantages of cloud storage include flexible accessibility, ease management, and cost savings. Despite these advantages, cloud storage is inevitably experiencing a variety of security issues because it stores data in an external cloud storage that is outside of the control of a data owner. The key reason that the cloud storage security differs from the existing computer server security is that a cloud server that provides the cloud storage service is not fully trusted so that the cloud server can access the stored data and leak the sensitive information [1], [2].

The easiest way to keep users data secure in cloud storage is to encrypt the data and store it in the cloud. In this case, in order to share the encrypted data with many users, it is needed to effectively control access to the encrypted data according to the authority of the dynamically changing user. That is, the cloud storage system needs to revoke some users whose credentials are no longer valid so that revoked users cannot access data. In addition, the cloud storage system should be able to prevent previously revoked users to gain access to encrypted data that were created long ago by using their old private keys after colluding with the cloud server.

To solve these problems in cloud storage, we can use attribute-based encryption (ABE), which provides access control to ciphertexts. Boldyreva *et al.* [3] proposed a revocable ABE (RABE) scheme that extends the ABE scheme by providing the ability to revoke a user's private key. Sahai *et al.* [4] proposed a revocable-storage ABE (RS-ABE) scheme by extending the concept of RABE that provides the ciphertext update functionality to prevent previously revoked users from accessing previously created ciphertexts in cloud storage. After that, Lee *et al.* [5] proposed efficient RS-ABE schemes that can update ciphertexts more efficiently by combining a self-updatable encryption (SUE) scheme and an

The associate editor coordinating the review of this manuscript and approving it for publication was Junggab Son.

ABE scheme. Therefore, RS-ABE schemes, which provide user revocation and ciphertext update, can be a solution to the problem of cloud storage described above. Recently, Xu *et al.* [6] proposed an RABE scheme that combines an ABE scheme with an IBE scheme by introducing new time encoding functions to efficiently support ciphertext update than the existing RS-ABE schemes. Compared with the most efficient RS-ABE scheme of Lee et al., the RABE scheme of Xu et al. is more efficient in terms of ciphertext size and update key size.

In this paper, we show that it is possible to break the security of the RABE scheme of Xu *et al.* [6]. A key feature of cloud storage is that a cloud server is not fully trusted [2]. In other words, the cloud server faithfully performs the tasks requested by users, but is curious about the information of the users' data. Thus the cloud server should also be considered as an inside attacker. However, Xu et al. have overlooked that the cloud server can be this type of attackers. Suppose that a cloud server is an inside attacker in the RABE scheme of Xu et al. Then the cloud server first gathers revoked credentials (revoked private keys) of users. Note that these revoked credentials are usually no harm to the system since they are safely disabled in publicly broadcasted key updates. Next, the cloud server derives a new ciphertext associated with past time from stored original ciphertexts in cloud storage without modifying the original ciphertexts. Then the cloud server can sufficiently decrypt the derived ciphertext with past time by using the revoked credentials and publicly available key updates. Thus the RABE scheme of Xu et al. cannot be secure against this cloud server.

The organization of this paper is as follows: In Section 2, we first review the RABE scheme of Xu et al. and their security model. Then, in Section 3, we discuss our ciphertext outdate attack to the RABE scheme of Xu et al. by exploiting the time encoding functions of Xu et al. Finally, we conclude in Section 4.

II. REVOCABLE ATTRIBUTE-BASED ENCRYPTION

In this section, we review the RABE scheme of Xu *et al.* [6] and the security model of their RABE.

A. XU ET AL.'S CONSTRUCTION

Before explaining the RABE scheme of Xu *et al.* [6], we first define the time encoding functions proposed by them. The **TEncode** function converts a time period t to a bit string bt of $\log_2 \mathcal{T}$ length by appending zero value to the prefix of the bit string. The **CTEncode** function converts a time period t to an encoded bit string et by finding the first zero value and then converts all remaining values to zero. The definitions of these two time encoding functions are described follows:

TEncode$(t, \mathcal{T})$**:** It takes a decimal number t. It encodes t to a bit string bt. While $|bt| < \log_2 \mathcal{T}$, it performs $bt = 0 \| bt$. It returns the bit string bt.

CTEncode$(t, \mathcal{T})$**:** It takes a decimal number t. Let $[k]$ be the set $\{1, 2, \ldots, k\}$. It first sets an encoded string et as empty one. It next obtains a bit string bt by running **TEncode**$(t, \mathcal{T})$ and sets $chk = false$. For each $j \in [\log_2 \mathcal{T}]$, it performs the following steps: if $bt[j] = 1$ and $chk = false$, then it sets $et[j] = 1$; otherwise it sets $chk = true$ and $et[j] = 0$. It returns the encoded string et.

For example, we let the maximum time is $\mathcal{T} = 2^5$, and two time periods are $t = 5$ and $t^* = 7$. In this case, the function **TEncode**$(t = 5, \mathcal{T})$ returns $bt = 00101$, the function **TEncode**$(t^* = 7, \mathcal{T})$ returns $bt^* = 00111$, and the function **CTEncode**$(t^* = 7, \mathcal{T})$ returns $et^* = 00000$.

The RABE scheme of Xu et al. follows the existing design methodology of previous RABE schemes that combines an ABE scheme, a tree-based broadcast scheme, and an IBE scheme in bilinear groups. In addition, Xu et al. have changed the structure of ciphertext to provide ciphertext update functionality by devising a new ciphertext encoding method. To use a tree-based broadcast scheme, two additional functions **Path** and **KUNodes** should be defined. The function **Path** returns a set of nodes in a binary tree that are in the path from the root node to the specified leaf node, and the function **KUNodes** returns a set of nodes that are root nodes of sub-trees where the leaf nodes of all sub-trees can cover the set of all non-revoked leaf nodes in the binary tree. It is required that if a leaf node v is not revoked then **Path**$(v) \cap$**KUNode**$(RL) \neq \emptyset$ and if v is revoked then **Path**$(v) \cap$**KUNode**$(RL) = \emptyset$ where RL is the set of revoked leaf nodes. We omit the detailed descriptions of **Path** and **KUNodes** functions. For the more detailed definition of these functions, see the work of Boldyreva *et al.* [3]. The RABE scheme of Xu et al. is described as follows:

Setup$(1^\lambda, \mathcal{N}, \mathcal{T}, n)$**:** Let λ be the security parameter, $\mathcal{N}$ be the maximum number of users, $\mathcal{T}$ be the bounded system life time, and n be the maximum number of attributes. It obtains a bilinear group $(p, \mathbb{G}, \mathbb{G}_T, e)$ by running $\mathcal{G}(1^\lambda)$ where p is prime order of the groups. Let g be a generator of $\mathbb{G}$.

It selects a random exponent α and sets $g_1 = g^\alpha$. It also chooses random elements g_2, $\{T_i\}_{i \in [n+1]}$, $U_0, \{U_j\}_{j \in [\log_2 \mathcal{T}]} \in \mathbb{G}$ and defines $T(x) = g_2^{x^n} \prod_{i=1}^{n+1} T_i^{\Delta_{i,[n+1]}(x)}$ where $\Delta_{i,J}(x) = \prod_{j \in J, j \neq i} \frac{x-j}{i-j}$.

It sets a binary tree BT with at least $\mathcal{N}$ number of leaves. Finally, it outputs a revocation list $RL = \emptyset$, a state $ST = BT$, a master key $MK = \alpha$, and public parameters $PP = \big((p, \mathbb{G}, \mathbb{G}_T, e), g, g_1, g_2, \{T_i\}_{i \in [n+1]}, U_0, \{U_j\}_{j \in [\log_2 \mathcal{T}]}\big)$.

GenKey$(id, \mathbb{A}, MK, ST, PP)$**:** Let id be an identity and $\mathbb{A} = (M, \rho)$ be an access policy for attributes where M is a $d \times \ell$ matrix. It assigns the user identity id to a leaf node $\theta \in BT$.

For each node $x \in$**Path**(θ), it performs the following steps: 1) It fetches α_x from the node x. If α_x is not defined before, then it chooses a random $\alpha_x \in \mathbb{Z}_p$ and stores it in the node x. 2) Let $\vec{u}$ be a random ℓ dimensional vector over $\mathbb{Z}_p$ such that $1 \cdot \vec{u} = \alpha_x$. For each row i in the matrix M, it chooses a random exponent

r_i and sets a partial private key $PSK_{id,x} = \big(\{K_{i,0} = g_2^{M_i \cdot \vec{u}} T(i)^{r_i}, K_{i,1} = g^{r_i}\}_{i \in [d]}\big)$.

Finally, it outputs a private key $SK_{id} = \big(\{PSK_{id,x}\}_{x \in Path(\theta)}\big)$ and an updated state $ST = BT$.

UpdateKey(t, RL, MK, ST, PP)**:** Let t be a revocation epoch and RL be the revocation list. It obtains a bit string bt by running **TEncode**$(t, \mathcal{T})$. Let $\mathcal{V}_{bt}$ be the set of all j for which $bt[j] = 0$.

For each node $x \in$**KUNodes**(BT, RL, t), it performs the following steps: 1) It fetches α_x from the node x. If α_x is not defined before, then it chooses a random $\alpha_x \in \mathbb{Z}_p$ and stores it in the node x. 2) It chooses a random exponent r and obtains a partial key update $PKU_{t,x} = \big(U_0 = g_2^{\alpha - \alpha_x}(U_0 \prod_{j \in \mathcal{V}_{bt}} U_j)^r, U_1 = g^r\big)$.

Finally, it outputs a key update $KU_t = \big(\{PUK_{t,x}\}_{x \in KUNodes(BT,RL,t)}\big)$.

DeriveDK(SK_{id}, KU_t, PP)**:** Let $SK_{id} = (\{PSK_{id,x}\}_{x \in Path(\theta)})$ and $KU_t = (\{PKU_{t,x}\}_{x \in KUNodes(BT,RL,t)})$.

If $Path(\theta) \cap KUNodes(BT, RL, t) = \emptyset$, then it outputs $\perp$. Otherwise, it finds a unique node $x \in$**Path**$(\theta) \cap$**KUNodes**(BT, RL, t) and retrieves $PSK_{id,x}$ and $PKU_{t,x}$ for the node x from SK_{id} and KU_t respectively.

Finally it outputs a decryption key $DK_{id,t} = \big(\{PSK_{id,x}, PKU_{t,x}\}\big)$.

Encrypt(S, t, m, PP)**:** Let S be an attribute set, t be time, and m be a message. It obtains an encoded string et by running **CTEncode**$(t, \mathcal{T})$. Let $\mathcal{V}_{et}$ be the set of all j for which $et[j] = 0$.

It chooses a random exponent $s \in \mathbb{Z}_p$ and outputs an original ciphertext $CT_t = \big(C = e(g_1, g_2)^s \cdot m, C_1 = g^s, \{C_{2,i} = T(i)^s\}_{\rho(i) \in S}, E_1 = U_0^s, \{E_{2,j} = U_j^s\}_{j \in \mathcal{V}_{et}}\big)$.

UpdateCT(CT_t, t', PP): Let $CT_t = (C, C_1, \{C_{2,i}\}, E_1, \{E_{2,j}\}_{j \in \mathcal{V}_{et}})$ be an original ciphertext for time t and t' be update time such that $t \leq t'$.

If $t' < t$, then it returns $\perp$ to indicate that the time t' is invalid. Otherwise, it obtains a bit string bt by running **TEncode**$(t, \mathcal{T})$. It chooses a random exponent $s' \in \mathbb{Z}_p$ for randomization and outputs an updated ciphertext $CT_{t'} = \big(C = C \cdot e(g_1, g_2)^{s'}, C_1 = C_1 \cdot g^{s'}, \{C_{2,i} = C_{2,i} \cdot T(i)^{s'}\}_{\rho(i) \in S}, E_{t'} = (C_1 \prod_{j \in \mathcal{V}_{bt}} C_{2,j}) \cdot (U_0 \prod_{j \in \mathcal{V}_{bt}} U_{2,j})^{s'}\big)$.

Decrypt$(CT_t, DK_{id,t}, PP)$: Let $CT_t = (C, C_1, \{C_{2,i}\}, E_t)$ be an update ciphertext for time t and $DK_{id,t} = (PSK_{id,x}, PKU_{t,x})$ be a decryption key where $PSK_{id,x} = (\{K_{i,0}, K_{i,1}\}_{i \in [d]})$ and $PKU_{t,x} = (U_0, U_1)$.

It computes a first component $A_1 = \prod_{\rho(i) \in S}(e(C_1, K_{i,0}) / e(C_{i,0}, K_{i,1}))^{w_i}$. Next, it computes a second component $A_2 = e(C_1, U_0)/e(E_t, U_1)$.

It outputs a decrypted message m by computing $C/(A_1 \cdot A_2)$.

Revoke(id, t, RL, ST)**:** Let id be an identity and t be revocation time. It adds (id, t) to RL and returns the updated revocation list RL.

A cloud storage system consists of four entities: a trusted center, a cloud server, a data owner, and a data user. The trusted center first runs **Setup** to obtain MK and PP and publishes PP. For each data user, the trusted center runs **GenKey** to generate each private key SK_{id} of each data user id. For each current time epoch t, the trusted center periodically runs **UpdateKey** to obtain a key update KU_t for non-revoked users and publishes KU_t. If a user with id is revoked, then the trusted center runs **Revoke** to add this user to the revoked list. A data owner who has a message m can create an original ciphertext CT_t at time t by running **Encrypt** and then he securely sends CT_t to the cloud server for storing it in cloud storage. If a data user want to access to the ciphertext in the cloud storage on time t', then the cloud server first computes an updated ciphertext $CT_{t'}$ by running **UpdateCT** on the original ciphertext and gives the updated ciphertext to the data user. Next, the data user can decrypt the ciphertext $CT_{t'}$ by running **Decrypt** if he has a private key and his private key SK_{id} is not yet revoked in a key update $KU_{t'}$ on time t'.

B. SECURITY MODEL

We describe the security model of the RABE scheme as defined by Xu *et al.* [6]. The selective IND-RABE-CPA security is defined as the following game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$:

Init: $\mathcal{A}$ first submits a challenge attribute set S^*.

Setup: $\mathcal{C}$ generates an empty revocation list rl, a state ST, a master key MK, and public parameters PP by running the setup algorithm **Setup**$(\lambda, \mathcal{N}, \mathcal{T}, n)$, and then it gives PP to $\mathcal{A}$.

Phase 1: $\mathcal{A}$ may adaptively request private key, key update, and revocation queries to the following oracles.

- The private key generation oracle takes an identity id and an access structure $\mathbb{A}$ as input, and returns a private key SK_{id} by running **GenKey**$(id, \mathbb{A}, MK, ST, PP)$.
- The key update oracle takes time t as input, and returns a key update KU_t by running **UpdateKey**(t, RL, MK, ST, PP).
- The revocation oracle takes a revoked identity id and time t as input, and updates the revocation list by running **Revoke**(id, t, RL, ST).

Challenge: $\mathcal{A}$ submits challenge time $t^* \in \mathcal{T}$ and two challenge messages m_0^*, m_1^* of the same size with the following constraints:

- If a private key for an identity id and an access structure $\mathbb{A}$ such that $\mathbb{A}(S^*) = 1$ was queried to the private key generation oracle, then the revocation of the identity id must be queried on time t such that $t \leq t^*$ to the revocation oracle.
- If a non-revoked user with the identity id whose access structure $\mathbb{A}$ satisfies the challenge attribute set S^*, then id should not be previously queried to the private key generation oracle.

$\mathcal{C}$ flips a random bit $b \in \{0, 1\}$ and creates a challenge ciphertext CT^* by running **Encrypt**(S^*, t^*, m_b^*, PP).

Phase 2: $\mathcal{A}$ continues to request private key, key update, and revocation queries. $\mathcal{C}$ handles the queries as the same as before and following the restrictions defined in the challenge phase.

Guess: Finally $\mathcal{A}$ outputs a bit b'.

An RABE scheme is selectively IND-RABE-CPA secure if for any probabilistic polynomial time adversary $\mathcal{A}$, the advantage of $\mathcal{A}$ in the above RABE game defined as $\Pr[b = b'] - \frac{1}{2}$ is negligible in the security parameter λ.

III. CIPHERTEXT OUTDATE ATTACK

In this section, we show that there is an effective adversary against the RABE scheme of Xu *et al.* [6]. To do this, we first analyze the properties of two time encoding functions, **TEncode** and **CTEncode**, proposed by Xu et al. through the following two lemmas. The key to the following two lemmas is that a challenge original ciphertext associated with challenge time t^* can be changed to a ciphertext element associated with the past time t.

Lemma 1: Let st be a bit string in $\{0, 1\}^{\log_2 \mathcal{T}}$ *and* $\mathcal{V}_{st}$ *be the set of all j such that* $st[j] = 0$. *There exist time periods* $t, t^* \in \mathcal{T}$ *such that* $t < t^*$ *and* $\mathcal{V}_{bt} \subseteq \mathcal{V}_{et^*}$ *where bt is obtained from TEncode*$(t, \mathcal{T})$ *and* et^* *is obtained from CTEncode*$(t^*, \mathcal{T})$.

Proof: For the notational simplicity, we set $\mathcal{T} = 2^\tau$. To prove this lemma, we first randomly choose time periods t, t^* satisfying $0 < t < t^* < 2^{\tau-1}$. Then, we run **TEncode**$(t, \mathcal{T})$ to get a bit string $bt \in \{0, 1\}^\tau$ and **TEncode**$(t^*, \mathcal{T})$ to get another bit string $bt^* \in \{0, 1\}^\tau$. Since the time periods t and t^* are smaller than $2^{\tau-1}$, the first bit value $bt[1]$ and $bt^*[1]$ of the two bit strings bt and bt^* have the same bit 0. Now let's analyze the bit string et^* obtained by running **CTEncode**$(t^*, \mathcal{T})$. In the **CTEncode** algorithm, the algorithm finds the first position with a bit value of 0 in the bt^* bit string and then sets all subsequent bit values to a value of zero. Thus, the resulting bit string et^* becomes a bit string with 0 value in all positions since $bt^*[1] = 0$ is already fixed. Therefore, the set $\mathcal{V}_{et^*}$ consists of $\{1, 2, \ldots, \tau\}$ and the set of $\mathcal{V}_{bt}$ should be a subset of $\{1, 2, \ldots, \tau\}$ since $0 < t$. □

As an example, let us look at the encoding results for two time periods $t = 5$ and $t^* = 7$ when the maximum time is $\mathcal{T} = 2^5$. Since $\mathcal{T} = 2^5$, the function **TEncode**$(t = 5, \mathcal{T})$ returns a bit string $bt = 00101$, the function **TEncode**$(t^* = 7, \mathcal{T})$ returns a bit string $bt^* = 00111$, and the function **CTEncode**$(t^* = 7, \mathcal{T})$ returns the bit string $et^* = 00000$. Because $t = 5 < t^* = 7$ and $\mathcal{V}_{bt} = \{1, 2, 4\} \subseteq \mathcal{V}_{et^*} = \{1, 2, 3, 4, 5\}$, we can easily show that two time periods $t = 5$ and $t^* = 7$ are one example of Lemma 1.

Lemma 2: If there exist time periods $t, t^* \in \mathcal{T}$ *such that* $t < t^*$ *and* $\mathcal{V}_{bt} \subseteq \mathcal{V}_{et^*}$ *where bt is obtained from* ***TEncode***$(t, \mathcal{T})$ *and* et^* *is obtained from* ***CTEncode***$(t^*, \mathcal{T})$*, then a ciphertext element* E_t *for time t can be derived from an original ciphertext* CT^* *for time* t^*.

Proof: As the same as in Lemma 1, we randomly choose time periods t and t^* to satisfy $0 < t < t^* < 2^{\tau-1}$. In the RABE scheme of Xu et al., the original ciphertext CT^* for the time t^* includes ciphertext elements E_1 and $E_{2,j}$ for all $j \in \mathcal{V}_{et^*}$. As shown in the previous Lemma 1, the set $\mathcal{V}_{et^*}$ is defined as a set of all indices from 1 to τ since all bit values of et^* are composed of 0. In the description of the **UpdateCT** algorithm, the ciphertext element E_t can be derived by composing the elements E_1 and $E_{2,j}$ for all $j \in \mathcal{V}_{bt}$ of CT^*. Therefore, it is possible to construct the element E_t since $\mathcal{V}_{bt} \subseteq \mathcal{V}_{et^*}$ is satisfied by Lemma 1. □

By using the previous two lemmas, we show that a cloud server which stores original ciphertexts generated by data owner can obtain sensitive information of the original ciphertexts by gathering revoked credentials of revoked users.

Theorem 3: There exists a probabilistic polynomial-time adversary that can break the selective IND-RABE-CPA security of Xu et al.'s RABE scheme.

Proof: The main idea of our attack is for an inside adversary, which is a cloud server, to derive an outdated ciphertext CT_t of past time t from the original challenge ciphertext CT^* of challenge time t^* such that $t < t^*$. If the adversary gathers revoked credentials of revoked users from the Internet, then it can decrypt the outdated ciphertext by combining the revoked credential with publicly available key updates.

A detailed adversary algorithm $\mathcal{A}$ breaking the RABE scheme of Xu et al. is described as follows:

1) Initially $\mathcal{A}$ submits a challenge attribute set S^* and receives public parameters PP.
2) After that, $\mathcal{A}$ gathers revoked credentials (private keys) of revoked users from the Internet. Note that revoked credentials of users can be available in Internet since they are safely revoked by the system although these credentials are intensionally revealed by an hacker or accidently revealed by a honest user.
   Let SK_{id^*} be one of the obtained revoked private keys with an identity id^* and an access policy $\mathbb{A}$ that satisfies $\mathbb{A}(S^*) = 1$. Because this private key SK_{id^*} is revoked by a trusted center, we have that it is revoked on time t^* in a key update KU_{t^*}, but not yet revoked in a key update KU_t such that $t < t^*$. $\mathcal{A}$ sets past time t and challenge time t^* that satisfies the condition described in Lemma 1.
3) Next $\mathcal{A}$ also gathers key updates from the Internet since each key update KU_t is publicly broadcasted by the trusted center per each time period t. It now derive a decryption key $DK_{id^*,t}$ by combining the gathered private key SK_{id^*} and the gathered key update KU_t since SK_{id^*} was not yet revoked on past time $t < t^*$. Note that it cannot derive DK_{id^*,t^*} for the challenge time t^* because SK_{id^*} was already revoked on time t^*.
4) In the challenge step, $\mathcal{A}$ submits the challenge time t^*, randomly chosen two challenge messages m_0^*, m_1^*, and receives a challenge original ciphertext CT^*.
   Let $CT^* = (C, C_1, \{C_{2,i}\}, E_1, \{E_{2,j}\}_{j \in \mathcal{V}_{et^*}})$ be the original ciphertext on the time t^* where et^* is obtained from **CTEncode**$(t^*, \mathcal{T})$. Consider the set $\mathcal{V}_{bt}$ for the fixed past time t. The ciphertext element E_t can be derived

from the original ciphertext CT^* by Lemma 2 because $\mathcal{V}_{bt} \subseteq \mathcal{V}_{et^*}$ is satisfied. Thus $\mathcal{A}$ can easily derive an outdated ciphertext $CT_t = (C, C_1, C_{2,i}, E_t)$ associated with the challenge attribute set S^* and the past time t by performing re-randomization.

5) Finally, $\mathcal{A}$ obtains the message m^* by decrypting CT_t using $DK_{id^*,t}$ and outputs a bit b' by comparing m^* with the challenge messages.

Now we analyze the success probability of the adversary $\mathcal{A}$ described above. As shown above, the decryption succeeds because the behavior of $\mathcal{A}$ satisfy the constraints of the security model and the outdated ciphertext is also a valid ciphertext with the correct distribution. Therefore, $\mathcal{A}$ wins the RABE security game since the advantage of $\mathcal{A}$ is $1/2$. $\square$

A. DISCUSSIONS

In this section, we consider two possible defences against the ciphertext outdate attacks.

1) WEAKER SECURITY MODEL

The reason for the above attack is that a cloud server can decrypt a stored original ciphertext by deriving an outdated ciphertext from the original ciphertext and using gathering revoked credentials of revoked users. A naive approach to prevent this devastating attack is to weaken the security model of RABE for cloud storage. In other words, if an inside attacker such as a cloud server performs an attack, then the inside attacker should be prohibited to gathering revoked credentials of revoked users from the Internet. However, this weaker security model only provides a limited security since it is very hard to forbid the inside attacker to (passively) gathering some useful information from the Internet [7].

2) SELF-UPDATABLE ENCRYPTION

Sahai *et al.* [4] first devised a new encryption scheme that supports ciphertext updates in cloud storage by using the delegation feature of ABE. After that, Lee *et al.* [5] proposed a self-updatable encryption scheme by combining a binary tree with the key delegation feature of hierarchical identity-based encryption. A self-updatable encryption scheme provides the most efficient ciphertext update and it is proven to be secure against collusion attacks. In order to enhance the security of Xu *et al.*'s RABE scheme, their RABE scheme can be modified to use a self-updatable encryption scheme instead of using time encoding functions. In addition, if a self-updatable encryption scheme is used, a data owner does not need to use a secure channel when he sends a ciphertext to cloud storage.

IV. CONCLUSION

In this paper, we showed that a cloud server can perform the ciphertext outdate attack to the RABE scheme of Xu *et al.* This attack was possible because the cloud server is not a fully trusted entity and it can derive another ciphertext associated with past time from the original ciphertext stored by a user. Although Xu *et al.* showed that their RABE scheme is secure against outside attackers, they didn't consider the cloud server can be an inside attacker in the security proof. One naive defence against this attack is to consider a weaker security model where an insider attacker who has access to the original ciphertext cannot obtain revoked credentials of users by preventing the inside attacker from accessing to the Internet. A better defence for this attack is to use the self-updatable encryption scheme of Lee *et al.* [5] for efficient ciphertext updating since an RABE scheme that uses a self-updatable encryption scheme can be secure against collusion attacks.

REFERENCES

[1] S. Kamara and K. Lauter, "Cryptographic cloud storage," in *Financial Cryptography and Data Security*, vol. 6054. Berlin, Germany: Springer, 2010, pp. 136–149.

[2] M. D. Ryan, "Cloud computing security: The scientific challenge, and a survey of solutions," *J. Syst. Softw.*, vol. 86, no. 9, pp. 2263–2268, Sep. 2013.

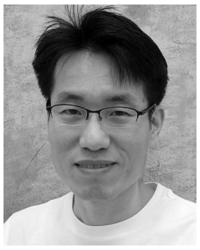
[3] A. Boldyreva, V. Goyal, and V. Kumar, "Identity-based encryption with efficient revocation," in *Proc. ACM Conf. Comput. Commun. Secur.*, Oct. 2008, pp. 417–426.

[4] A. Sahai, H. Seyalioglu, and B. Waters, "Dynamic credentials and ciphertext delegation for attribute-based encryption," in *Advances in Cryptology—CRYPTO 2012*, vol. 7417. Berlin, Germany: Springer, 2012, pp. 199–217.

[5] K. Lee, S. G. Choi, D. H. Lee, J. H. Park, and M. Yung, "Self-updatable encryption: Time constrained access control with hidden attributes and better efficiency," in *Advances in Cryptology—ASIACRYPT 2013*, vol. 8269. Berlin, Germany: Springer, 2013, pp. 235–254.

[6] S. Xu, G. Yang, Y. Mu, and R. H. Deng, "Secure fine-grained access control and data sharing for dynamic groups in the cloud," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 8, pp. 2101–2113, Aug. 2018.

[7] R. A. Popa, C. M. S. Redfield, N. Zeldovich, and H. Balakrishnan, "CryptDB: Protecting confidentiality with encrypted query processing," in *Proc. SOSP*, Oct. 2011, pp. 85–100.



KWANGSU LEE received the B.S. degree in computer science from Yonsei University, South Korea, in 1998, the M.S. degree in computer science from KAIST, South Korea, in 2000, and the Ph.D. degree in information security from Korea University, in 2011. He is currently an Associate Professor with the Department of Computer and Information Security, Sejong University, Seoul, South Korea. His research interests include public-key cryptography, functional encryption, cryptographic protocols, provable security, and post-quantum cryptography.