Two-Factor Data Security Protection Mechanism for Cloud Storage System

Joseph K. Liu, Kaitai Liang, Willy Susilo, Jianghua Liu, and Yang Xiang, *Senior Member, IEEE*

Abstract—In this paper, we propose a two-factor data security protection mechanism with factor revocability for cloud storage system. Our system allows a sender to send an encrypted message to a receiver through a cloud storage server. The sender only needs to know the identity of the receiver but no other information (such as its public key or its certificate). The receiver needs to possess two things in order to decrypt the ciphertext. The first thing is his/her secret key stored in the computer. The second thing is a unique personal security device which connects to the computer. It is impossible to decrypt the ciphertext without either piece. More importantly, once the security device is stolen or lost, this device is revoked. It cannot be used to decrypt any ciphertext. This can be done by the cloud server which will immediately execute some algorithms to change the existing ciphertext to be un-decryptable by this device. This process is completely transparent to the sender. Furthermore, the cloud server cannot decrypt any ciphertext at any time. The security and efficiency analysis show that our system is not only secure but also practical.

Index Terms—Two-factor, factor revocability, security, cloud storage

1 Introduction

Cloud storage [10], [43], [44], [45], [49] is a model of networked storage system where data is stored in pools of storage which are generally hosted by third parties. There are many benefits to use cloud storage. The most notable is data accessibility. Data stored in the cloud can be accessed at any time from any place as long as there is network access. Storage maintenance tasks, such as purchasing additional storage capacity, can be offloaded to the responsibility of a service provider. Another advantage of cloud storage is data sharing between users. If Alice wants to share a piece of data (e.g., a video) to Bob, it may be difficult for her to send it by email due to the size of data. Instead, Alice uploads the file to a cloud storage system so that Bob can download it at anytime.

Despite its advantages, outsourcing data storage also increases the attack surface area at the same time. For example, when data is distributed, the more locations it is stored the higher risk it contains for unauthorized physical access to the data. By sharing storage and networks with many other users it is also possible for other unauthorized users to access your data. This may be due to mistaken actions, faulty equipment, or sometimes because of criminal intent. A promising solution to offset the risk is to deploy encryption technology. Encryption can protect data as it is being transmitted to and from the cloud service. It can further protect data that is stored at the service provider. Even there is an unauthorized adversary who has gained access to the cloud, as the data has been encrypted, the adversary cannot get any information about the plaintext. Asymmetric encryption allows the encryptor to use only the public information (e.g., public key or identity of the receiver) to generate a ciphertext while the receiver uses his/her own secret key to decrypt. This is the most convenient mode of encryption for data transition, due to the elimination of key management existed in symmetric encryption.

Enhanced Security Protection. In a normal asymmetric encryption, there is a single secret key corresponding to a public key or an identity. The decryption of ciphertext only requires this key. The key is usually stored inside either a personal computer or a trusted server, and may be protected by a password. The security protection is sufficient if the computer/server is isolated from an opening network. Unfortunately, this is not what happens in the real life. When being connected with the world through the Internet, the computer/server may suffer from a potential risk that hackers may intrude into it to compromise the secret key without letting the key owner know. In the physical security aspect, the computer storing a user decryption key may be used by another user when the original computer user (i.e. the key owner) is away (e.g., when the user goes to toilet for a while without locking the machine). In an enterprise or college, the sharing usage of computers is also common. For example, in a college, a public computer in a copier room will be shared with all students staying at the same floor. In these cases, the secret key can be compromised by some attackers who can access the victim's personal data stored in the cloud system. Therefore, there exists a need to enhance the security protection.

- *J.K. Liu is with Monash University, Australia. E-mail: joseph.liu@monash.edu.*
- *K. Liang is with Aalto University, Finland. E-mail: kaitai.liang@aalto.fi.*
- *W. Susilo is with the University of Wollongong, Australia. E-mail: wsusilo@uow.edu.au.*
- *J. Liu is with the Fujian Provincial Key Laboratory of Network Security and Cryptology, School of Mathematics and Computer Science, Fujian Normal University, China. E-mail: jianghualiu11@gmail.com.*
- *Y. Xiang is with the School of Information Technology, Deakin University, Australia. E-mail: yang@deakin.edu.au.*

Manuscript received 28 Sept. 2014; revised 26 Apr. 2015; accepted 16 July 2015. Date of publication 29 July 2015; date of current version 16 May 2016. Recommended for acceptance by H. Jin.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TC.2015.2462840

An analogy is e-banking security. Many e-banking applications require a user to use both a password and a security device (two factors) to login system for money transfer. The security device may display a one-time password to let the user type it into the system, or it may be needed to connect with the computer (e.g., through USB or NFC). The purpose of using two factors is to enhance the security protection for the access control.

As cloud computing becomes more mature and there will be more applications and storage services provided by the cloud, it is easy to foresee that the security for data protection in the cloud should be further enhanced [12], [18], [42], [47]. They will become more sensitive and important, as if the e-banking analogy. Actually, we have noticed that the concept of two-factor encryption, which is one of the encryption trends for data protection,[1] has been spread into some real-world applications, for example, full disk encryption with Ubuntu system, AT&T two factor encryption for Smartphones,[2] electronic vaulting and druva—cloud-based data encryption.[3] However, these applications suffer from a potential risk about factor revocability that may limit their practicability. Note we will explain it later. A flexible and scalable two-factor encryption mechanism is really desirable in the era of cloud computing. That motivates our work.

1.1 Some Naive Approaches

We discuss some naive approaches for enhancement of security protection and explain why they are not the best candidate to achieve the goal of flexibility.

1) *Double encryption*: A security device (with an additional public key or serial number) is still required. The encryption process is executed twice. First encrypt the plaintext corresponding to the public key or identity of the user. Then encrypt it again corresponding to the public key or serial number of the security device. For the decryption stage, the security device first decrypts once. The partially decrypted ciphertext is then passed to the computer which uses the user secret key to further decrypt it. Without either part (user secret key or security device) one cannot decrypt the ciphertext.

   It seems that this naive approach can achieve our goal. However, there exist many practical issues that it cannot solve. For example,

   - If the user has lost his security device, then his/her corresponding ciphertext in the cloud cannot be decrypted forever! That is, the approach cannot support security device update/revocability.
   - The sender needs to know the serial number/public key of the security device, in additional to the user's identity/public key. That makes the encryption process more complicated. In the case of identity-based encryption, the concept of "identity-based" has been totally lost as the sender needs to know not only the identity*but another serial number*!

2) *Split the secret key into two parts*: Another naive way to think of is to simply split the secret key into two parts. The first part is stored in the computer while the second part is embedded into a security device. Similar to the above approach, without either part one cannot decrypt the ciphertext.

   Again it seems that this approach can achieve our goal. However, note that the security of a normal encryption scheme cannot be guaranteed if part of the secret key has been exposed. The security is only guaranteed if the whole secret key has not been exposed to the adversary. In other words, if we simply split the secret key into two parts, the adversary with either part may have non-negligible chance to decrypt (or at least to know some information about the plaintext). This is not the case that we expect.

   There exists another cryptographic primitive called "leakage-resilient encryption" [1], [15], [37]. The security of the scheme is still guaranteed if the leakage of the secret key is up to certain bits such that the knowledge of these bits does not help to recover the whole secret key. However, though using leakage resilient primitive can safeguard the leakage of certain bits, there exists another practical limitation. Suppose we put part of the secret key into the security device. Unfortunately the device is stolen. The user needs to obtain a replacement device so that he can continue to decrypt his corresponding secret key. The trivial way is to copy the same bits (as in the stolen device) to the new device by the private key generator (PKG). This approach can be easily achieved. Nevertheless, there exists security risk. If the adversary (who has stolen the security device) can also break into the computer where the other part of secret key is stored, then it can decrypt all ciphertext corresponding to the victim user. The most secure way is to cease the validity of the stolen security device.

   The same analogy is the online banking. A user needs to have a security device (together with the knowledge of his/her password) in order to login the e-banking service. If the security device is reported as lost, the user can no longer use the old device to login. Thus using leakage resilient primitive cannot provide this security feature which is considered as the most important criterion of two-factor security protection.

3) *Other methods*: Some real-world systems, such as AT&T and druva, also leverage two-factor encryption techniques to protect message from being leaked to malicious users. However, their techniques suffer from a potential practical risk. Below we take druva system as an example. In a druva system, a message is first encrypted under a user key k_1, and next uploaded to a cloud server. The user key k_1 is further encrypted by another user key k_2, and stored in the server as well. The key k_2 is held by the user. When retrieving the message, the user needs to use k_2 to

1. http://www.datamation.com/data-center/trends-in-data-protection-prevention-and-recovery.html
2. http://www.securityweek.com/att-offer-carrier-provided-two-factor-encryption-smartphones
3. http://www.druva.com/

recover k_1 which is further used to recover m. It is undeniable that this message-key-encrypt mechanism is much better than the mode only using a single key to encrypt an outsourced data, and storing the ciphertext along with the key in the server. Nevertheless, this mechanism suffers from a potential risk in practice (which we have mentioned previously): once the user loses the key k_2, all data of the user stored in the cloud cannot be retrieved. The lack of revocability for encryption factor limits the flexibility of the system.

1.2 Our Contributions

In this paper, we propose a novel two-factor security protection mechanism for data stored in the cloud. Our mechanism provides the following nice features:

1) Our system is an IBE (Identity-based encryption)-based mechanism. That is, the sender only needs to know the identity of the receiver in order to send an encrypted data (ciphertext) to him/her. No other information of the receiver (e.g., public key, certificate etc.) is required. Then the sender sends the ciphertext to the cloud where the receiver can download it at anytime.
2) Our system provides two-factor data encryption protection. In order to decrypt the data stored in the cloud, the user needs to possess two things. First, the user needs to have his/her secret key which is stored in the computer. Second, the user needs to have a unique personal security device which will be used to connect to the computer (e.g., USB, Bluetooth and NFC). It is impossible to decrypt the ciphertext without either piece.
3) More importantly, our system, for the first time, provides security device (one of the factors) revocability. Once the security device is stolen or reported as lost, this device is*revoked*. That is, using this device can no longer decrypt any ciphertext (corresponding to the user) in any circumstance. The cloud will immediately execute some algorithms to change the existing ciphertext to be*un-decryptable*by this device. While the user needs to use his new/replacement device (together with his secret key) to decrypt his/her ciphertext. This process is completely transparent to the sender.
4) The cloud server cannot decrypt any ciphertext at any time.

We provide an estimation of the running time of our prototype to show its practicality, using some benchmark results. We also note that although there exist some naive approaches that seem to achieve our goal, we have discussed in Section 1.1 that there are many limitations by each of them and thus we believe our mechanism is the first to achieve all the above mentioned features in the literature.

2 Related Work

We first review some solutions which may contain similar functionalities. We will further explain why they cannot fully achieve our goal.

2.1 Cryptosystems with Two Secret Keys

There are two kinds of cryptosystems that requires two secret keys for decryption. They are certificateless cryptosystem (CLC) and certificate-based cryptosystem.

Certificateless cryptosystem was first introduced in [2] and further improvements can be found in [4], [23], [28]. It combines the merits of identity-based cryptosystem (IBC) and the traditional public-key infrastructure (PKI). In a CLC, a user with an identity chooses his own user secret key and user public key. At the same time the authority (called the Key Generation Centre (KGC)) further generates a partial secret key according to his identity. Encryption or signature verification requires the knowledge of both the public key and the user identity. On the opposite, decryption or signature generation requires the knowledge of both the user secret key and the partial secret key given by the KGC. Different from the traditional PKI, there is no certificate required. Thus the costly certificate validation process can be eliminated. However, the encryptor or the signature verifier still needs to know the user public key. It is less convenient than IBC where only identity is required for encryption or signature verification.

Similar to CLC, another primitive called certificate-based cryptosystem (CBC) was introduced in [19]. Further variants may include [3], [29], [30], [31], [33]. The concept is almost the same as CLC, except that the partial secret key given by the KGC (which is called the *certificate*) is a signature of the identity *and* the public key of the user by the KGC. (Note that in CLC, the partial secret key given by the KGC is just the signature of the identity of the user.) Due to the similarities, CBC faces the same disadvantages as CLC mentioned above.

2.2 Cryptosystems with Online Authority

Mediated cryptography was first introduced in [7] for the purpose of revocation of public keys. It requires an online mediator, referred to a SEcurity Mediator (SEM), for every transaction. The SEM also provides a control of security capabilities. If the SEM does not cooperate then no transactions with the public key are possible any longer. In other words, any revoked user cannot get the cooperation from the SEM. That means revoked users cannot decrypt any ciphertext successfully.

Later on, this notion was further generalized as security mediated certificateless (SMC) cryptography [11], [48]. In a SMC system, a user has a secret key, public key and an identity. The user secret key and the SEM are required to decrypt a ciphertext or sign a message. On the opposite side, the user public key and the corresponding identity are needed for signature verification or encryption. Since the SEM is controlled by the revocation authority, the authority can refuse to provide any cooperation for revoked user so that no revoked user can generate signature or decrypt ciphertext.

Note that SMC is different from our concept. The main purpose of SMC is to solve the revocation problem. Thus the SME is controlled by the authority and it has to be online for every signature signing and ciphertext decryption. Furthermore, it is not identity-based. The encryptor (or signature verifier) needs to know the corresponding

public key in addition to the identity. That makes the system less practical and looses the advantages of using identity-based system.

2.3 Cryptosystem with Security Device

The paradigm of key-insulated cryptography was introduced in [16] and variants were proposed in [17], [22], [25], [32]. There is a physically-secure but computationally-limited device in the system. A long-term key is stored in this device, while a short-term secret key is kept by users on a powerful but insecure device where cryptographic computations take place. Short term secrets are then refreshed at discrete time periods via interaction between the user and the base while the public key remains unchanged throughout the lifetime of the system. The user obtains a partial secret key from the device at the beginning of each time period. He then combines this partial secret key with the one from the previous period, in order to renew the secret key for the current time period.

Different from our concept, key-insulated cryptosystem requires all users to update their key in every time period. It may require some costly time synchronization algorithms between users which may not be practical in many scenarios. The key update process requires the security device. Once the key has been updated, the signing or decryption algorithm does *not* require the device anymore within the same time period. While our concept *does* require the security device every time the user tries to decrypt the ciphertext. Furthermore, there is no key updating required in our system. Thus we do not require any synchronization within the whole system.

2.4 Cryptosystem with Revocability

Since our system is an IBE-based mechanism, we below introduce IBE-based systems supporting revocability. The first revocable IBE is proposed by Boneh and Franklin [8], in which a ciphertext is encrypted under an identity id and a time period T, and a non-revoked user is issued a private key $sk_{id,T}$ by a PKG such that the user can access the data in T. Boldyreva, Goyal and Kumar [6] proposed the security notion for revocable IBE. To achieve adaptive security, Libert and Vergnaud [26] proposed a revocable IBE scheme based on the combination of attribute-based encryption and IBE. Recently, Seo and Emura [39] formalized a revised notion for revocable IBE. Since its introduction, there are many variants of revocable IBE, such as [38]. The premise of a revocable IBE system is mainly related to a time period: next the decryption rights of the next time period relies on a secret token (for the next time period) issued by PKG and a current time period key. However, this premise yields inconvenience once the current time period key is lost.

Another cryptosystem supporting revocability is proxy re-encryption (PRE). Decryption rights delegation is introduced in [35]. Blaze, Bleumer and Strauss [5] formally defined the notion of PRE. To employ PRE in the IBE setting, Green and Ateniese [20] defined the notion of identity-based PRE (IB-PRE). Later on, Tang, Hartel and Jonker [41] proposed a CPA-secure IB-PRE scheme, in which delegator and delegatee can belong to different domains. After that there are many IB-PRE systems have been proposed to support different user requirements, e.g., [13], [24], [34], [36], [40]. Among of the previously introduced IB-PRE systems, [20] is the most efficient one without loss of revocability. We state that leveraging [20] can only achieve one of our design goals, revocability, but not two-factor protection.

3 Overview

3.1 Our Intuition

Inspired by [21], we propose a two-factor data security protection mechanism. Before giving the description of our mechanism, we first give an intuition on it. In our system, we have the following entities:

- Private key generator: It is a trusted party responsible for issuing private key of every user.
- Security device issuer (SDI): It is a trusted party responsible for issuing security device of every user.
- Sender (*Alice*): She is the sender (and the creator) of the ciphertext. She only knows the identity (e.g., e-mail address) of the receiver but nothing else related to the receiver. After she has created the ciphertext, she sends to the cloud server to let the receiver for download.
- Receiver (*Bob*): He is the receiver of the ciphertext and has a unique identity (e.g., email address). The ciphertext is stored on a cloud storage while he can download it for decryption. He has a private key (stored in his computer) and a security device (that contains some secret information related to his identity). They are given by the PKG. The decryption of ciphertext requires both the private key and the security device.
- Cloud server: The cloud server is responsible for storing all ciphertext (for receiver to download). Once a user has reported lost of his security device (and has obtained a new one from the PKG), the cloud acts as a proxy to re-encrypt all his past and future ciphertext corresponding to the new device. That is, the old device is revoked.

We further illustrate our mechanism's framework in Figs. 1 and 2. When a new system user, say Bob, joins our system, a PKG will issue a private key, and SDI will issue a security device to him. Both the private key and the security device are necessary for recovering a data from its encrypted format.

In ordinary data sharing, a data sender, say Alice, first encrypts the sharing data under the identity of a data receiver, say Bob, and next uploads the ciphertext to the cloud server. Here we refer to this ciphertext as first-level ciphertext. After receiving the first-level ciphertext from Alice, the cloud server then turns the ciphertext to become a second-level ciphertext for the corresponding security device belonging to Bob. Bob then downloads the second-level ciphertext from the cloud, and next recovers the data from its encrypted form by using his private key and security device.

When the security device of Bob is either lost or stolen, Bob first reports the issue to the SDI. The SDI then issues a

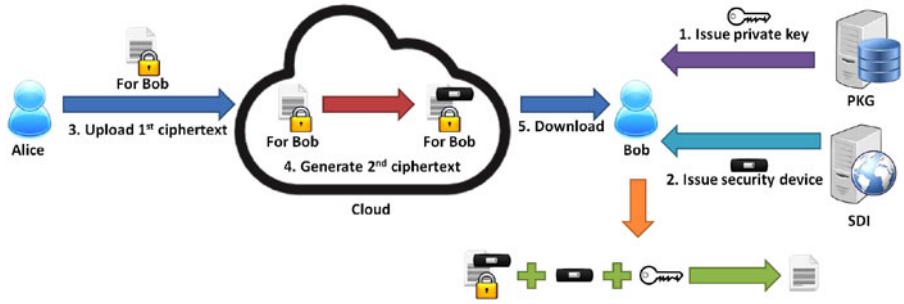


Fig. 1. Ordinary data sharing.

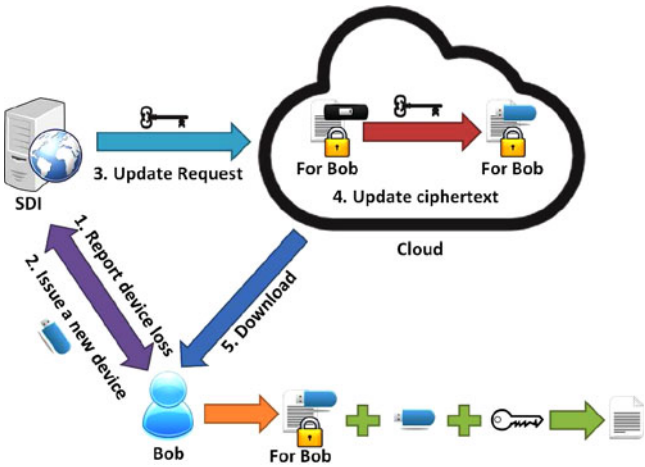


Fig. 2. Update ciphertext after issuing a new security device.

new security device to Bob, and meanwhile, it sends a request of updating Bob's corresponding ciphertext along with a special key to the cloud server. The cloud server updates the ciphertexts of Bob under an old security device to the ones under a new device. However, it does not gain access to the underlying data in the update process. Here Bob is allowed to download and recover the data by using his private key and new security device.

3.2 Assumptions

In our system, we assume that the PKG is a trusted party.[4] We further assume that the SDI is trusted, and the cloud service is semi-trust. That is, it is honest that it will execute all prescribed algorithms but it is curious. It may try to decrypt the ciphertext stored in the cloud storage. We also assume that an honest system user will not expose his/her security device and secret key to an adversary, and the security device is temper resistant.

3.3 Threat Model

In this paper, we consider the following threats:

1) **Type-I:** *Decrypt without security device*: The adversary tries to decrypt the ciphertext without the security device, or using a revoked security device, or using another security device belonging to others. It can have its own secret key.

2) **Type-II:** *Decrypt without secret key*: The adversary tries to decrypt the ciphertext without any secret key. It can have its own security device.

Note that the above threat model has already captured the semi-trust behaviour of the cloud server.

3.4 Notations

In Table 1, we introduce the notations used in our system.

4 Details of Our Proposed Mechanism

4.1 Mathematical Preliminaries

Bilinear maps. Let $BSetup$ denote an algorithm that, on input the security parameter k, outputs the parameters for a bilinear map as $(q, g, \mathbb{G}, \mathbb{G}_T, e)$, where $\mathbb{G}$ and $\mathbb{G}_T$ are two multiplicative cyclic groups with prime order $q \in \Theta(2^k)$ and g is a generator of $\mathbb{G}$. The efficient mapping $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ has three properties: (1) *Bilinearity*: for all $g \in \mathbb{G}$ and $a, b \in_R \mathbb{Z}_q^*$, $e(g^a, g^b) = e(g, g)^{ab}$; (2) *Non-degeneracy*: $e(g, g) \neq 1_{\mathbb{G}_T}$, where $1_{\mathbb{G}_T}$ is the unit of $\mathbb{G}_T$; (3) *Computability*: e can be efficiently computed.

TABLE 1
Frequently Used Notations

$\mathbb{Z}_q^*$	all positive integers (except 0) after module a prime q
$\oplus$	exclusive OR
$r \in_R \mathbb{Z}_q^*$	randomly choose an r from $\mathbb{Z}_q^*$
$\{0,1\}$	a bit with value either 0 or 1
$A\|\|B$	string A concatenates string B
ID_i	the identity of user i
tpk_i	the public information of the (old) security device of a user ID_i
tsk_i	the secret information embedded in the (old) security device of a user ID_i
$\widetilde{tpk_i}$	the public information of the new security device of a user ID_i
sk_{ID_i}	the secret key of a user ID_i
m	a message
C_1	the first-level ciphertext of a message m
C_2	the second-level ciphertext of a message m
$rk_{tpk_i \rightarrow \widetilde{tpk_i}}$	the information used to update a second-level ciphertext under an old security device to another (second-level) ciphertext under a new security device
$C_2^{(up)}$	the updated second-level ciphertext of a message m under a new security device

4. This is a normal assumption in all identity-based systems.

Decisional Bilinear Diffie-Hellman (BDH) Assumption [46]. For an algorithm $\mathcal{A}$, define its advantage as $Adv_{\mathcal{A}}^{BDH}(k) = |Pr[\mathcal{A}(g, g^a, g^b, g^c, e(g,g)^{abc}) = 1] - Pr[\mathcal{A}(g, g^a, g^b, g^c, e(g,g)^z) = 1]|$, where $a, b, c, z \in_R \mathbb{Z}_q^*$. We say the BDH assumption holds, if for any PPT algorithm $\mathcal{A}$, $Adv_{\mathcal{A}}^{BDH}(k)$ is negligible in k.

q-weak Decision Bilinear Diffie-Hellman Inversion (q-wDBDHI) Assumption. For an algorithm $\mathcal{A}$, define its advantage as $Adv_{\mathcal{A}}^{q-wDBDHI}(k) = |Pr[\mathcal{A}(g, g^a, \ldots, g^{a^q}, g^b, e(g,g)^{b/a}) = 1] - Pr[\mathcal{A}(g, g^a, \ldots, g^{a^q}, g^b, e(g,g)^z) = 1]|$, where $a, b, z \in_R \mathbb{Z}_q^*$. We say the q-wDBDHI asummption holds, if for any PPT algorithm $\mathcal{A}$, $Adv_{\mathcal{A}}^{q-wDBDHI}(k)$ is negligible in k. In our proof, we set $q = 1$, that is, the 1-wDBDHI assumption [27].

Target Collision Resistant Hash Function [14]. A TCR hash function H guarantees that given a random element x which is from the valid domain of H, a PPT adversary A cannot find $y \neq x$ such that $H(x) = H(y)$. We let $Adv_{H,A}^{TCR} = Pr[(x,y) \leftarrow \mathcal{A}(1^k) : H(x) = H(y), x \neq y, x, y \in DH]$ be the advantage of A in successfully finding collisions from a TCR hash function H, where DH is the valid input domain of H, and k is the security parameter. If a hash function is chosen from a TCR hash function family, $Adv_{H,A}^{TCR}$ is negligible.

4.2 Our Construction

Construction Roadmap. We leverage two different encryption technologies: one is IBE and the other is traditional Public Key Encryption (PKE). We first allow a user to generate a first level ciphertext under a receiver's identity. The first-level ciphertext will be further transformed into a second level ciphertext corresponding to a security device. The resulting ciphertext can be decrypted by a valid receiver with secret key and security device. Here, one might doubt that our construction is a trivial and straightforward combination of two different encryptions. Unfortunately, this is not true due to the fact that we need to further support security device revocability. A trivial combination of IBE and PKE cannot achieve our goal. To support revocability, we employ re-encryption technology such that the part of ciphertext for an old security device can be updated for a new device if the old device is revoked. Meanwhile, we need to generate a special key for the above ciphertext conversion. We also guarantee that the cloud server cannot achieve any knowledge of message by accessing the special key, the old ciphertext and the updated ciphertext. We further use hash-signature method to "sign" ciphertext such that once an component of ciphertext is tempered by adversary, the cloud and ciphertext receiver can tell. From the above presentations, we can see that our two-factor protection system with security device revocability cannot be obtained by trivially combining an IBE with a PKE. We present the system description as follows.

1) *Setup phase:* the setup phase generates all public parameters and master secret key used throughout the execution of system. The public parameters are shared with all parties participating into the system (including data sender/receiver, cloud server and a PKG), while the master secret key is given to the PKG. The details of setup phase are as follows.
   a) Set $\mathbb{G}$ and $\mathbb{G}_T$ to be groups of prime order q, and $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ to be a bilinear map.
   b) Choose $g, g_2, h \in \mathbb{G}$, $\alpha \in_R \mathbb{Z}_q^*$, the target collision resistant hash functions: $H_1 : \mathbb{G} \rightarrow \mathbb{Z}_q^*$, $H_2 : \{0,1\}^{2k} \rightarrow \mathbb{Z}_q^*$, $H_3 : \mathbb{G}_T \rightarrow \{0,1\}^{2k}$, $H_4 : \{0,1\}^* \rightarrow \mathbb{G}$, $H_5 : \{0,1\}^* \rightarrow \mathbb{G}$, and set $g_1 = g^\alpha$, where k is the security parameter as well as the bit length of message.
   c) Set the public parameters $param$ to be $(k, q, g, g_1, g_2, h, e(g,g), e(g_1,g_2), H_1, H_2, H_3, H_4, H_5, F(.))$, and the master secret key msk to be g_2^α, where $F(ID) = u_0 \cdot \prod_{j \in \mathcal{V}} u_j$, $u_0, u_j, \ldots, u_n \in_R \mathbb{G}$, and ID is an n-bit string and $\mathcal{V}$ is the set of all j for which the jth bit of ID is equal to 1.
2) *Key and device issued phase:* A SDI and a PKG will respectively generate a security device and a secret key for a registered user ID_i in secure channel such that the user can combine the security device with the secret key to recover message from its encrypted format. The details of key and security device issued phase are as follows.
   a) The SDI chooses $z_{i,1}, z_{i,2} \in_R \mathbb{Z}_q^*$, and sets the security device's description information as tpk_i: $(tpk_{i,1} = g^{z_{i,1}}, tpk_{i,2} = g^{z_{i,2}})$, and its corresponding secret information as tsk_i: $(tsk_{i,1} = z_{i,1}, tsk_{i,2} = z_{i,2})$. The SDI finally delivers the security device to a user ID_i.
   b) The SDI stores the tuple (ID_i, tpk_i) in a list $List$ shared with the cloud storage system.
   c) The PKG sets the secret key for a user ID_i as
   $$sk_{ID_i} = (sk_{ID_i,1}, sk_{ID_i,2}) = (g_2^\alpha F(ID_i)^s, g^s),$$
   where $s \in_R \mathbb{Z}_q^*$.
3) *First-level ciphertext generation phase:* a data sender encrypts a data under the identity of a data receiver, and further sends the encrypted data to the cloud server. Knowing public parameters $param$, a data $m \in \{0,1\}^k$ and a receiver's identity ID_i, a data sender encrypts a data to a first level encryption as follows. Note the first-level ciphertext generation is built on top of Waters IBE [46].
   a) Choose $\phi \in_R \{0,1\}^k$, set $t = H_2(m, \phi)$, compute $c_1 = (m||\phi) \oplus H_3(e(g_1,g_2)^t)$, $c_2 = g^t$, $c_3 = F(ID_i)^t$, $c_4 = H_4(c_1, c_2, c_3)^t$.
   b) Send the first-level ciphertext $C_1 = (c_1, c_2, c_3, c_4)$ to the cloud server.
4) *Second-level ciphertext phase:* after receiving the first-level ciphertext of a data from the data sender, the cloud server generates the second-level ciphertext. Knowing public parameters $param$, a first level encryption for the user, and the information (ID_i, tpk_i) stored in $List$, the cloud server encrypts $C_1 = (c_1, c_2, c_3, c_4)$ to a second-level ciphertext as follows.
   a) Choose $\beta_1, \beta_2 \in_R \{0,1\}^k$, set $r = H_2(\beta_1, \beta_2)$, compute $c_5 = c_1 \oplus (\beta_1||\beta_2)$, $c_6 = (\beta_1||\beta_2) \oplus H_3(e(g,g)^r)$, $c_7 = (tpk_{i,1} \cdot tpk_{i,2}^{H_1(tpk_i)})^r$, $c_8 = h^r$, and $c_9 = H_5(c_5, c_6, c_7, c_8)^r$.

b) Output the second-level ciphertext $C_2 = (c_2, c_3, c_4, c_5, c_6, c_7, c_8, c_9)$ to the cloud.

5) *Device updated phase:* Once a device of a user needs to be updated due to some incidences (e.g., it is either lost or stolen), the user first reports the issue to the SDI. The SDI then issues a new device for the user.
   a) The SDI chooses $z_{i,1}, z_{i,2} \in_R \mathbb{Z}_q^*$, and sets the security device's description information as tpk_i: $(tpk_{i,1} = g^{z_{i,1}},\ tpk_{i,2} = g^{z_{i,2}})$, and its corresponding secret information as tsk_i: $(tsk_{i,1} = z_{i,1},\ tsk_{i,2} = z_{i,2})$. The SDI finally delivers the security device to a user ID_i.
   b) The SDI further updates the list $List$.
6) *Ciphertext updated phase:* The SDI notifies the cloud server to update the ciphertext of the user by sending a special piece of information.
   a) The SDI first sends a piece of information to the cloud server so as to inform the cloud to execute the ciphertext updated process. The information $rk_{tpk_i \to \widetilde{tpk_i}} = (rk_1, rk_2, ID_i, tpk_i, \widetilde{tpk_i})$ is constructed as

$$rk_1 = \left(\widetilde{tpk}_{i,1} \cdot \widetilde{tpk}_{i,2}^{H_1(\widetilde{tpk_i})}\right)^{(\delta_{i,1}+\delta_{i,2}\cdot H_1(tpk_i))^{-1}} \cdot h^{\xi},$$
$$rk_2 = \left(tpk_{i,1} \cdot tpk_{i,2}^{H_1(tpk_i)}\right)^{\xi},$$

   where $\xi \in_R \mathbb{Z}_q^*$, $tsk_i = (\delta_{i,1}, \delta_{i,2})$ is the decryption key of the old security device.
   b) After receiving the information $rk_{tpk_i \to tpk_j}$, the cloud server updates the ciphertext C_2 as follows.
      i) Parse C_2 as $(c_2, c_3, c_4, c_5, c_6, c_7, c_8, c_9)$, and $rk_{tpk_i \to tpk_j}$ as $(rk_1, rk_2, ID_i, tpk_i, tpk_j)$.
      ii) Check

$$e(c_7, h) = e\left(tpk_{i,1} \cdot tpk_{i,2}^{H_1(tpk_i)}, c_8\right),$$
$$e(c_8, H_5(c_5, c_6, c_7, c_8)) = e(h, c_9). \quad (1)$$

      If the equations do not hold, output $\perp$; else proceed.
      iii) Compute $c_{10} = \frac{e(c_7, rk_1)}{e(c_8, rk_2)}$.
      iv) Output an updated ciphertext $C_2^{(up)} = (c_2, c_3, c_4, c_5, c_6, c_{10})$. Note that although the number of elements in the updated ciphertext is less than that of the original ciphertext, the actual size of the updated ciphertext is larger than that of the original one. This can be seen in our efficiency analysis in Table 4.
7) *Data recovery phase.* A data receiver uses a decryption key and a device to recover the data as follows.
   a) If the ciphertext has not been updated, run as
      - Parse C_2 as $(c_2, c_3, c_4, c_5, c_6, c_7, c_8, c_9)$, sk_{ID_i} as $(sk_{ID_i,1}, sk_{ID_i,2})$ and tsk_i as $(tsk_{i,1}, tsk_{i,2})$.
      - Security device computation phase: the device intakes a component of the ciphertext c_7, and outputs the result $\theta = c_7^{\frac{1}{tsk_{i,1}+tsk_{i,2}\cdot H_1(tpk_i)}}$.
      - Secret key decryption phase: given the partial decryption result θ, the user uses his/her secret key to recover the message as follows. Compute $e(g,g)^r = e(g,\theta)$ and $c_1 = c_5 \oplus c_6 \oplus H_3(e(g,g)^r)$. Further compute $e(g_1,g_2)^t = \frac{e(c_2, sk_{ID_i,1})}{e(c_3, sk_{ID_i,2})}$, and $m||\phi = c_1 \oplus H_3(e(g_1,g_2)^t)$.
      - If $c_2 = g^{H_2(m,\phi)}$, $c_3 = F(ID_i)^{H_2(m,\phi)}$, $c_4 = H_4(c_1,c_2,c_3)^{H_2(m,\phi)}$, $c_7 = (tpk_{i,1} \cdot tpk_{i,2}^{H_1(tpk_i)})^{H_2(\beta_1,\beta_2)}$, $c_8 = h^{H_2(\beta_1,\beta_2)}$, and $c_9 = H_5(c_5,c_6,c_7,c_8)^{H_2(\beta_1,\beta_2)}$, output m; otherwise, output $\perp$.
   b) If the ciphertext has been updated, run as
      - Parse $C_2^{(up)}$ as $(c_2, c_3, c_4, c_5, c_6, c_{10})$, sk_{ID_i} as $(sk_{ID_i,1}, sk_{ID_i,2})$ and tsk_i as $(tsk_{i,1}, tsk_{i,2})$.
      - Security device computation phase: the device intakes a component of the ciphertext c_{10}, and outputs $\hat{\theta} = c_{10}^{\frac{1}{tsk_{i,1}+tsk_{i,2}\cdot H_1(tpk_i)}} = e(g,g)^r$.
      - Secret key decryption phase: given the partial decryption result $\hat{\theta}$, the user recovers m as in the above secret key decryption phase, computing $c_1 = c_5 \oplus c_6 \oplus H_3(e(g,g)^r)$, $e(g_1,g_2)^t = \frac{e(c_2, sk_{ID_i,1})}{e(c_3, sk_{ID_i,2})}$, and $m||\phi = c_1 \oplus H_3(e(g_1,g_2)^t)$.

4.3 Discussions

- Multiple revocability for device. Our construction supports one-time device revocability that may be not sufficient enough in practice. We here show that the system can be extended to support multiple revocability by leveraging the technique introduced in [9]. We revise the $rk_{tpk_i \to \widetilde{tpk_i}}$ as $\frac{\widetilde{tsk}_{i,1}+\widetilde{tsk}_{i,2}H_1(\widetilde{tpk_i})}{tsk_{i,1}+tsk_{i,2}H_1(tpk_i)}$, and the ciphertext update component c_{10} as $c_7^{rk_{tpk_i \to \widetilde{tpk_i}}}$. The updated ciphertex is identical to the original one except for c_{10} taking place of c_7. We note that c_7 is not an input for H_5 here. It is not difficult to see that a user can recover the underlying message by using a updated security device corresponding to $\widetilde{tpk_i}$.
- Revocability for identity factor. It is possible to extend our construction to support identity revocability as well by leveraging the IBPRE technology in [20]. We will leave this as a future work.
- Feasibility. Our system requires an SDI to issue a security device to a user in the registration phase. It is much like the case where a bank client is issued a e-banking token when opening a bank account. The device can be directly delivered to the user by mail or in person. In practice, the security device can be constructed from a USB token that is extremely portable for users. Meanwhile, the user also achieves a

secret key given by a PKG. The secret key can be stored in the user's PC or clouds based on the preference of the user.

The two-factor protection is necessary for high valuable sensitive data, such as personal genome information and company commercial secret. A user may not always gain access to his PC. Suppose a patient has to check his encrypted medical record stored in a cloud storage system in a publicly used computer. He may download his secret key as well as an encrypted record to the local computer, and next plug in a security device to unlock the record with the secret key. This message recovery is almost identical to the login operation of on-line banking where user needs to use a login password along with a security token (sometimes with a smart-phone). Compared to a smart-phone, a USB token is portable. After reading the record, the patient can just plug out the device and leave. Since the decryption depends on both the secret key and the device, even the computer is corrupted by an intruder, the intruder still cannot access the record. We note that to date some information, such as visit history, download history, may be easily leaked from browser, the usage of security device combining with a secret key can double protect the secrecy of information to a large extent.

5 System Evaluation

5.1 Security Analysis

We separate two security levels for our scheme: one is allowing an adversary to achieve the secret key of user but not the corresponding secure device, and the other is the reversed case.

For Type-I Security. Here we allow an adversary to obtain the secret key of a user but not the corresponding security device. We analyze the security of our scheme under the model of Type-I.

Practical analysis: An adversary $\mathcal{A}$ now is given the secret key sk_{ID_i} of user ID_i. We show that $\mathcal{A}$ cannot recover the underlying message by only leveraging knowledge of sk_{ID_i} as follows.

Suppose there is a ciphertext $C_2 = (c_2, c_3, c_4, c_5, c_6, c_7, c_8, c_9)$ for a user ID_i, which is stored in the cloud server, where $c_2 = g^t$, $c_3 = F(ID_i)^t$, $c_4 = H_4(c_1, c_2, c_3)^t$, $c_5 = (m||\phi) \oplus H_3(e(g_1, g_2)^t) \oplus (\beta_1||\beta_2)$, $c_6 = (\beta_1||\beta_2) \oplus H_3(e(g, g)^r)$, $c_7 = (tpk_{i,1} \cdot tpk_{i,2}^{H_1(tpk_i)})^r$, $c_8 = h^r$, and $c_9 = H_5(c_5, c_6, c_7, c_8)^r$, $t = H_2(m, \phi)$, $r = H_2(\beta_1, \beta_2)$. $\mathcal{A}$ can compute

$$\begin{aligned} & H_3\left(\frac{e(c_2, sk_{ID_i,1})}{e(c_3, sk_{ID_i,2})}\right) \oplus c_5 \\ &= H_3\left(\frac{e(g^t, g_2^{\alpha} F(ID_i)^s)}{e(F(ID_i)^t, g^s)}\right) \oplus c_5 \\ &= H_3(e(g_1, g_2)^t) \oplus H_3(e(g_1, g_2)^t) \oplus (m||\phi) \oplus (\beta_1||\beta_2) \\ &= (m||\phi) \oplus (\beta_1||\beta_2), \end{aligned} \tag{2}$$

$$\begin{aligned} & H_3\big(e(g, c_7^{\frac{1}{tsk_{i,1}+tsk_{i,2}\cdot H_2(tpk_i)}})\big) \\ &= H_3\big(e(g, (tpk_{i,1} tpk_{i,2}^{H_1(tpk_i)})^{\frac{r}{tsk_{i,1}+tsk_{i,2}H_1(tpk_i)}})\big) \\ &= H_3(e(g, g)^r). \end{aligned} \tag{3}$$

From E.q. (3), it can be seen that $\mathcal{A}$ can retrieve $H_3(e(g, g)^r)$ (without given the security device) as long as it can correctly guess the secret components $tsk_{i,1}$ and $tsk_{i,2}$ simultaneously with probability $\frac{1}{q^2}$. Alternatively, if $\mathcal{A}$ is able to correctly guess the output of H_3 with probability $\frac{1}{2^{2k}}$, then it can recover $\beta_1||\beta_2$ so as to gain access to the message m.

Theoretical analysis:If an adversary $\mathcal{A}$ recovers the message by a given secret key, we can build an algorithm $\mathcal{B}$ breaking the 1-wDBDHI assumption.

Setup Phase. $\mathcal{B}$ is given an instance of the 1-wDBDHI problem, i.e. $(g, A = g^a, B = g^b, T)$, where T either is random or is equal to $e(g, g)^{\frac{b}{a}}$. $\mathcal{B}$ chooses a $F(.)$ as in the real scheme, chooses $\alpha, \omega \in_R \mathbb{Z}_q^*$, sets $y = A = g^a$, and returns $param = (k, q, g, g_1 = g^{\alpha}, g_2 = y, h = y^{\omega}, e(g, g), e(g_1, g_2), H_1, H_2, H_3, H_4, H_5, F(.))$ to $\mathcal{A}$. H_1, H_2, H_3, H_4 and H_5 are chosen as in the real scheme, and meanwhile, they are random oracles controlled by $\mathcal{B}$.

- H_1 : On receipt of a tpk_i, if there exists a tuple (tpk_i, φ_1) in the list $List^{H_1}$, $\mathcal{B}$ returns φ_1 to $\mathcal{A}$. Otherwise, $\mathcal{B}$ chooses a $\varphi_1 \in_R \mathbb{Z}_q^*$, returns it to $\mathcal{A}$, and adds (tpk_i, φ_1) to the list $List^{H_1}$.
- H_2 : On receipt of a tuple (m, ϕ), if there exists a tuple (m, ϕ, φ_2) in the list $List^{H_2}$, $\mathcal{B}$ returns φ_2 to $\mathcal{A}$. Otherwise, $\mathcal{B}$ chooses a $\varphi_2 \in_R \mathbb{Z}_q^*$, returns it to $\mathcal{A}$, and adds (m, ϕ, φ_2) to the list $List^{H_2}$.
- H_3 : On receipt of a $R \in \mathbb{G}_T$, if there exists a tuple (R, ξ) in the list $List^{H_3}$, $\mathcal{B}$ returns ξ to $\mathcal{A}$. Otherwise, $\mathcal{B}$ chooses a $\xi \in_R \{0, 1\}^{2k}$, returns it to $\mathcal{A}$, and adds (R, ξ) to the list $List^{H_3}$.
- H_4 : On receipt of a tuple (c_1, c_2, c_3), if there exists a tuple (c_1, c_2, c_3, φ_3) in the list $List^{H_4}$, $\mathcal{B}$ returns g^{φ_3} to $\mathcal{A}$. Otherwise, $\mathcal{B}$ chooses a $\varphi_3 \in_R \mathbb{Z}_q^*$, returns g^{φ_3} to $\mathcal{A}$. $\mathcal{B}$ then adds (c_1, c_2, c_3, φ_3) to the list $List^{H_4}$.
- H_5 : On receipt of a tuple (c_5, c_6, c_7, c_8), if there exists a tuple (c_5, c_6, c_7, c_8, φ_4) in the list $List^{H_4}$, $\mathcal{B}$ returns g^{φ_4} (resp. $g^{a\varphi_4}$) to $\mathcal{A}$. Otherwise, if $c_5 = c_5^*$, $c_6 = c_6^*$, $c_7 = c_7^*$, and $c_8 = c_8^*$, $\mathcal{B}$ chooses a $\varphi_4 \in_R \mathbb{Z}_q^*$, and sets $g^{a\varphi_4}$ for $\mathcal{A}$; else it returns g^{φ_4} to $\mathcal{A}$. $\mathcal{B}$ then adds (c_5, c_6, c_7, c_8, φ_4) to the list $List^{H_5}$.

Phase 1. $\mathcal{A}$ issues the following queries to $\mathcal{B}$.

1) *Security device queries.* $\mathcal{A}$ issues an ID_i to $\mathcal{B}$ for querying its corresponding security device. $\mathcal{B}$ chooses $coin_i \in \{0, 1\}$ so that $Pr[coin_i = 1] = \vartheta$ (ϑ will be determined later) and works as follows.
   - If $coin_i = 0$, $\mathcal{B}$ aborts and outputs a random bit $b \in \{0, 1\}$. $\mathcal{B}$ will add a tuple (ID_i, tpk_i, $\perp$, $coin_i = 0$, $z_{i,1}$, $z_{i,2}$) to $DeviceList$, where $tpk_{i,1} = y^{z_{i,1}}$, $tpk_{i,2} = y^{z_{i,2}}$ and $z_{i,1}, z_{i,2} \in_R \mathbb{Z}_q^*$.
   - Otherwise, if there exists a tuple (ID_i, tpk_i, tsk_i, $coin_i = 1$, $z_{i,1}$, $z_{i,2}$) in the list $DeviceList$, $\mathcal{B}$ returns tsk_i along with tpk_i to $\mathcal{A}$; else, $\mathcal{B}$ chooses $z_{i,1}, z_{i,2} \in_R \mathbb{Z}_q^*$, sets $tsk_i = (z_{i,1}, z_{i,2})$, $tpk_{i,1} = g^{z_{i,1}}$ and $tpk_{i,2} = g^{z_{i,2}}$. $\mathcal{B}$ then adds (ID_i, tpk_i, tsk_i, $coin_i = 1$, $z_{i,1}$, $z_{i,2}$) to $DeviceList$, and returns (tsk_i, tpk_i) to $\mathcal{A}$.
2) *Secret key queries.* $\mathcal{A}$ issues an ID_i to $\mathcal{B}$ for querying the secret key of ID_i. $\mathcal{B}$ then checks whether there

exists a tuple (ID_i, sk_{ID_i}) in $SecretKeyList$ or not. If yes, $\mathcal{B}$ returns sk_{ID_i}; else, $\mathcal{B}$ generates sk_{ID_i} as in the real scheme with knowledge of α and next adds (ID_i, sk_{ID_i}) to $SecretKeyList$.

3) *Ciphertext update queries.* $\mathcal{A}$ issues a tuple (C_2, tpk_i) to $\mathcal{B}$ for querying an update ciphertext $C_2^{(up)}$ under (ID_i, $\widetilde{tpk}_i$). $\mathcal{B}$ first checks whether there are tuples (m, ϕ, t), (β_1, β_2, r) in the list $List^{H_2}$ so that $c_2 = g^t$, $c_4 = H_4(c_1, c_2, c_3)^t$, $c_7 = (tpk_{i,1} \cdot tpk_{i,2}^{H_1(tpk_i)})^r$, $c_8 = h^r$ and $c_9 = H_5(c_5, c_6, c_7, c_8)^r$. If no, output $\perp$; else proceed. $\mathcal{B}$ further recovers (ID_i, tpk_i, $*$, $coin_i$, $z_{i,1}$, $z_{i,2}$) from the list $DeviceList$.
   - If $coin_i = 1$, $\mathcal{B}$ uses $z_{i,1}$ and $z_{i,2}$ to construct the information $rk_{tpk_i \to \widetilde{tpk}_i}$ as in the real scheme. $\mathcal{B}$ then computes C_{10} by using $rk_{tpk_i \to \widetilde{tpk}_i}$, and outputs $C_2^{(up)}$ as in the real scheme.
   - If $coin_i = 0$, and $\widetilde{tpk}_i$ is given to $\mathcal{A}$, $\mathcal{B}$ outputs $\perp$. Otherwise, $\mathcal{B}$ computes $c_{10} = e(g^r, \widetilde{tpk}_{i,1} \cdot \widetilde{tpk}_{i,2}^{H_1(\widetilde{tpk}_i)})$, and next outputs ($c_2$, c_3, c_4, c_5, c_6, c_{10}) as the updated ciphertext.
4) *Data recovery queries.* $\mathcal{A}$ issues a ciphertext to $\mathcal{B}$. $\mathcal{B}$ recovers the message as follows.
   - For ciphertext C_2, $\mathcal{B}$ first checks whether there are tuples (m, ϕ, t), (β_1, β_2, r) in the list $List^{H_2}$ so that $c_2 = g^t$, $c_4 = H_4(c_1, c_2, c_3)^t$, $c_7 = (tpk_{i,1} \cdot tpk_{i,2}^{H_1(tpk_i)})^r$, $c_8 = h^r$ and $c_9 = H_5(c_5, c_6, c_7, c_8)^r$. If no, $\mathcal{B}$ outputs $\perp$. Otherwise, $\mathcal{B}$ recovers tuples ($R_1 = e(g_1, g_2)^t$, ξ_1) and ($R_2 = e(g, g)^r$, ξ_2) from the list $List^{H_3}$, and computes $m||\phi = (c_5 \oplus (c_6 \oplus \xi_2)) \oplus \xi_1$.
   - For ciphertext $C_2^{(up)}$, $\mathcal{B}$ first checks whether there are tuples (m, ϕ, t), (β_1, β_2, r) in the list $List^{H_2}$ so that $c_2 = g^t$, $c_4 = H_4(c_1, c_2, c_3)^t$ and $c_{10} = e(g^r, tpk_{i,1} \cdot tpk_{i,2}^{H_1(tpk_i)})$. If no, $\mathcal{B}$ outputs $\perp$. Otherwise, $\mathcal{B}$ recovers tuples ($R_1 = e(g_1, g_2)^t$, ξ_1) and ($R_2 = e(g, g)^r$, ξ_2) from the list $List^{H_3}$, and computes $m||\phi = (c_5 \oplus (c_6 \oplus \xi_2)) \oplus \xi_1$.

Challenge phase. $\mathcal{A}$ outputs m_0, m_1 and (ID_i^*, tpk_i^*). $\mathcal{B}$ searches tpk_i^* from the list $DeviceList$. If $coin_i^* = 1$, $\mathcal{B}$ aborts and outputs a $b \in \{0, 1\}$. Else, $\mathcal{B}$ proceeds.

- For original ciphertext, $\mathcal{B}$ chooses $\beta_1^*, \beta_2^*, \phi^* \in_R \{0,1\}^k$, $b \in_R \{0, 1\}$, and $t \in_R \mathbb{Z}_q^*$. It issues $e(g_1, g_2)^t$ to H_3 to obtain ξ^* and sets $c_1^* = (m_b||\phi^*) \oplus \xi^*$, $c_2^* = g^t$, $c_3^* = F(ID_i^*)^t$. It then issues c_1^*, c_2^*, c_3^* to H_4 to obtain φ_3^*, and sets $c_4^* = g^{t\varphi_3^*}$. $\mathcal{B}$ further recovers $z_{i,1}^*$, $z_{i,2}^*$ from $DeviceList$, issues tpk_i^* to H_1 to obtain φ_1^*, and sets $c_5^* = c_1^* \oplus (\beta_1^*||\beta_2^*)$, $c_6^* = (\beta_1^*||\beta_2^*) \oplus H_3(T)$, $c_7^* = B^{z_{i,1}^* + z_{i,2}^* \varphi_1^*}$ and $c_8^* = B^{\omega}$. It finally issues (c_5^*, c_6^*, c_7^*, c_8^*) to H_5 to obtain φ_4^*, and sets $c_9^* = B^{\varphi_4^*}$. $\mathcal{B}$ outputs $C_2^* = (c_2^*, c_3^*, c_4^*, c_5^*, c_6^*, c_7^*, c_8^*, c_9^*)$.
- For updated ciphertext, $\mathcal{B}$ sets $C_2^{*(up)} = (c_2^*, c_3^*, c_4^*, c_5^*, c_6^*, c_{10}^*)$ as $c_2^* = g^t$, $c_3^* = F(ID_i^*)^t$, $c_4^* = g^{t\varphi_3^*}$, $c_5^* = c_1^* \oplus (\beta_1^*||\beta_2^*)$, $c_6^* = (\beta_1^*||\beta_2^*) \oplus H_3(T)$ and $c_{10}^* = e(g, B^{z_{i,1}^* + z_{i,2}^* \varphi_1^*})$.

We imply $H_2(\beta_1^*, \beta_2^*) = \frac{b}{a}$. If $b = b'$, $\mathcal{B}$ guesses $T = e(g, g)^{\frac{b}{a}}$; else, T is a random element in $\mathbb{G}_T$.

Phase 2. As in Phase 1 (but with restrictions).

Guess phase. $\mathcal{A}$ outputs a guess bit $b' \in \{0, 1\}$.

Probabilistic Analysis. The simulations of H_i ($i \in \{1, 4, 5\}$) are perfect. If $\mathcal{A}$ does not either issue (β_1^*, β_2^*) to H_2 or issue T to H_3 before the challenge phase, the simulations of H_2 and H_3 are perfect. We denote by $AskH_2^*$ and $AskH_3^*$ the evens that (β_1^*, β_2^*) has been issued to H_2, and T has been issued to H_3, respectively.

The responses to the security device queries, secret key queries, and the challenge phase are perfect as long as $\mathcal{B}$ does not abort. We let $Abort$ be the event of $\mathcal{B}$ aborting in the responses to the security device queries or in the challenge phase. Thus, we have $Pr[\neg Abort] \geq \vartheta^{q_{sd}}(1 - \vartheta)$, which is maximized at $\vartheta_{opt} = \frac{q_{sd}}{1+q_{sd}}$, where q_{sd} is the total number of security device queries. By using ϑ_{opt}, we have the probability $Pr[\neg Abort]$ is at least $\frac{1}{\hat{e}(1+q_{sd})}$, where $\hat{e}$ denotes the base of the natural logarithm.

The simulation of ciphertext update queries is perfect as well unless $\mathcal{A}$ is able to issue a valid original ciphertext without the help of H_2. We state that this incident will occur with probability $Pr[CUError] \leq \frac{q_{cu}}{q}$, where q_{cu} is the total number of ciphertext update queries.

The simulation of data recovery queries is perfect except that $\mathcal{B}$ rejects the queries of some valid ciphertexts. This kind of exception happens when an issued ciphertext can be constructed without querying H_3. We set $valid$, $AskH_2$, $AskH_3$ to be the events that a given ciphertext is valid, (β_1, β_2) has been issued to H_2, and $e(g, g)^r$ has been issued to H_3, respectively. From the simulation, we have $Pr[valid|\neg AskH_3] \leq \frac{q_{H_3}}{2^l} + \frac{1}{q}$, and $Pr[valid|\neg AskH_2] \leq \frac{q_{H_2}}{2^l} + \frac{1}{q}$, where q_{H_2} and q_{H_3} are the total numbers of querying H_2 and H_3, respectively. We let $Pr[DRErr]$ be the probability that the event $valid|(\neg AskH_2 \vee \neg AskH_3)$ occurs, then we have $Pr[DRErr] \leq (\frac{q_{H_2}+q_{H_3}}{2^l} + \frac{2}{q})q_{dr}$,where q_{dr} denotes the total number of data recovery queries.

Let Bad denote the event that $(H_2^*|\neg H_3^*) \vee H_3^* \vee CUErr \vee DRErr|\neg Abort$. We have

$$\begin{aligned}\epsilon &= |Pr[b = b'] - \frac{1}{2}| \leq \frac{1}{2} Pr[Bad] \\ &= \frac{1}{2} Pr[(H_2^*|\neg H_3^*) \vee H_3^* \vee CUErr \vee DRErr|\neg Abort] \\ &\leq \frac{1}{2Pr[\neg Abort]} \left(AskH_3^* + \frac{q_{H_2} + (q_{H_2} + q_{H_3})q_{dr}}{2^l} + \frac{2q_{dr} + q_{cu}}{q} \right).\end{aligned}$$

After organizing the inequality above, we have

$$\begin{aligned}AskH_3^* &\geq 2\epsilon Pr[\neg Abort] - \frac{q_{H_2} + (q_{H_2} + q_{H_3})q_{dr}}{2^l} - \frac{2q_{dr} + q_{cu}}{q} \\ &\geq \frac{2\epsilon}{\hat{e}(1 + q_{sd})} - \frac{q_{H_2} + (q_{H_2} + q_{H_3})q_{dr}}{2^l} - \frac{2q_{dr} + q_{cu}}{q}.\end{aligned}$$

Therefore, we have

$$\begin{aligned}\epsilon' &\geq \frac{1}{q_{H_3}} (AskH_3^*) \\ &\geq \frac{1}{q_{H_3}} \left(\frac{2\epsilon}{\hat{e}(1 + q_{sd})} - \frac{q_{H_2} + (q_{H_2} + q_{H_3})q_{dr}}{2^l} - \frac{2q_{dr} + q_{cu}}{q} \right).\end{aligned}$$

TABLE 2
Computation Comparison I

Schemes	[2]	[20]	Ours
Secret Key Generation	$2C_e$	C_e	$2C_e$
Security Device Generation	$\perp$	$\perp$	$2C_e$
Ciphertext Generation	$C_e + C_{\bar{e}} + 3C_p$	$4C_e + C_p$	first-level Ciph.: $3C_e + C_{\bar{e}}$ second-level Ciph.: $4C_e + C_{\bar{e}}$
Ciphertext Update	$\perp$	$2C_e + 5C_p$	$5C_e + 6C_p$
Device Update	$\perp$	$\perp$	$2C_e$
Data Recovery (From Original Ciph.)	$C_e + C_p$	$4C_e + 2C_p$	$8C_e + 2C_p$
Data Recovery (From Updated Ciph.)	$\perp$	$C_e + 2C_p$	$7C_e + C_{\bar{e}} + 2C_p$

The running time of $\mathcal{B}$ is bounded by

$$t' \leq t + O(1)(q_{H_1} + q_{H_2} + q_{H_3} + q_{H_4} + q_{H_5} + q_{sd} + q_{sk} + q_{cu} + q_{dr}) + t_e(2q_{sd} + 3q_{sk} + q_{H_2}(5q_{cu} + 5q_{dr}) + 5q_{cu} + q_{dr}) + 2t_p q_{cu},$$

where q_{H_i} ($i \in \{1, 2, 3, 4, 5\}$) denotes the total number of random oracle H_i queries, and q_{sk} denote the total number of secret key queries, t_e and t_p denote the running time of an exponentiation and a pairing, respectively.

For Type-II security. We allow an adversary to obtain the security device but not the corresponding secret key. We analyze the security under the model of Type-II.

Practical analysis: An adversary $\mathcal{A}$ is given the security device only. We show that it cannot recover the message by using the device. Complementary to the case of Type-I adversary, Type-II $\mathcal{A}$ can only compute

$$\begin{aligned} & H_3(e(g, c_7^{\frac{1}{tsk_{i,1}+tsk_{i,2}\cdot H_1(tpk_i)}})) \\ &= H_3(e(g, (tpk_{i,1} \cdot tpk_{i,2}^{H_1(tpk_i)})^{\frac{r}{tsk_{i,1}+tsk_{i,2}H_1(tpk_i)}})) \\ &= H_3(e(g,g)^r). \end{aligned} \tag{4}$$

Thus, $c_5 \oplus (c_6 \oplus H_3(e(g,g)^r)) = (m||\phi) \oplus H_3(e(g_1, g_2)^t)$.

$$\begin{aligned} H_3\left(\frac{e(c_2, sk_{ID_i,1})}{e(c_3, sk_{ID_i,2})}\right) &= H_3\left(\frac{e(g^t, g_2^{\alpha} F(ID_i)^s)}{e(F(ID_i)^t, g^s)}\right) \\ &= H_3(e(g_1, g_2)^t). \end{aligned} \tag{5}$$

From the above equations, it can be seen that $\mathcal{A}$ can recover m, if it can make a correct guess on either the output of H_3 or the secret key exponents α and r with respective probability $\frac{1}{2^{2k}}$ and $\frac{1}{q^2}$.

Theoretical analysis: If an adversary $\mathcal{A}$ can recover the message by a given security device, we can build an algorithm $\mathcal{B}$ to break the BDH assumption.

Setup phase. $\mathcal{B}$ is given an instance of the BDH problem, i.e. $(g, A = g^a, B = g^b, C = g^c, T)$, where T either is random or is equal to $e(g,g)^{abc}$. In this proof, we mainly leverages Waters proof technique [46], and reuses the three functions, $F(ID) = (q - 4q_{sk}\bar{k}) + x' + \sum_{j\in\mathcal{V}} x_j$, $J(ID) = y' + \sum_{j\in\mathcal{V}} y_j$ and $K(ID)$ (0, if $x' + \sum_{j\in\mathcal{V}} x_j = 0 \ mod \ 4q_{sk}$; 1, otherwise) proposed in [46], where q_{sk} is the number of secret key queries, an integer $\bar{k} \in_R [0, n]$, $\overrightarrow{x} = (x_j)$ and $\overrightarrow{y} = (y_j)$ are two random n-length vectors, x' is chosen from $[0, 4q_{sk} - 1]$ and $y' \in_R \mathbb{Z}_q^*$. $\mathcal{B}$ sets $g_1 = A$, $g_2 = B$, $h = g^{\omega}$, $u_0 = g_2^{p-\bar{k}4q_{sk}+x'} g^{y'}$ and $u_j = g_2^{x_j} g^{y_j}$, where $\omega \in_R \mathbb{Z}_q^*$. $\mathcal{B}$ finally outputs $param = (k, q, g, g_1, g_2, h, e(g,g), e(g_1, g_2), H_1, H_2, H_3, H_4, H_5, F(.))$, where H_1, H_2, H_3, H_4 and H_5 are chosen as in the real scheme.

- H_1 : On receipt of a tpk_i, if there exists a tuple (tpk_i, φ_1) in the list $List^{H_1}$, $\mathcal{B}$ returns φ_1 to $\mathcal{A}$. Otherwise, $\mathcal{B}$ chooses a $\varphi_1 \in_R \mathbb{Z}_q^*$, returns it to $\mathcal{A}$, and adds (tpk_i, φ_1) to the list $List^{H_1}$.
- H_2 : On receipt of a tuple (m, ϕ), if there exists a tuple (m, ϕ, φ_2) in the list $List^{H_2}$, $\mathcal{B}$ returns φ_2 to $\mathcal{A}$. Otherwise, $\mathcal{B}$ chooses a $\varphi_2 \in_R \mathbb{Z}_q^*$, returns it to $\mathcal{A}$, and adds (m, ϕ, φ_2) to the list $List^{H_2}$.
- H_3 : On receipt of a $R \in \mathbb{G}_T$, if there exists a tuple (R, ξ) in the list $List^{H_3}$, $\mathcal{B}$ returns ξ to $\mathcal{A}$. Otherwise, $\mathcal{B}$ chooses a $\xi \in_R \{0,1\}^{2k}$, returns it to $\mathcal{A}$, and adds (R, ξ) to the list $List^{H_3}$.
- H_4 : On receipt of a tuple (c_1, c_2, c_3), if there exists a tuple (c_1, c_2, c_3, φ_3) in the list $List^{H_4}$, $\mathcal{B}$ returns g^{φ_3} to $\mathcal{A}$. Otherwise, $\mathcal{B}$ chooses a $\varphi_3 \in_R \mathbb{Z}_q^*$ and returns g^{φ_3} to $\mathcal{A}$. $\mathcal{B}$ then adds (c_1, c_2, c_3, φ_3) to the list $List^{H_4}$.
- H_5 : On receipt of a tuple (c_5, c_6, c_7, c_8), if there exists a tuple (c_5, c_6, c_7, c_8, φ_4) in the list $List^{H_4}$, $\mathcal{B}$ returns g^{φ_4}. Otherwise, $\mathcal{B}$ chooses a $\varphi_4 \in_R \mathbb{Z}_q^*$, and returns g^{φ_4} to $\mathcal{A}$. $\mathcal{B}$ then adds (c_5, c_6, c_7, c_8, φ_4) to the list $List^{H_5}$.

Phase 2. $\mathcal{A}$ issues the following queries to $\mathcal{B}$.

1) *Security device queries.* $\mathcal{A}$ issues an ID_i to $\mathcal{B}$. $\mathcal{B}$ generates any security device as in the real scheme.
2) *Secret key queries.* $\mathcal{A}$ issues an ID_i to $\mathcal{B}$. If $K(ID) = 0$, $\mathcal{B}$ aborts. Else, $\mathcal{B}$ chooses an $s \in_R \mathbb{Z}_q^*$, and constructs sk_{ID_i} as $(g_1^{\frac{-J(ID)}{F(ID)}}(u_0 \prod_{j\in\mathcal{V}} u_j)^s, g_1^{\frac{-1}{F(ID)}} g^s)$.
3) *Ciphertext update queries.* $\mathcal{A}$ issues a tuple (C_2, tpk_i) to $\mathcal{B}$ for querying an update ciphertext $C_2^{(up)}$ under (ID_i, tpk_j). $\mathcal{B}$ first checks whether there are tuples (m, ϕ, t), (β_1, β_2, r) in the list $List^{H_2}$ so that $c_2 = g^t$, $c_4 = H_4(c_1, c_2, c_3)^t$, $c_7 = (tpk_{i,1} \cdot tpk_{i,2}^{H_1(tpk_i)})^r$, $c_8 = h^r$ and $c_9 = H_5(c_5, c_6, c_7, c_8)^r$. If no, output $\perp$; else proceed. $\mathcal{B}$ can generate any update ciphertext for $\mathcal{A}$ as it can generate any security device.
4) *Data recovery queries.*
   - For ciphertext C_2, $\mathcal{B}$ first checks whether there are tuples (m, ϕ, t), (β_1, β_2, r) in the list $List^{H_2}$ so that $c_2 = g^t$, $c_4 = H_4(c_1, c_2, c_3)^t$, $c_7 = (tpk_{i,1} \cdot$

TABLE 3
Communication Comparison I

Schemes	[2]	[20]	Ours
Secret Key Size	$\|\mathbb{G}\|$	$\|\mathbb{G}\|$	$2\|\mathbb{G}\|$
Security Device Size	$\perp$	$\perp$	$2\|\mathbb{G}\|+2\|\mathbb{Z}_q\|$
Original Ciphertext Size	$\|\mathbb{G}\|+2l$	$2\|\mathbb{G}\|+\|\mathbb{G}_T\|+l$	$6\|\mathbb{G}\|+4l$
Updated Ciphertext Size	$\perp$	$\|\mathbb{G}\|+\|\mathbb{G}_T\|+2l$	$3\|\mathbb{G}\|+\|\mathbb{G}_T\|+4l$
Cost in Ciphertext Update	$\perp$	$\|\mathbb{G}\|+l$	$2\|\mathbb{G}\|$

$tpk_{i,2}^{H_1(tpk_i)})^r$, $c_8 = h^r$ and $c_9 = H_5(c_5, c_6, c_7, c_8)^r$. If no, $\mathcal{B}$ outputs $\perp$. Otherwise, $\mathcal{B}$ recovers tuples $(R_1 = e(g_1, g_2)^t,\ \xi_1)$ and $(R_2 = e(g, g)^r,\ \xi_2)$ from the list $List^{H_3}$, and computes $m||\phi = (c_5 \oplus (c_6 \oplus \xi_2)) \oplus \xi_1$.

- For ciphertext $C_2^{(up)}$, $\mathcal{B}$ first checks whether there are tuples (m, ϕ, t), (β_1, β_2, r) in the list $List^{H_2}$ so that $c_2 = g^t$, $c_4 = H_4(c_1, c_2, c_3)^t$ and $c_{10} = e(g^r, tpk_{i,1} \cdot tpk_{i,2}^{H_1(tpk_i)})$. If no, $\mathcal{B}$ outputs $\perp$. Otherwise, $\mathcal{B}$ recovers tuples $(R_1 = e(g_1, g_2)^t,\ \xi_1)$ and $(R_2 = e(g, g)^r,\ \xi_2)$ from the list $List^{H_3}$, and computes $m||\phi = (c_5 \oplus (c_6 \oplus \xi_2)) \oplus \xi_1$.

Challenge phase. $\mathcal{A}$ outputs m_0, m_1 and (ID_i^*, tpk_i^*) to $\mathcal{B}$. If $x' + \sum_{j \in \mathcal{V}} x_j \neq 4\bar{k}q_{sk}$, $\mathcal{B}$ aborts. Else, $\mathcal{B}$ proceeds.

- For original ciphertext, $\mathcal{B}$ chooses $\beta_1^*, \beta_2^*, \phi^* \in_R \{0,1\}^k$ and $b \in_R \{0,1\}$. It sets $c_1^* = (m_b||\phi^*) \oplus H_3(T)$, $c_2^* = C$, $c_3^* = C^{J(ID_i^*)}$, $c_4^* = C^{\varphi_3}$. $\mathcal{B}$ further issues (β_1^*, β_2^*) to H_2 to obtain r (i.e. φ_2), and sets $c_5^* = c_1^* \oplus (\beta_1^*||\beta_2^*)$, $c_6^* = (\beta_1^*||\beta_2^*) \oplus H_3(e(g,g)^r)$, $c_7^* = (tpk_{i,1}^* tpk_{i,2}^{*H_1(tpk_i^*)})^r$ and $c_8^* = g^{\omega r}$. It finally issues $(c_5^*, c_6^*, c_7^*, c_8^*)$ to H_5 to obtain φ_4^*, and sets $c_9^* = g^{\varphi_4^*}$. $\mathcal{B}$ outputs $C_2^* = (c_2^*, c_3^*, c_4^*, c_5^*, c_6^*, c_7^*, c_8^*, c_9^*)$.
- For updated ciphertext, $\mathcal{B}$ sets $C_2^{*(up)} = (c_2^*, c_3^*, c_4^*, c_5^*, c_6^*, c_{10}^*)$ as $c_2^* = C$, $c_3^* = C^{J(ID_i^*)}$, $c_4^* = C^{\varphi_3}$, $c_5^* = (m_b||\phi^*) \oplus H_3(T) \oplus (\beta_1^*||\beta_2^*)$, $c_6^* = (\beta_1^*||\beta_2^*) \oplus H_3(e(g,g)^r)$ and $c_{10}^* = e(g^r, tpk_{i,1}^* tpk_{i,2}^{*H_1(tpk_i^*)})$.

  The above ciphertexts imply $H_2(m_b, \phi^*) = c$. If $b = b'$, $\mathcal{B}$ guesses $T = e(g,g)^{abc}$; else, T is a random element in $\mathbb{G}_T$.

Phase 2. Same as Phase 1 (but with restrictions).

Guess phase. $\mathcal{A}$ outputs a guess bit $b' \in \{0,1\}$.

Probabilistic analysis:We analyze the probability by using the same method presented in the security analysis of Type-I adversary. Combining the non-abort probability in [46], we have $\epsilon' \geq \frac{1}{q_{H_3}}(AskH_3^*) \geq \frac{1}{q_{H_3}}\left(\frac{2\epsilon}{8(n+1)q_{sk}} - \frac{q_{H_2}+(q_{H_2}+q_{H_3})q_{dr}}{2^l} - \frac{2q_{dr}+q_{cu}}{q}\right)$.

The running time of $\mathcal{B}$ is bounded by

$$t' \leq t + O(1)(q_{H_1} + q_{H_2} + q_{H_3} + q_{H_4} + q_{H_5} + q_{sd} + q_{sk} + q_{cu} + q_{dr}) + t_e(2q_{sd} + 4q_{sk} + q_{H_2}(5q_{cu} + 5q_{dr}) + 5q_{cu} + q_{dr}) + 2t_p q_{cu}.$$

5.2 Efficiency Analysis

We analyze the efficiency of our mechanism as well as its comparison with [2] (the most efficient two-secret protection system but no revocability) and [20] (the most efficient single secret system with revocability) in terms of computational and communicational cost. We state that both [2] and [20] only achieve partial design goal of our system. The purpose of the comparison is to show that our system achieves more functionalities but not requiring a great increase of complexity. In other words, we would like to show our system is practical for real-world implementation.

Before the efficiency analysis, we define the notations we used in tables. We let $|\mathbb{G}|$ and $|\mathbb{G}_T|$ denote the bit-length of an element in groups $\mathbb{G}$ and $\mathbb{G}_T$, l denote the bit-length of security parameter, $|\mathbb{Z}_q|$ denote the bit-length of an element in $\mathbb{Z}_q$, $C_p, C_{\bar{e}}, C_e$ denote the computation cost of a bilinear pairing, an exponentiation in $\mathbb{G}_T$, and an exponentiation in $\mathbb{G}$, respectively. We suppose the three schemes share the same security parameter. Note by $\perp$ we mean non-applicable.

Theoretical comparison.We present the theoretical comparison in Tables 2 and 3 for computation and communication complexity, respectively. From Table 2, it can be seen that our system requires additional computation cost in security device generation and update, whereas others do not need any cost. This is because ours supports security device revocability. In ciphertext generation, our system does not require any pairings operation, and it is worth of mentioning that the second level ciphertext generation cost can be offloaded to a cloud server. Compared to [20] for other metrics, our system only requires slight extra cost; while we just need an additional pairing in ciphertext update. A similar phenomenon does exist in Table 3 in the sense that our system needs extra communication cost in delivery of security device. Except for this, our communication complexity is very closed to that of others.

Practical comparison. For real-time complexity test, we set the testbed to be: Pentium (R) G640 CPU, 3.33 GB RAM, 500

TABLE 4
Computation Comparison (Running Time in Second) II

Schemes	[2]	[20]	Ours
Secret Key Generation	0.007311	0.003123	0.007311
Security Device Generation	$\perp$	$\perp$	0.006164
Ciphertext Generation	0.049203	0.027515	first-level Ciph.: 0.010380 second-level Ciph.: 0.026214
Ciphertext Update	$\perp$	0.055677	0.065312
Device Update	$\perp$	$\perp$	0.006606
Data Recovery (From Original Ciph.)	0.018146	0.036948	0.049095
Data Recovery (From Updated Ciph.)	$\perp$	0.021569	0.032797

TABLE 5
Communication Comparison (Length of Size in Bit) II

Schemes	[2]	[20]	Ours
Secret Key Size	160	160	320
Security Device Size	$\perp$	$\perp$	640
Original Ciphertext Size	480	1,504	1,600
Updated Ciphertext Size	$\perp$	1,504	2,144
Cost in Ciphertext Update	$\perp$	320	320

G/5,400 rpm hard disk, C programming language, and Ubuntu 10.10 OS; pairing type is a with 160-bit group order (using a supersingular curve $Y^2 = X^3 + X$). In the experiment, to achieve the corresponding security level, we set l to be 160 bits, $|\mathbb{Z}_q| = 160$ bits, $|\mathbb{G}| = 160$ bits and $|\mathbb{G}_T| = 1,024$ bits, respectively. We show the running time comparison and practical communication comparison in Tables 4 and 5, respectively. The experimental results are somehow similar to the theoretical ones. Our system needs extra running time in device generation and update. In practice, if we make security device as a USB disk and deliver it to a registered user by mail/in person, there is no need for paying the price for communication cost in the metrics of "Security Device Size" and "Cost in Ciphertext Update". From Table 4, we see that our running time is nearly the same as that of [20], and meanwhile, our system outperforms [20] and [2] in encryption. In the communication cost, our scheme suffers from the largest price in "Updated Ciphertext Size" due to a reason that the scheme outputs a pairing in the update phase. However, we state that the price is only an approximately 50 percent increase from that of [20] in the same metric, which is an acceptable increment.

In summary, through the comparison, we can see that our scheme achieves two factors protection and security device revocability without requiring a great amount of additional complexity.

6 Conclusions

In this paper, we introduced a novel two-factor data security protection mechanism for cloud storage system, in which a data sender is allowed to encrypt the data with knowledge of the identity of a receiver only, while the receiver is required to use both his/her secret key and a security device to gain access to the data. Our solution not only enhances the confidentiality of the data, but also offers the revocability of the device so that once the device is revoked, the corresponding ciphertext will be updated automatically by the cloud server without any notice of the data owner. Furthermore, we presented the security proof and efficiency analysis for our system.

Acknowledgments

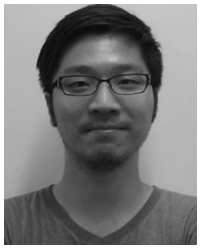
Joseph K. Liu is supported by National Natural Science Foundation of China (61472083). Kaitai Liang is supported by Privacy-Aware Retrieval and Modelling of Genomic Data, Academy of Finland (13283250). Willy Susilo is partially supported by the Australian Research Council Discovery Project ARC DP130101383. Yang Xiang is supported by the Australian Research Council Discovery Projects DP150103732, DP140103649, LP140100816 and LP120200266. This work is also supported by National Natural Science Foundation of China (61472083, 61402110, U1405255). Kaitai Liang is the corresponding author.

References

[1] A. Akavia, S. Goldwasser, and V. Vaikuntanathan, "Simultaneous hardcore bits and cryptography against memory attacks," in *Proc. 6th Theory Cryptography Conf.*, 2009, pp. 474–495.
[2] S. S. Al-Riyami and K. G. Paterson, "Certificateless public key cryptography," in *Proc. 9th Int. Conf. Theory Appl. Cryptol.*, 2003, pp. 452–473.
[3] M. H. Au, J. K. Liu, W. Susilo, and T. H. Yuen, "Certificate based (linkable) ring signature," in *Proc. Inf. Security Practice Experience Conf.*, 2007, pp. 79–92.
[4] M. H. Au, Y. Mu, J. Chen, D. S. Wong, J. K. Liu, and G. Yang, "Malicious KGC attacks in certificateless cryptography," in *Proc. 2nd ACM Symp. Inf., Comput. Commun. Security*, 2007, pp. 302–311.
[5] M. Blaze, G. Bleumer, and M. Strauss, "Divertible protocols and atomic proxy cryptography," in *Proc. Int. Conf. Theory Appl. Cryptographic Techn.*, 1998, pp. 127–144.
[6] A. Boldyreva, V. Goyal, and V. Kumar, "Identity-based encryption with efficient revocation," in *Proc. ACM Conf. Comput. Commun. Security*, 2008, pp. 417–426.
[7] D. Boneh, X. Ding, and G. Tsudik, "Fine-grained control of security capabilities," *ACM Trans. Internet Techn.*, vol. 4, no. 1, pp. 60–82, 2004.
[8] D. Boneh and M. Franklin, "Identity-based encryption from the Weil pairing," in *Proc. 21st Annu. Int. Cryptol. Conf.*, 2001, pp. 213–229.
[9] R. Canetti and S. Hohenberger, "Chosen-ciphertext secure proxy re-encryption," in *Proc. ACM Conf. Comput. Commun. Security*, 2007, pp. 185–194.
[10] H. C. H. Chen, Y. Hu, P. P. C. Lee, and Y. Tang, "NCCloud: A network-coding-based storage system in a cloud-of-clouds," *IEEE Trans. Comput.*, vol. 63, no. 1, pp. 31–44, Jan. 2014.
[11] S. S. M. Chow, C. Boyd, and J. M. G. Nieto, "Security-mediated certificateless cryptography," in *Proc. 9th Int. Conf. Theory Practice Public-Key Cryptography*, 2006, pp. 508–524.
[12] C.-K. Chu, S. S. M. Chow, W.-G. Tzeng, J. Zhou, and R. H. Deng, "Key-aggregate cryptosystem for scalable data sharing in cloud storage," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 2, pp. 468–477, Feb. 2014.
[13] C.-K. Chu and W.-G. Tzeng, "Identity-based proxy re-encryption without random oracles," in *Proc. 10th Int. Con. Inf. Security*, 2007, pp. 189–202.
[14] R. Cramer and V. Shoup, "Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack," *SIAM J. Comput.*, vol. 33, no. 1, pp. 167–226, Jan. 2004.
[15] Y. Dodis, Y. T. Kalai, and S. Lovett, "On cryptography with auxiliary input," in *Proc. 41st Annu. ACM Symp. Theory Comput.*, 2009, pp. 621–630.
[16] Y. Dodis, J. Katz, S. Xu, and M. Yung, "Key-insulated public key cryptosystems," in *Proc. Int. Conf. Theory Appl. Cryptographic Techn.*, 2002, pp. 65–82.
[17] Y. Dodis, J. Katz, S. Xu, and M. Yung, "Strong key-insulated signature schemes," in *Proc. Int. Conf. Theory Appl. Cryptographic Techn.*, 2003, pp. 130–144.
[18] L. Ferretti, M. Colajanni, and M. Marchetti, "Distributed, concurrent, and independent access to encrypted cloud databases," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 2, pp. 437–446, Feb. 2014.
[19] C. Gentry, "Certificate-based encryption and the certificate revocation problem," in *Proc. Int. Conf. Theory Appl. Cryptographic Techn.*, 2003, pp. 272–293.
[20] M. Green and G. Ateniese, "Identity-based proxy re-encryption," in *Proc. 5th Int. Conf. Appl. Cryptography Netw. Security*, 2007, pp. 288–306.
[21] H. Guo, Z. Zhang, J. Zhang, and C. Chen, "Towards a secure certificateless proxy re-encryption scheme," in *Proc. 7th Int. Conf. Provable Security*, 2013, pp. 330–346.
[22] G. Hanaoka, Y. Hanaoka, and H. Imai, "Parallel key-insulated public key encryption," in *Proc. 10th Int. Conf. Practice Theory Public-Key Cryptography*, 2006, pp. 105–122.
[23] Y. H. Hwang, J. K. Liu, and S. S. M. Chow, "Certificateless public key encryption secure against malicious kgc attacks in the standard model," *J. UCS*, vol. 14, no. 3, pp. 463–480, 2008.

[24] K. Liang, Z. Liu, X. Tan, D. S. Wong, and C. Tang, "A CCA-secure identity-based conditional proxy re-encryption without random oracles," in *Proc. 15th Int. Conf. Inf. Security Cryptol.*, 2012, pp. 231–246.
[25] B. Libert, J.-J. Quisquater, and M. Yung, "Parallel key-insulated public key encryption without random oracles," in *Proc. 10th Int. Conf. Public Key Cryptography*, 2007, pp. 298–314.
[26] B. Libert and D. Vergnaud, Adaptive-id secure revocable identity-based encryption, in *Proc. Cryptographers Track RSA Conf.*, 2009, pp. 1–15.
[27] B. Libert and D. Vergnaud, "Unidirectional chosen-ciphertext secure proxy re-encryption," *IEEE Trans. Inf. Theory*, vol. 57, no. 3, pp. 1786–1802, Mar. 2011.
[28] J. K. Liu, M. H. Au, and W. Susilo, "Self-generated-certificate public key cryptography and certificateless signature/encryption scheme in the standard model: Extended abstract," in *Proc. 2nd ACM Symp. Inf., Comput. Commun. Security*, 2007, pp. 273–283.
[29] J. K. Liu, J. Baek, W. Susilo, and J. Zhou, "Certificate-based signature schemes without pairings or random oracles," in *Proc. 11th Int. Conf. Inf. Security*, 2008, pp. 285–297.
[30] J. K. Liu, J. Baek, and J. Zhou, "Certificate-based sequential aggregate signature," in *Proc. 2nd ACM Conf. Wireless Netw. Security*, 2009, pp. 21–28.
[31] J. K. Liu, F. Bao, and J. Zhou, "Short and efficient certificate-based signature," in *Proc. Netw. Workshops*, 2011, pp. 167–178.
[32] J. K. Liuand D. S. Wong, "Solutions to key exposure problem in ring signature," *Int. J. Netw. Security*, vol. 6, no. 2, pp. 170–180, 2008.
[33] J. K. Liu and J. Zhou, "Efficient certificate-based encryption in the standard model," in *Proc. 6th Int. Conf. Security Cryptography Netw.*, 2008, pp. 144–155.
[34] S. Luo, Q. Shen, and Z. Chen, "Fully secure unidirectional identity-based proxy re-encryption," in *Proc. 14th Int. Conf. Inf. Security Cryptol.*, 2011, pp. 109–126.
[35] M. Mambo and E. Okamoto, "Proxy cryptosystems: Delegation of the power to decrypt ciphertexts," *IEICE Trans.*, vol. E80-A, no. 1, pp. 54–63, 1997.
[36] T. Matsuo, "Proxy re-encryption systems for identity-based encryption," in *Proc. 1st Int. Conf. Pairing-Based Cryptography*, 2007, pp. 247–267.
[37] M. Naor and G. Segev, "Public-key cryptosystems resilient to key leakage," in *Proc. 29th Annu. Int. Cryptol. Conf.*, 2009, pp. 18–35.
[38] A. Sahai, H. Seyalioglu, and B. Waters, "Dynamic credentials and ciphertext delegation for attribute-based encryption," in *Proc. 32nd Annu. Cryptol. Conf. Adv. Cryptol.*, 2012, pp. 199–217.
[39] J. H. Seo and K. Emura, "Efficient delegation of key generation and revocation functionalities in identity-based encryption," in *Proc. Cryptographers Track RSA Conf.*, 2013, pp. 343–358.
[40] J. Shao and Z. Cao, "Multi-use unidirectional identity-based proxy re-encryption from hierarchical identity-based encryption," *Inf. Sci.*, vol. 206, pp. 83–95, 2012.
[41] Q. Tang, P. H. Hartel, and W. Jonker, "Inter-domain identity-based proxy re-encryption," in *Information Security and Cryptology*. Berlin, Germany: Springer-Verlag, 2008, pp. 332–347.
[42] V. Varadharajan and U. K. Tupakula, "Security as a service model for cloud environment," *IEEE Trans. Netw. Service Manage.*, vol. 11, no. 1, pp. 60–75, Mar. 2014.
[43] C. Wang, S. S. M. Chow, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for secure cloud storage," *IEEE Trans. Comput.*, vol. 62, no. 2, pp. 362–375, Feb. 2013.
[44] C. Wang, Q. Wang, K. Ren, N. Cao, and W. Lou, "Toward secure and dependable storage services in cloud computing," *IEEE Trans. Services Comput.*, vol. 5, no. 2, pp. 220–232, Apr.–Jun. 2012.
[45] H. Wang, "Proxy provable data possession in public clouds," *IEEE Trans. Services Comput.*, vol. 6, no. 4, pp. 551–559, Oct.–Dec. 2013.
[46] B. Waters, "Efficient identity-based encryption without random oracles," in *Proc. 24th Annu. Int. Conf. Theory Appl. Cryptographic Techn.*, 2005, pp. 114–127.
[47] K. Yang, X. Jia, K. Ren, B. Zhang, and R. Xie, "DAC-MACS: Effective data access control for multiauthority cloud storage systems," *IEEE Trans. Inf. Forensics Security*, vol. 8, no. 11, pp. 1790–1801, Nov. 2013.
[48] W.-S. Yap, S. S. M. Chow, S.-H. Heng, and B.-M. Goi, "Security mediated certificateless signatures," in *Proc. 5th Int. Conf. Appl. Cryptography Netw. Security*, 2007, pp. 459–477.
[49] Y. Zhu, G.-J. Ahn, H. Hu, S. S. Yau, H. G. An, and C. Hu, "Dynamic audit services for outsourced storages in clouds," *IEEE Trans. Services Comput.*, vol. 6, no. 2, pp. 227–238, Apr.–Jun. 2013.

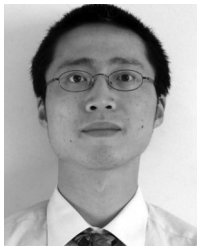
Joseph K. Liu received the PhD degree in information engineering from the Chinese University of Hong Kong in July 2004, specializing in cyber security, protocols for securing wireless networks, privacy, authentication, and provable security. He is currently a senior lecturer at Monash University, Australia. His current technical focus is particularly cyber security in the cloud computing paradigm, smart city, lightweight security, and privacy enhanced technology. He has published more than 80 referred journal and conference papers and received the Best Paper Award from ESORICS 2014. He has served as the program chair of ProvSec 2007, 2014, and as the program committee of more than 35 international conferences.



Kaitai Liang received the PhD degree from the Department of Computer Science, City University of Hong Kong. He is currently a postdoctoral researcher at the Department of Computer Science, Aalto University, Finland. His research interest is applied cryptography; in particular, cryptographic protocols, encryption/signature, and RFID. He is also interested in cybersecurity, such as network security, big data security, privacy enhanced technology, and security in cloud computing.

Willy Susilo received the PhD degree in computer science from the University of Wollongong, Australia. He is a professor and the head in the School of Computing and Information Technology, University of Wollongong, Australia. He is also the director in the Centre for Computer and Information Security Research, University of Wollongong. He received the prestigious ARC Future fellow by the Australian Research Council. His main research interests include cloud security, cryptography and information security. He has served as a program committee member in major international conferences.

Jianghua Liu received the BS degree from the Department of Electronic Information Science and Technology, Hainan Normal University, China, in 2013. He is currently a graduate at the Fujian Provincial Key Laboratory of Network Security and Cryptology, School of Mathematics and Computer Science, Fujian Normal University, China. His research interests include cryptography and information security.



Yang Xiang received the PhD degree in computer science from Deakin University, Australia. He is currently a full professor at the School of Information Technology, Deakin University. He is the director in the Network Security and Computing Lab (NSCLab) and the associate head of School (Industry Engagement). His research interests include network and system security, distributed systems, and networking. In particular, he is currently leading his team developing active defense systems against large-scale distributed network attacks. He has published more than 200 research papers in many international journals and conferences, such as *IEEE Transactions on Computers*, *IEEE Transactions on Parallel and Distributed Systems*, *IEEE Transactions on Information Security and Forensics*, and *IEEE Journal on Selected Areas in Communications*. He serves as the associate editor of the *IEEE Transactions on Computers*, *IEEE Transactions on Parallel and Distributed Systems*, *Security and Communication Networks* (Wiley), and the editor of *Journal of Network and Computer Applications*. He is the coordinator, Asia for IEEE Computer Society Technical Committee on Distributed Processing (TCDP). He is a senior member of the IEEE.