CloudKeyBank: Privacy and Owner Authorization Enforced Key Management Framework

Xiuxia Tian, Ling Huang, Tony Wu, Xiaoling Wang, *Member, IEEE*, and Aoying Zhou, *Member, IEEE*

Abstract—Explosive growth in the number of passwords for web based applications and encryption keys for outsourced data storage well exceeds the management limit of users. Therefore, outsourcing keys (including passwords and data encryption keys) to professional password managers (honest-but-curious service providers) is attracting the attention of many users. However, existing solutions in a traditional data outsourcing scenario are unable to simultaneously meet the following three security requirements for keys outsourcing: 1) Confidentiality and privacy of keys; 2) Search privacy on identity attributes tied to keys; 3) Owner controllable authorization over his/her shared keys. In this paper, we propose CloudKeyBank, the first unified key management framework that addresses all the three goals above. Under our framework, the key owner can perform privacy and controllable authorization enforced encryption with minimum information leakage. To implement CloudKeyBank efficiently, we propose a new cryptographic primitive named Searchable Conditional Proxy Re-Encryption (SC-PRE) which combines the techniques of Hidden Vector Encryption (HVE) and Proxy Re-Encryption (PRE) seamlessly, and propose a concrete SC-PRE scheme based on existing HVE and PRE schemes. Our experimental results and security analysis show the efficiency and security goals are well achieved.

Index Terms—SC-PRE, search privacy, key management, keys outsourcing

1 Introduction

With the rapid deployment of web applications such as online banking, shopping, social networks and data storage (e.g., Amazon S3 and GoogleDrive), managing the ever-growing number of passwords and data encryption keys is becoming a big burden for many users. For example, our recent survey from UC Berkeley students in Fig. 1b shows that each student has to remember an average of 23 different application accounts that are shown in Fig. 1a. To remember them, 85 percent students admit that their passwords are almost the same except for bank and email accounts. However, the weak and shared passwords across accounts make them easy to be compromised, which in turn leaks more passwords related to private and sensitive data.

To resolve the contradiction between easy of management and secure passwords, Fig. 1b shows that around 14 percent of students use web based password managers to manage their passwords, which enables them to choose many complex and unique passwords for different application sites. The success of web based password managers such as Lastpass,[1] with over a million users in 2011, PasswordBox,[2] with over a million users in less than three months in 2013, and other similar tools,[3, 4, 5] demonstrate that users have a strong will to outsource their passwords to a centralized key management provider who can relieve them from the overwhelming burden of memorization and management.

As pointed out in the survey,[6] privacy problems are the main concern of cloud users (more than 85 percent) in outsourced data storage, which is also true for outsourced keys storage. More than 90 percent of students shown in Fig. 1b are concerned about the privacy of their keys, which mainly involves two situations: 1) they do not fully trust the service providers because there is no governance about how keys can be used by them and whether the key owner can actually control their keys on their own; 2) they trust the service providers, but keys could be disclosed if there exists an misbehaving internal employee or broken server. Therefore encrypting key tuples just like encrypting normal data tuples before outsourcing seems to be a promising solution to maintaining trust and ensuring the key owners' control over their own privacy.

However, different from the normal data tuples, key tuples in a key database shown in Fig. 2 are composed of attributes (e.g., identity attributes, key attributes) with different privacy requirements. The privacy requirements of key attributes in the Key attribute group is higher than that

- *X. Tian is with the College of Computer Science and Technology, Shanghai University of Electric Power, Shanghai 200090, China. E-mail: xxtian@fudan.edu.cn.*
- *L. Huang is with the EECS, Intel Labs, and the Intel Science and Technology Center on Secure Computing, UC Berkeley, Berkeley, CA 94720. E-mail: ling.huang@intel.com.*
- *T. Wu is with the EECS, UC Berkeley, Berkeley, CA 94720. E-mail: tony.wu@berkeley.edu.*
- *X. Wang is with the Software Engineering Institute, East China Normal University, Shanghai, China. E-mail: xlwang@sei.ecnu.edu.cn.*
- *A. Zhou is with the Institute for Data Science and Engineering, East China Normal University, Shanghai, China. E-mail: ayzhou@sei.ecnu.edu.cn.*

Manuscript received 19 Nov. 2014; revised 16 May 2015; accepted 6 July 2015. Date of publication 16 July 2015; date of current version 3 Nov. 2015. Recommended for acceptance by H. T. Shen.
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/TKDE.2015.2457903

1. https://lastpass.com
2. https://www.passwordbox.com
3. https://www.my1login.com/content/index.php
4. http://www.needmypassword.com
5. http://www.roboform.com
6. http://www.fujitsu.com/downloads/SOL/fai/reports/fujitsu_personal-data-in-the-cloud.pdf

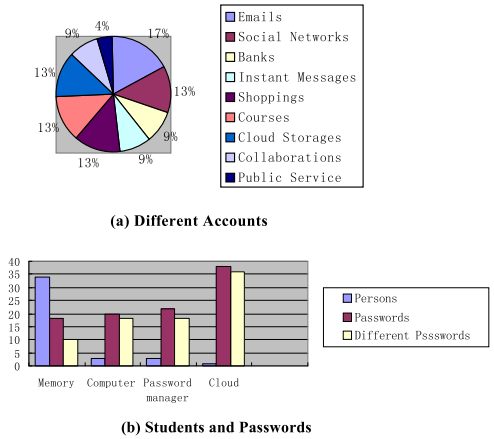


Fig. 1. Passwords distribution for different applications.

of identity attributes in the Search attribute group, because the resulted information leakage by the former is much larger than that by the latter. For example, the leakage of identity attributes {*www.uhs.berkeley.edu*, *John Hu*, 5106128976} will only expose the personal data (identity attributes in the Search attribute group) of the key owner *JohnHu*, which violates the identity unlinkability privacy of users [1]; the leakage of data encryption key $rft78390$ of key owner *JohnHu* means the disclosure of more personal data (related profiles and outsourced encrypted data), which not only violates the confidentiality and privacy of keys but also breaches the privacy of related profiles and outsourced data due to the lost control on data encryption keys. Therefore, based on the different sensitive attributes in key tuples, we identify that the following three critical security requirements need to be achieved for secure keys outsourcing.

- *First*. The keys have high sensitivity and need to be hidden from the honest-but-curious service provider and malicious attackers. This involves *confidentiality and privacy of keys*—only the authorized users can derive the shared keys of the key owner through the authorized decryption computation.
- *Second*. The keys are always stored with many sensitive identity attributes (in the Search attribute group instead of the access control policy (ACP)) of key owners and are searched based on them. This involves *search privacy on identity attributes*—the honest-but-curious key service provider can not derive any identity attribute tied with keys from the submitted search query, but can evaluate the query against the encrypted key database correctly.
- *Third*. The keys have strong ownership because they are used to protect many other sensitive information of the key owner. This involves *owner controllable authorization including key authorization and query authorization*—only the key owner can specify and control in a fine-grained way who has the rights to access his/her shared keys through authorization on key attributes (key authorization) and authorization on submitted search query (query authorization).

However, most of the currently proposed approaches [12], [13], [14], [15], [17], [18], [21], [25] in traditional data outsourcing scenario just support one or two of the three

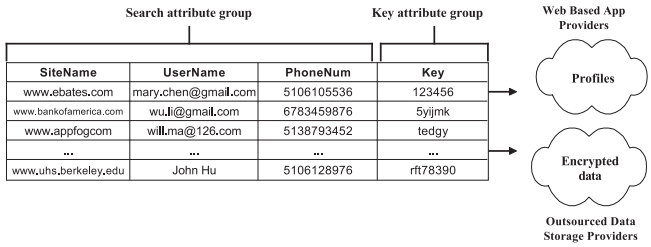


Fig. 2. Key tuples in key database.

identified security requirements to some extent. Encrypting key tuples like encrypting data tuples in an all or nothing way [12], [13], [21] can guarantee the confidentiality and privacy of keys, but does not consider the key authorization and the different privacy requirements of sensitive attributes in key tuples. Encrypting search keywords (identity attributes in key tuples) based on searchable symmetric encryption (SSE) [14], [15] or hierarchical predicate encryption [26] can guarantee the search privacy on key tuples, but does not consider the key authorization and the dependence relation between identity attributes and key attributes. Encrypting identity attributes and related identity conditions in the access control policy [17], [18], [25] achieves the identity and related condition privacy of users, but does not consider key authorization based on the identity attributes in key tuples and query authorization on submitted search query. Therefore, in outsourced keys storage, a challenging problem is to find an encryption scheme which can encrypt the key tuples in a way that the different privacy requirements of sensitive attributes in the key tuples can be satisfied.

To efficiently solve the identified secure problems above, to the best of our knowledge, we are the first to explore and present CloudKeyBank, an unified key management framework with enforced privacy and owner controllable authorization described in Section 2 and constructed in Section 4. The realization of CloudKeyBank framework is mainly through the following contributions:

1) To implement the proposed CloudKeyBank framework, we propose a new cryptographic primitive named Searchable Conditional Proxy Re-Encryption (SC-PRE) described in Section 3.6, which combines the techniques of hidden vector encryption (HVE) and proxy re-encryption (PRE) seamlessly. We also propose a concrete SC-PRE scheme based on the existing schemes described in Section 4. SC-PRE efficiently solves the challenge of performing a key tuple encryption so that the different privacy requirements of attributes are achieved in one encryption scheme. In SC-PRE we do not encrypt each key tuple t_i as a whole, but first divide each tuple into multiple attribute groups, and then encrypt different attribute groups in terms of the dependence relation between attribute groups. For example shown in Fig. 2, each key tuple is divided into two groups, one is the Search attribute group, denoted as a vector $\vec{x_i}$, the other is the Key attribute group, denoted as a vector $\vec{k_i}$, i.e. $t_i = \{\vec{x_i}, \vec{k_i}\}$. Assume $t_1 = \{\vec{x_1}, \vec{k_1}\}$, where $\vec{x_1} = \{www.ebates.com, mary.chen@gmail.com, 5106105536\}$, $\vec{k_1} = \{123456\}$.

The dependence relation between $\vec{x_1}$ and $\vec{k_1}$ is that $\vec{k_1}$ is searched based on $\vec{x_1}$.

2) To achieve the minimum information leakage in the process of privacy and owner controllable authorization enforcement, we introduce dual authorization tokens including the Query token and the Delegation token. By using the dual authorization tokens the key owner can encrypt the Key attribute group in such a way that only the user with the appropriate tokens can gain access to the shared key of key owner. The CloudKeyBank provider who stores the encrypted key database will not be able to see the content of the key at anytime even if he/she knows all Delegation tokens of the delegated users. Both the delegated user and the CloudKeyBank provider can not derive the private key of key owner from the submitted Query token, but the CloudKeyBank provider still can perform efficient search queries by evaluating the Query token against each encrypted key tuple.
3) To demonstrate the feasibility of CloudKeyBank framework, we analyze its security strength and evaluate its performance. Security analysis proves that the CloudKeyBank can efficiently support search privacy, key privacy and owner controllable authorization. Performance evaluation results demonstrate that our CloudKeyBank has appropriate communication and computation overhead, as well as response time.

The rest of the paper is organized as follows. We first give the description of our CloudKeyBank system architecture and attack surface in Section 2. Then Section 3 describes in detail the proposed cryptographic primitive SC-PRE, the definitions and preliminary tools used to construct our CloudKeyBank framework. We present the concrete construction of our SC-PRE scheme and its use in CloudKeyBank framework in Section 4, and then analyze the security and evaluate the performance in Section 5 and Section 6 in sequence. Section 7 discusses the related work and compares them with our solution. Finally, Section 8 concludes the paper and provides future research directions.

2 System Architecture and Attack Surface

In this section we first describe the CloudKeyBank system architecture, and then based on which we give the attack surface.

2.1 CloudKeyBank System Architecture

As shown in Fig. 3, CloudKeyBank architecture consists of four entities: Key owner, CloudKeyBank provider, Trusted client and Users.

1) *Key owner*. Key owner can be the password owner or data encryption key owner who outsources his/her key database (Key DB) to the CloudKeyBank provider. After that the encrypted key database (EDB) stored in CloudKeyBank provider can be accessed anywhere and anytime with minimum information leakage such as the size of Key DB. The key owner mainly completes the following three tasks: a) Constructing the customized access control policy (ACP) in terms of his/her practical keys sharing requirements; b) Depositing Key DB by using DepositKey protocol under the support of ACP; c) Distributing authorized Query tokens to the delegated user based on the user's registered information such as the wanted query and physical identity.

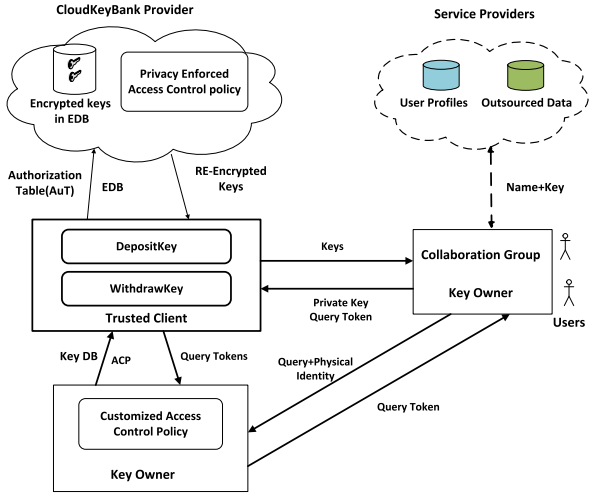


Fig. 3. CloudKeyBank system architecture.

2) *CloudKeyBank provider*. CloudKeyBank provider can be any professional password manager such as LastPass who provides privacy enforced access control on EDB. The CloudKeyBank provider mainly completes the following two tasks: a) To enforce the privacy of identity attributes in the Search attribute group, he/she can perform search query directly by evaluating the submitted Query token against the encrypted key tuples in EDB; b) To enforce the key authorization he/she can transform an encrypted key into the authorized re-encrypted key under the corresponding Delegation token stored in authorization table (AuT).
3) *Trusted client*. Trusted client is the primary privacy enforced component in CloudKeyBank framework. It mainly consists of two protocols: DepositKey and WithdrawKey. DepositKey protocol described in Section 4.2 provides Key DB encryption, token generation (including Query token and Delegation token). Withdrawkey protocol described in Section 4.3 provides the re-encryption of encrypted keys and the decryption of re-encrypted keys.
4) *Users*. There are two kinds of users in CloudKeyBank framework: Key owner and Collaboration group. Key owner corresponds to an individual user who deposits all his keys to CloudKeyBank provider and access them by himself. Collaboration group corresponds to a group of users where the key owner can share his/her keys with other users within the same collaboration group. By submitting the private key and authorized Query token, a delegated user can withdraw an authorized key by using WithdrawKey protocol under the support of privacy enforced access control policy (i.e. AuT in our solution).

As is shown in Fig. 3, Service providers, such as web based application providers and outsourced data storage

providers, are the part in the dotted cloud. In this paper we assume that the secure communication (protecting the security of transferred name and key) between the users and the service providers is based on HTTPS and outsourced data stored at these service providers are encrypted under data encryption keys. In this paper we only focus on how to achieve the privacy and owner controllable authorization of keys stored in CloudKeyBank provider, while do not consider the privacy of outsourced data protected by the keys.

2.2 Attack Surface

In CloudKeyBank architecture, based on outsourced key tuple $t_i = \{\vec{x_i}, \vec{k_i}\}$, we focus on the attacks on sensitive key attributes $\vec{k_i}$ and sensitive identity attributes $\vec{x_i}$. Corresponding to the attacks there mainly exist the following three kinds of attackers: 1) *Attacker1*: CloudKeyBank provider is a honest-but-curious inside attacker who is curious about key values in $\vec{k_i}$ (Key confidentiality) and identity values in $\vec{x_i}$ (Identity confidentiality and Linkability privacy), but can honestly provide efficient database operations given minimum information leakage. The minimum information leakage may include leakage on the total size of the Key DB and leakage on the random tuple identifier (e.g. the identifier ind_i for tuple to speed up the query efficiency), but never the direct exposure of plaintext keys or identities; 2) *Attacker2*: The malicious user is an outside attacker who wants to derive keys of the delegated user and thus impersonate him/her to do illegal actions (Key privacy and Key authorization); 3) *Attacker3*: The CloudKeyBank provider or the attacker in the middle may derive the private intent of the user from his/her submitted search query (Search privacy); 4) *Attacker4*: The malicious user may impersonate the legal user to submit search query in terms of the known background knowledge such as the possible search keywords (Query authorization).

3 Definition and Preliminaries

In this section we first describe database notations, equality predicate and intricate mathematics problems that our solution is based off. Then we present the basic encryption blocks HVE and PRE, and finally describe how we use them to build our new cryptographic primitive SC-PRE.

3.1 Database Notations

We follow some formal notation used in [14], [15], but with our modifications discussed below. Assume $[\ell]$ denotes the set $\{1, 2, \ldots, \ell\}$. For a vector $\vec{v}$ we denotes $|\vec{v}|$ as the dimension (length) of $\vec{v}$. Therefore for $i \in [|\vec{v}|]$, $v[i]$ denotes the ith component of $\vec{v}$. Assume function $f(m) \rightarrow y$ denotes that y is the output of a randomized polynomial-time algorithm f on input m. If f is randomized then y is a random variable. Based on the dimension definition of vector $\vec{v}$, we list the following notations used later in this paper. Assume n is the number of elements in a vector set $\mathbf{w}$, i.e. $|\mathbf{w}| = n$, then we denote $\mathbf{w} = \{\vec{w_1}, \vec{w_2}, \ldots, \vec{w_n}\}$, where $\vec{w_i} = (w_i[1], w_i[2], \ldots, w_i[|\vec{w_i}|])$, $|\vec{w_i}|$ is the number of components in vector $\vec{w_i}$, and denote $\mathbf{x} = \{\vec{x_1}, \vec{x_2}, \ldots, \vec{x_n}\}$, where $\vec{x_i} = (x_i[1], x_i[2], \ldots, x_i[|\vec{x_i}|])$, $|\vec{x_i}|$ is the number of components in vector $\vec{x_i}$.

As is shown in Fig. 1, assume a Key DB (denoted as DB) consists of a collection of d key tuples, i.e $DB = \{t_1, t_2, \ldots, t_d\}$. Each key tuple t_i is composed of a set of keywords W_i and K_i, $1 \leq i \leq d$. W_i is a set of identity attribute-value pairs in the Search attribute group and K_i is a set of key attribute-value pairs in the Key attribute group. K_i may be $K_i = \{pw_i\}$ for a web based application or $K_i = \{e_i, ind_i\}$ for outsourced data storage application, where pw_i is a password and e_i is a randomly chosen symmetric data encryption key, and ind_i is the index identifier for encrypted key tuple stored on some Service provider. Assume λ is the security parameter used to parameterize the stored keys and assume the identifiers and keywords are bit strings. Therefore Key DB can be denoted as $DB = \{t_1, t_2, \ldots, t_d\} = (W_i, K_i)_{i=1}^{d}$ a list of keyword-set pairs where $W_i \subseteq \{0,1\}^*$ and $K_i \subseteq \{0,1\}^{\lambda}$. Assume $W = \bigcup_{i=1}^{d} W_i$ and $K = \bigcup_{i=1}^{d} K_i$. Let $*$ be a special symbol denoting a wildcard keyword. The wildcard $*$ denotes that the keyword related to $*$ is not involved with any attribute-value pairs, so we give the following two notations $W_* = W \cup \{*\}$ and $K_* = K \cup \{*\}$.

3.2 Equality Predicate

Assume there exist any two ℓ-dimensional vectors $\vec{x} \in \mathbf{x}$ and $\vec{w} \in \mathbf{w}$: *encryption vector* $\vec{x} = (x_i[1], \ldots, x_i[\ell]) \in W^{\ell}$ and *token vector* $\vec{w} = (w_i[1], \ldots, w_i[\ell]) \in (W_*)^{\ell}$.

In the encryption phase, the sender encrypts a message $m \in M$ to ciphertext $c \in C$ under *encryption vector* $\vec{x}$ and the receiver computes a *token* for *token vector* $\vec{w}$. Assume P is a boolean formula which denotes the equality predicate. Then the *token* can be used to decrypt the ciphertext c if and only if $\vec{x}$ and $\vec{w}$ match in sequence shown in Equation (1):

$$P_{\vec{w}}(\vec{x}) = \begin{cases} 1 & \text{if for all } j \in |\vec{w}|,\ w[j] \neq * \text{ and } w[j] = x[j], \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

A *search query* denoted as $P_{\vec{w}}$ is specified by *token vector* $\vec{w} \in W_*$ and the boolean formula P on $\vec{w}$. $DB(P_{\vec{w}})$ denotes the set of tuples in Key DB which satisfies $P_{\vec{w}}$. $\vec{k} \subseteq DB(P_{\vec{w}})$ iff Equation (1) succeeds where $\vec{k}$ is a vector from vector set $\mathbf{k} = \{\vec{k_1}, \vec{k_2}, \ldots, \vec{k_n}\}$, $\vec{k_i} = (k_i[1], k_i[2], \ldots, k_i[|\vec{k_i}|])$ is a vector corresponding to key attributes in the ith key tuple t_i.

3.3 Bilinear Map and Intricate Mathematics Problem

The construction and security of our proposed SC-PRE are based on the properties of bilinear map and the intricate mathematics problem of extended decisional bilinear Diffie-Hellman problem (eDBDH).

Bilinear map. For simplicity, we introduce the symmetric bilinear maps following notations in [10]. Let G and G_T be two cyclic multiplication groups of prime order p. g is a generator of G. The bilinear map $e : G \times G \rightarrow G_T$ is a function with the following properties:

1) *Bilinearity*: For any $u, v \in G$ and $a, b \in Z_p^*$, the equation $e(u^a, v^b) = e(u, v)^{ab}$ succeeds.
2) *Nondegeneracy*: $e(g, g) \neq 1$, such that if g is a generator of G, $e(g, g)$ is the generator of G_T.
3) *Computability*: There exists an efficient algorithm to compute bilinear map e.

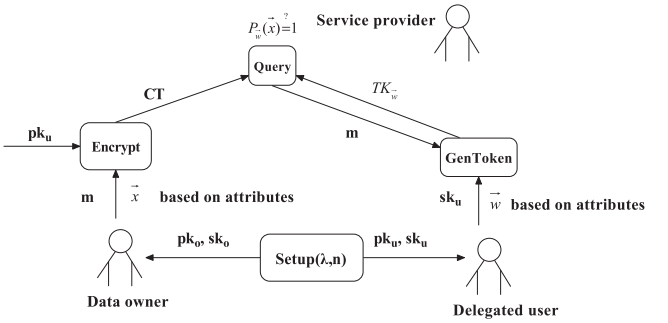


Fig. 4. An interaction among different polynomial algorithms of HVE.

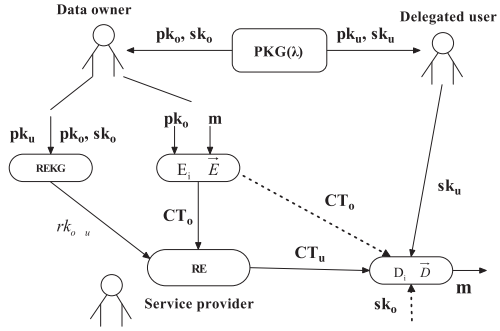


Fig. 5. An interaction among different polynomial algorithms of PRE.

eDBDH. As is described in [28], given $(g, g^a, g^b, g^c, e(g,g)^{bc^2}, R) \in G^4 \times G_T^2$, it is difficult to determine whether $R = e(g,g)^{abc}$ or R is random in G_T, $a, b, c \in Z_p^*$ are chosen randomly.

3.4 Hidden Vector Encryption (HVE)

HVE [6] was introduced to solve the problem of performing search query on encrypted data without leaking the search predicates or hidden attributes. As is defined in Definition 3.1, HVE [6], [10], [11] is a type of predicate encryption where two vectors $\vec{x}$ and $\vec{w}$ over attributes are chosen, $\vec{x}$ is associated with a ciphertext, and $\vec{w}$ is associated with a token. The ciphertext matches the token if and only if the two vectors meet with Equation (1).

Definition 3.1 (HVE). *As shown in Fig. 4, HVE consists of the following four polynomial time algorithms:* ***Setup***, ***Encrypt***, ***GenToken*** *and* ***Query****:*

- ***Setup***(λ, n). *In the system parameter generation phase, the trusted authorization center takes as input a security parameter λ and vector dimension n (the number of attributes). It outputs a public/private key pair (pk, sk) such as (pk_o, sk_o) for data owner and (pk_u, sk_u) for delegated user.*
- ***Encrypt***$(pk_u, \vec{x}, m)$. *In the encryption phase, data owner takes as input the delegated user's public key pk_u, a chosen vector $\vec{x} \in \mathbf{x}$ and a message $m \in M$. It outputs a ciphertext $CT \in C$.*
- ***GenToken***$(sk_u, \vec{w})$. *In the token generation phase, delegated user takes as input his private key sk_u and a chosen vector $\vec{w} \in \mathbf{w}$. It outputs a token $TK_{\vec{w}}$ to evaluate $P_{\vec{w}}$ from ciphertext CT.*
- ***Query***$(TK_{\vec{w}}, CT)$. *Taking as input the token $TK_{\vec{w}}$ for $\vec{w}$ and a ciphertext CT, it outputs a message m if $P_{\vec{w}} = 1$, otherwise outputs $\perp$ and terminates the process.*

3.5 Proxy Re-Encryption (PRE)

As is described in [28], [31], PRE is a type of public key encryption which allows a service provider to transform or re-encrypt a ciphertext computed under the data owner's public key into a re-ciphertext that can be opened or decrypted by the receiver's private key.

Definition 3.2 (PRE). *As shown in Fig. 5, PRE is denoted as a tuple $(PKG, REKG, \vec{E}, RE, \vec{D})$ in which the corresponding five polynomial time algorithms are described as follows:*

- ***PKG***(λ). *On input a security parameter λ, the key generation algorithm (PKG) outputs a public/private key pair (pk, sk) such as (pk_o, sk_o) for data owner and (pk_u, sk_u) for delegated user.*
- $\vec{E}(pk_o, m)$. *In the encryption phase, data owner takes as input his/her public key pk_o and a message $m \in M$, for all $E_i \in \vec{E}$, the encryption algorithm (E_i) outputs a ciphertext $CT_o \in C$.*
- $\vec{D}(sk_o, CT_o)$. *In the decryption phase, On input the private key sk_o of data owner and ciphertext CT_o, there exists a decryption algorithm $D_i \in \vec{D}$ which outputs the message m. This is the normal public key encryption/decryption denoted as the dotted line with CT_o and sk_o.*
- ***REKG***$(pk_o, sk_o, pk_u, sk_u^*)$. *On input the public/private key pairs of the data owner and the delegated user, the re-encryption algorithm (REKG) outputs a re-encryption key $rk_{o\to u}$ for the service provider. sk_u^* means that it is not necessary for the delegated user to be involved in the generation of re-encryption key $rk_{o\to u}$.*
- ***RE***$(rk_{o\to u}, CT_o)$. *Taking as input the re-encryption key $rk_{o\to u}$ and ciphertext CT_o, the re-encryption algorithm(RE) outputs re-ciphertext CT_u.*

CT_u can be decrypted to m under the private key sk_u of the delegated user. This is the proxy re-encryption/decryption denoted as the solid line with CT_u and sk_u.

3.6 New Cryptographic Primitive SC-PRE

From the definitions of HVE in Section 3.4 and PRE in Section 3.5 knowing that any one of them does not be used directly to guarantee the different privacy requirements of sensitive attributes in a key tuple.

- HVE can be used to guarantee the search privacy on identity attributes, but it does not support the key privacy and key authorization.
- PRE can be used to guarantee the confidentiality of keys and key authorization, but it does not support the search privacy on identity attributes and query authorization on submitted query.

The naive motivation is to use PRE on key attributes first and use HVE on identity attributes second, but it will result in the following efficiency problems: 1) the increasing burden of users by remembering at least two pairs of public/private parameters; 2) the increasing security and cost caused by at least two times of public/private key distribution; 3) the increasing communication and computation cost

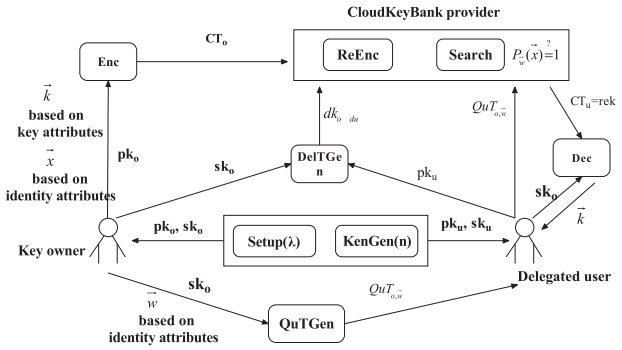


Fig. 6. An interaction among different polynomial algorithms of SC-PRE.

between the key owner and the delegated users by encrypting and decrypting at least two times. Therefore, it is necessary to design a new encryption scheme to solve the problems above efficiently. In this paper, we propose a new cryptographic primitive, named Searchable Conditional Proxy Re-Encryption and shown in Fig. 6 and defined in Definition 3.3, which combines the techniques of HVE and PRE seamlessly.

Definition 3.3 (SC-PRE). *SC-PRE consists of the following eight polynomial algorithms* ***Setup***, ***KeyGen***, ***Enc***, ***QuTGen***, ***DelTGen***, ***ReEnc***, ***Search*** *and* ***Dec***.

- ***Setup*** *denoted as* $Setup(\lambda) \rightarrow params$. *It takes the security parameter* λ *as input and outputs system parameters* $params$:

$$params = (p, G, G_T, e, g, e(g,g), H_1).$$

  $params$ *is publicly known.* G *and* G_T *are two cyclic multiplication groups of prime order* p. g *is a generator of* G *and* $e(g,g)$ *is a generator of* G_T. $H_1 : \{0,1\}^* \rightarrow G$.
- ***KeyGen*** *denoted as* $KeyGen(|\vec{x}|) \rightarrow (pk, sk)$. *It takes as input the number of identity attribute-value pairs* $|\vec{x}|$. *The key generation algorithm (KenGen) outputs a public and private key pair* (pk, sk) *for the user. For example* (pk_o, sk_o) *for the key owner,* (pk_{du}, sk_{du}) *for the delegated user of the key owner.*
- ***Enc*** *denoted as* $Enc(pk_o, \vec{x}, \vec{k}) \rightarrow CT$. *It takes as input the public* pk_o *of key owner, a tuple* $t = \{\vec{x}, \vec{k}\}$, *vector* $\vec{k} \subseteq K$ *of key attributes and vector* $\vec{x} \subseteq W$ *of identity attributes. The encryption algorithm (Enc) based on HVE and PRE outputs the ciphertext CT of key tuple. In this process* $\vec{k}$ *is first transformed to the corresponding* ek *so as to hide the real* $\vec{k}$ *from the CloudKeyBank provider.*
- ***QuTGen*** *denoted as* $QuTGen(sk_o, \vec{w}) \rightarrow QuT_{o,\vec{w}}$. *It takes as input the private key* sk_o *of key owner and vector* $\vec{w} = (w[1], w[2], \ldots, w[|\vec{w}|])$ *of identity attributes,* $\vec{w} \subseteq W_*$. *The query token generation algorithm (QuTGen) outputs a query token* $QuT_{o,\vec{w}}$ *for subsequent privacy search query on EDB and query authorization on submitted queries.*
- ***DelTGen*** *denoted as* $DelTGen(sk_o, pk_{du}) \rightarrow dk_{o \rightarrow du}$. *It takes as input the private key* sk_o *of key owner and the public key* pk_{du} *of the delegated user. The delegation key generation algorithm (DelTGen) outputs the delegation token* $dk_{o \rightarrow du}$ *derived from the partial parameters of* sk_o *and* pk_{du}.
- ***Search*** *denoted as* $Search(CT, QuT_{o,\vec{w}}) \rightarrow ek$. *It takes as input the tuple ciphertext CT under public key* pk_o *and key owner's query token* $QuT_{o,\vec{w}}$, *and outputs the encrypted key* ek *on the condition that* $P_{\vec{w}}(\vec{x}) = 1$ *succeeds by evaluating* $QuT_{o,\vec{w}}$ *against the hidden vector* $\vec{x}$ *in encrypted tuple* CT, *otherwise outputs* $\perp$.
- ***ReEnc*** *denoted as* $ReEnc(ek, dk_{o \rightarrow du}) \rightarrow rek$, *conditional encryption algorithm. It takes as input the result of Search algorithm and the delegation token* $dk_{o \rightarrow du}$ *of the delegated user. The re-encryption algorithm (ReEnc) transforms encrypted key* ek *into re-encrypted key* rek *under* $dk_{o \rightarrow du}$; *otherwise it outputs* $\perp$ *if the result of Search algorithm is* $\perp$.
- ***Dec*** $(sk_{du}, rek) \rightarrow \vec{k}$. *Taking as input the private key* sk_{du} *and the re-encrypted key* rek, *the decryption algorithm (Dec) outputs plaintext* $\vec{k}$. *In terms of the customized access control policy of key owner,* sk_{du} *may be* sk_o *of the key owner or* sk_{du} *of the delegated user.*

By comparing the definition of SC-PRE with definitions of proxy re-encryption with keyword search (PRES) [33] and conditional proxy re-encryption (C-PRE) [34], knowing that SC-PRE can support the functionalities of PRES and C-PRE. It is not the focus of this paper, so we omit the details here.

4 SC-PRE Based CloudKeyBank Framework

In this section, based on known schemes HVE [10], [11] and PRE [28], we present a concrete SC-PRE scheme which provides the privacy and authorization enforced encryption support for our CloudKeyBank framework by distributing its eight polynomial algorithms into the following two protocols: DepositKey protocol described in Section 4.2 and WithdrawKey protocol describe in Section 4.3.

4.1 Preprocessing Work

The preprocessing work mainly includes *Setup* and *KeyGen* algorithms of SC-PRE and customized access control policy of key owner.

1) *Setup*(λ) algorithm generates global system parameters $params = (p, G, G_T, e, g, e(g,g), H_1)$, $H_1 : \{0,1\}^* \rightarrow G$, which are shared between users in CloudKeyBank framework.
2) *KeyGen*(n) algorithm generates public/private key (pk, sk) pairs for system users including key owner and delegated users. For $n = |\vec{x}_\ell|$, $\vec{x}_\ell \subseteq W_\ell$, $1 \leq \ell \leq d$, d the number of tuples in DB, $i \in \{1,2\}$, $j \in \{1,2,\ldots,n\}$, $k \in \{1,\ldots,n\}$. It picks the following random exponents: $a_1, a_2 \in Z_p^*, \vec{b_i} = (b_i[1], \ldots, b_i[j], \ldots, b_i[n])$, $b_i[j] \in Z_p^*$, $g_1, g_2 \in G$, $\vec{h_k} = (h_k[1], h_k[2], h_k[3])$. It sets $y_1 = g^{a_1}$, $y_2 = g^{a_2}$, $\vec{v_i} = (v_i[1], \ldots, v_i[j], \ldots, v_i[n])$, $v_i[j] = g^{b_i[j]} \in G$. In addition, it sets $\omega_1 = e(g_1, y_1) \cdot e(g_2, y_2) \in G_T$, $\omega_2 = e(g,g) \in G_T$. The generated (pk, sk) pairs for key owner o and delegated user du are denoted respectively as follows:

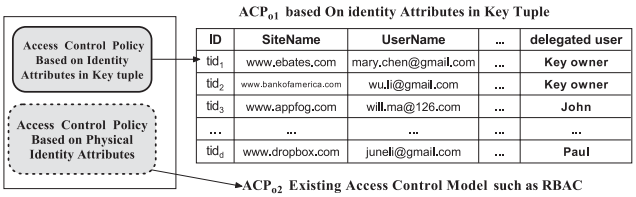


Fig. 7. The customized access control policy of key owner.

$$pk_o = (g, y_1, y_2, \vec{h_1}, \ldots, \vec{h_n}, \vec{v_1}, \vec{v_2}, \omega_1, \omega_2),$$

$$sk_o = (a_1, a_2, \vec{b_1}, \vec{b_2}, g_1, g_2),$$

$$pk_{du} = (g, y_1', y_2', \vec{h_1'}, \ldots, \vec{h_n'}, \vec{v_1'}, \vec{v_2'}, \omega_1', \omega_2'),$$

$$sk_{du} = (a_1', a_2', \vec{b_1'}, \vec{b_2'}, g_1', g_2').$$

3) *Customized Access Control Policy of Key Owner*. The key owner can customize his/her own access control policy based on two types of identity attributes shown in Fig. 7: identity attributes in key tuples, identity attributes in physical certificates such as unique identity number in identity card. For example the first type of access control policy ACP_{o1} is used to control query authorization based on identity attributes in key tuples, while the other ACP_{o2} (any existing access control model such as RBAC) is used to distribute query authorization to authenticated and authorized delegated users.

4.2 DepositKey Protocol

As shown in Fig. 8, DepositKey protocol is used to help the key owner outsource his/her keys to the CloudKeyBank provider with minimum information leakage such as the leakage of tuple identifier tid_ℓ. It is performed at the Trusted client and mainly includes the following three functional parts: DB encryption, Query token generation and Delegation token generation.

1) *DB encryption*. It is implemented through Encryption/ Decryption module shown in Fig. 8. To speed up query efficiency on Encrypted Key DB (EDB), we set each key tuple t_l in Key DB(DB) with the following form:

$$t_l = \{tid_l, \vec{x_l}, \vec{k_l}\} \in DB,$$

where $DB = (tid_l, W_l, K_l)_{l=1}^d$ is the Key DB of key owner, $\vec{x_l} \subseteq W_\ell$ and $\vec{k_l} \subseteq K_\ell$. The corresponding encrypted key tuple ct_l in EDB with the following form:

$$ct_l = \{tid_l, CT_l\} \in EDB.$$

tid_l is the randomly generated tuple identifier and the minimum information leakage in CloudKeyBank framework. Each ciphertext CT_l is generated by algorithm *Enc* of SC-PRE.

- $Enc(pk_o, \vec{x_l}, \vec{k_l}) \rightarrow CT_l$. *Enc* is implemented through the following two algorithms: level one encryption algorithm (*Lev1Enc*) and level two encryption algorithm (*Lev2Enc*).

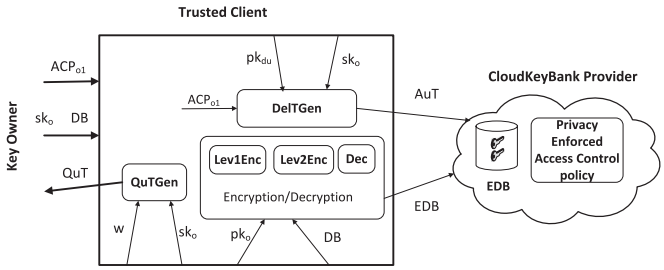


Fig. 8. DepositKey protocol.

  - $Lev1Enc(pk_o, \vec{x_l}, \vec{k_l}) \rightarrow CT_l$. It first picks three random exponents $f_1, f_2, s \in Z_p^*$, transforms $\vec{k_l}$ to $k = k_l[1]||, \ldots, ||k_l[n']$ and blinds k to $k' = k(\omega_2^s) \in G_T$. Then it encrypts the blinded key k' and vector $\vec{x_l}$ to ciphertext CT_l under the public key pk_o of key owner. The generated CT_l is denoted as follows:

$$CT_l = \{C_{1\vec{x_l}}, C_{2\vec{x_l}}, C_1, C_2, C_3, C_4, C_5, C_6\},$$

$$C_{1\vec{x_l}} = \Pi_{i \in [|\vec{x_l}|]} (h_i[1] h_i[2]^{x_l[i]})^{f_1} v_1[i]^{f_2}$$

$$= \Big((h_1[1] h_1[2]^{x_l[1]})^{f_1} v_1[1]^{f_2}, \ldots,$$

$$(h_n[1] h_n[2]^{x_l[n]})^{f_1} v_1[n]^{f_2} \Big),$$

$$C_{2\vec{x_l}} = \Pi_{i \in [|\vec{x_l}|]} h_i[3]^{f_1} v_2[i]^{f_2}$$

$$= (h_1[3]^{f_1} v_2[1]^{f_2}, \ldots, h_n[3]^{f_1} v_2[n]^{f_2}),$$

$$C_1 = y_1^{f_1}, C_2 = y_2^{f_2}, C_3 = g^{f_2},$$

$$C_4 = \omega_1^{f_1} \omega_2^{a_2 s}, C_5 = \omega_1^{f_1} k', C_6 = g^s.$$

  The *Lev1Enc* empowers the key owner with authorization capability that only the key owner with Query token QuT_o and private key sk_o can derive the original key $\vec{k_l}$ from CT_l.

  - $Lev2Enc(pk_o, \vec{x_l}, \vec{k_l}) \rightarrow CT_l$. CT_l generated by *Lev2Enc* is almost the same as that by *Lev1Enc* except for the following difference:

$$C_4 = \omega_1^{f_1} g^s, C_5 = \omega_1^{f_1} k',$$

$$C_6 = H_1(C_{1\vec{x_l}}, C_{2\vec{x_l}}, C_1, C_2, C_3, C_4, C_5)^s.$$

  The *Lev2Enc* empowers the key owner with authorization capability that the key owner with Query token QuT_o and private key sk_o or the delegated user du with Query token QuT_o and private key sk_{du} can derive the original key $\vec{k_l}$ from CT_l.

Based on ACP_{o1} shown in Fig. 7, the key owner can choose *Lev1Enc* or *Lev2Enc* to encrypt tuple t_l to CT_l with

ID	SiteName	UserName	...	delegated user
tid_1	www.ebates.com	mary.chen@gmail.com	...	Key owner
tid_2	www.bankofamerica.com	wu.li@gmail.com	...	Key owner
tid_3	www.appfog.com	will.ma@126.com	...	John
...	...	...	...	...
tid_d	www.dropbox.com	juneli@gmail.com	...	Paul

(a) ACP_{o1} of Key owner

ID	Delegation token
tid_1	$dk_{o\ \ o}$
tid_2	$dk_{o\ \ o}$
tid_3	$dk_{o\ \ John}$
...	...
tid_d	$dk_{o\ \ Paul}$

(b) Authorization Table(AuT)

Fig. 9. ACP_o and authorization table.

enforced key authorization. For instance, only key owner *mary.chen@gmail.com* can access his own outsourced key by using *Lev1Enc*, while delegated user *Paul* can access the outsourced key of key owner *juneli@gmail.com* by using *Lev2Enc*.

2) *Query token generation*. It is implemented through algorithm *QuTGen* of SC-PRE shown in Fig. 8. The generated Query token $QuT_{o,\vec{w}_l}$ can hide the private key sk_o of key owner and the identity attributes in $\vec{w}_l$:

- $QuTGen(sk_o, \vec{w}_l) \rightarrow QuT_{o,\vec{w}_l}$. It sets $\vec{w}_l = (w_l[1], w_l[2], \ldots, w_l[n]) \subset W_*$ and takes $Ind(\vec{w}_l)$ as the set of all indexes i such that $w_l[i] \neq *$. To generate $QuT_{o,\vec{w}_l}$ for $\vec{w}_l$, it chooses random parameters $A_1, A_2 \in Z_p^*$, $r_i, d_i \in Z_p^*$ and $\eta_i, \tau_i \in Z_p^*$ such that $r_i a_1 + d_i a_2 = A_1$ and $\eta_i a_1 + \tau_i a_2 = A_2$ for all $i \in Ind(\vec{w}_l)$. The generated $QuT_{o,\vec{w}_l}$ is denoted as follows:

$$QuT_{o,\vec{w}_l} = (Q_{1\vec{w}_l}, Q_{2\vec{w}_l}, Q_{3\vec{w}_l}, Q_1, Q_2),$$

$$Q_{1\vec{w}_l} = g_1 \prod_{i \in Ind(\vec{w}_l)} (h_i[1] h_i[2]^{w_l[i]})^{r_i} h_i[3]^{\eta_i},$$

$$Q_{2\vec{w}_l} = g_2 \prod_{i \in Ind(\vec{w}_l)} (h_i[1] h_i[2]^{w_l[i]})^{d_i} h_i[3]^{\tau_i},$$

$$Q_{3\vec{w}_l} = g^{-\sum_{i \in Ind(\vec{w}_l)} (b_1[i] A_1 + b_2[i] A_2)},$$

$$Q_1 = g^{A_1}, Q_2 = g^{A_2}.$$

In terms of ACP_{o1} shown in Fig. 7, the key owner can enforce his/her query authorization by distributing appropriate Query token $QuT_{o,\vec{w}_l}$ to the delegated user. For instance, the delegated user *Paul* can submit a search query on EDB in a privacy preserving way if he has authorized $QuT_{o,\vec{w}_l}$ of key owner *juneli@gmail.com*, i.e. the CloudKeyBank provider can evaluate $QuT_{o,\vec{w}_l}$ against CT_l in EDB correctly, but do know nothing about the hidden sk_o and hidden identity attributes in CT_l.

3) *Delegation token generation*. It is implemented through algorithm *DelKGen* of SC-PRE shown in Fig. 8. The generated Delegation token $dk_{o \rightarrow du_j}$ can hide the private key sk_o of key owner and the identity attributes based conditions in key tuples.

- $DelKGen(sk_o, pk_{du_j}) \rightarrow dk_{o \rightarrow du_j}$. Given key owner's private key sk_o and delegated user's public key pk_{du_j}, it computes Delegation token $dk_{o \rightarrow du_j} \in G$ by using partial parameters from private key sk_o and public key pk_{du_j}:

$$dk_{o \rightarrow du_j} = (y_2')^{a_1},$$

where a_1 is from sk_o and $y_2' = g^{a_2'}$ is from pk_{du_j}. The Delegation token for the key owner is $dk_{o \rightarrow o} = (y_2)^{a_1}$.

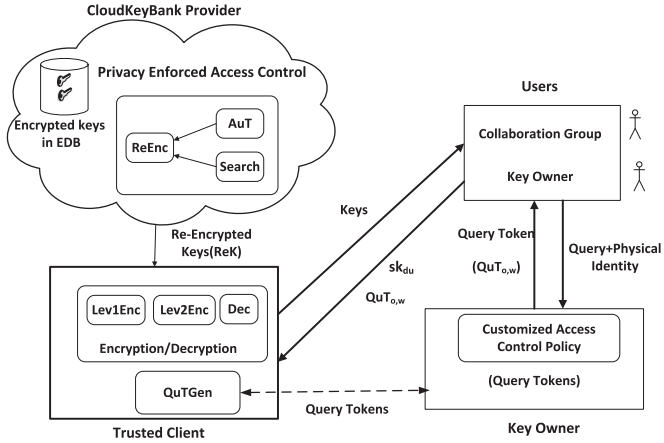


Fig. 10. WithdrawKey protocol.

In terms of ACP_{o1} shown in Fig. 7, the key owner can enforce his/her key authorization by delegating corresponding Delegation token $dk_{o \rightarrow du_j}$ to CloudKeyBank provider. For instance, Fig. 9b is the generated anonymous authorization table AuT corresponding to ACP_{o1} shown in Fig. 9a. $AuT = \{(tid_l)_{l=1}^{d}, (dk_{o \rightarrow du_j})_{j=1}^{d'}\}$, $d' \leq d$ means that each $dk_{o \rightarrow du_j}$ may be related to one or more tid_ls. After DepositKey protocol, shown in Fig. 8, the Trusted client sends $EDB = \{tid_l, CT_l\}_{l=1}^{d}$ and $AuT = \{(tid_l)_{l=1}^{d}, (dk_{o,du_j})_{j=1}^{d'}\}$ to the CloudKeyBank provider, $QuT_{o,\vec{w}} = (QuT_{o,\vec{w}_l})_{l=1}^{d}$ to the key owner.

4.3 WithdrawKey Protocol

As shown in Fig. 10, WithdrawKey protocol is used to help the delegated users obtain the shared keys of key owner with minimum privacy leakage such as the leakage of Query token $QuT_{o,\vec{w}_i}$. It is performed at both the CloudKeyBank provider and the Trusted Client, and mainly includes the following three functional parts: Search query on EDB, Conditional delegation and Key derivation.

1) *Search query on EDB*. It is implemented through algorithm *Search* of SC-PRE shown in Fig. 10, in which the CloudKeyBank provider can perform the privacy enforced search query by evaluating the submitted Query token QuT_{o,w_i} against ciphertext $CT_l \in (ct_l = \{tid_l, CT_l\})$. For each $ct_l \in EDB$, $1 \leq l \leq d$, the *Search* algorithm is performed as follows:

- $Search(CT_l, QuT_{o,\vec{w}_i}) \rightarrow ek_i$ *or* $\perp$, $ek_i = (cm_{i1}, cm_{i2}) \in G_T^2$ or $ek_i = (cm_{i1}, cm_{i2}) \in G \times G_T$ in our solution. Taking as input ciphertext CT_l and $QuT_{o,\vec{w}_j}$, search algorithm computes:

$$\frac{e(Q_1, C_{1\vec{x}_l}) \cdot e(Q_2, C_{2\vec{x}_l}) \cdot e(Q_{3\vec{w}_i}, C_3)}{e(Q_{1\vec{w}_i}, C_1) \cdot e(Q_{2\vec{w}_i}, C_2)} \cdot C_4 \rightarrow cm_{i1}, \tag{2}$$

$$\frac{e(Q_1, C_{1\vec{x}_l}) \cdot e(Q_2, C_{2\vec{x}_l}) \cdot e(Q_{3\vec{w}_i}, C_3)}{e(Q_{1\vec{w}_i}, C_1) \cdot e(Q_{2\vec{w}_i}, C_2)} \cdot C_5 \rightarrow cm_{i2}. \tag{3}$$

The output of *Search* algorithm depends on whether the following verification succeeds or not.

PWManager	LoginDomain	Name	Password	Collaboration	SearchPrivacy
LastPass		√	√	√	
My1login		√	√	√	
PasswordBox			√	√	
RoboForm					
NeedMyPassword					
CloudKeyBank	√	√	√	√	√

Fig. 11. Privacy comparison with web based password managers.

- If cm_{i1} is equal to $e(y_2, C_6)$, it outputs $ek_i||0 = (cm_{i1}, cm_{i2})||0$ to the *ReEnc* algorithm.
- if $e(cm_{i1}, H_1(C_{1\vec{x}_l}, C_{2\vec{x}_l}, C_1, C_2, C_3, C_4, C_5))$ is equal to $e(g, C_6)$, it outputs $ek_i||1 = (cm_{i1}, cm_{i2})||1$ to the *ReEnc* algorithm.
- If both of the two above are false, then it outputs $\perp$ to the *ReEnc* algorithm.

Search algorithm is the first level of access control enforcement by the CloudKeyBank Provider because of the following truth: to protect search privacy on identity attributes against the honest-but-curious CloudKeyBank provider, *Search* algorithm is implemented based on the fact described in HVE [10], [11] where two vectors $\vec{x}_l \subset W_l$ and $\vec{w}_i \subset W_*$ over identity attributes are chosen in SC-PRE. Therefore if evaluating $QuT_{o,\vec{w}_i}$ against ciphertext $CT_l \in EDB$ makes the $P_{\vec{w}_i}(\vec{x}_l) = 1$, *Search* algorithm outputs $ek_i||flag$, otherwise outputs $\perp$. Here $flag \in \{0, 1\}$ is the bit flag to denote ek_i for the key owner or for the delegated user.

2) *Conditional delegation*. It is implemented through algorithm *ReEnc* of SC-PRE shown in Fig. 10. *ReEnc* is a conditional re-encryption algorithm because its input depends on the output of *Search* algorithm. *ReEnc* algorithm is performed as follows:
   - *ReEnc*$(ek_i||flag, dk_{o\to du}) \to rek_i$ or $\perp$. Taking the output of *Search* algorithm and Delegation token $dk_{o\to du}$ from AuT as input, The *ReEnc* algorithm performs the re-encryption as follows:
     - If $flag = 0$, it outputs $rek_i = ek_i = (cm_{i1}, cm_{i2}) = (\alpha, \beta)$ to the key owner.
     - if $flag = 1$, it first transforms ciphertext $ek_i = (cm_{i1}, cm_{i2})$ to another ciphertext rek_i under Delegation token $dk_{o\to du} = g^{a_1 a_2'}$ from AuT. Then it computes $e(g^s, g^{a_1 a_2'}) = e(g, g)^{a_2' a_1 s}$ and outputs $rek_i = (e(g, g)^{a_2' a_1 s}, ke(g, g)^{a_1 s}) = (\omega_2^{a_2' a_1 s}, k\omega_2^{a_1 s}) = (\omega_2^{a_2' s'}, k\omega_2^{s'}) = (\alpha, \beta)$.
     - If both of the two above are false, it outputs $\perp$.

ReEnc algorithm is the second level of access control enforcement by the CloudKeyBank Provider because of the following truth: the CloudKeyBank provider can re-encrypt ek_i to rek_i under the Delegation token $dk_{o\to du}$ in AuT table. After performing the two algorithms above, the CloudKeyBank provider sends the re-encrypted key set $ReK = (rek_i)_{i=1}^{d''}$, $rek_i \in G_T^2$ corresponding to $QuT_{o,\vec{w}} = (QuT_{o,\vec{w}_i})_{i=1}^{d''}$ to the Trusted client shown in Fig. 10.

3) *Key derivation*. It is implemented through algorithm *Dec* of SC-PRE in Encryption/Decryption module shown in Fig. 10. The key owner with sk_o and the delegated user with sk_{du} can derive $\vec{k}_i$ from rek_i as follows:
   - *Dec*$(sk_{du}, rek_i) \to \vec{k}_i$. To decrypt ciphertext $rek_i = (cm_{i1}, cm_{i2})$ from *Lev1Enc* algorithm, the key owner uses his partial private key $a_2 \in sk_o$ to compute $k_i[1] = \beta/\alpha^{1/a_2}$. To decrypt ciphertext $rek_i = (\alpha, \beta)$ from *Lev2Enc* algorithm, the delegated user uses his private key $a_2' \in sk_{du_i}$ to compute $k_i[1] = \beta/e(\alpha, g)^{a_2'}$.

After WithdrawKey protocol, the Trusted client sends all the authorized keys $(\vec{k}_i)_{i=1}^{d'}$ to du through a secure channel shown in Fig. 10.

Security / Scheme	Key Confidentiality and Privacy	Search Privacy (on Identity Attributes)	Owner Controllable Authorization	
			Key Authorization (on Key Attributes)	Query Authorization
Hacigumus[12,13]	√			
Cash[14,15]		√		
Shang[17]	√			
Nabeel[18]	√			
Tian[21,24]	√		√	
Li[26]		√		√
Our Solution	√	√	√	√

Fig. 12. Comparison with proposed approaches in traditional data outsourcing scenario.

5 Security Analysis

In this section we first describe the privacy comparison between CloudKeyBank and proposed approaches. Then we analyze the security properties of CloudKeyKey framework from two aspects: Key privacy and search privacy, Owner controllable authorization enforcement.

5.1 Privacy Comparison with Password Managers

By comparing with the web based password managers such as LastPass, My1login shown in Fig. 11, we can see that CloudKeyBank framework can provide more privacy guarantees.

As shown in Fig. 11 they do not provide search privacy, and most of them classify the name and password field into the same security level. In practice the name is less sensitive than the password, therefore in our solution we classify name into the Search attribute group. My 1login allows fine-grained controls on read/write permissions of the shared passwords, however, they do not prevent a collaborator from updating the account password. In PasswordBox there exists a severe privacy breach because it only encrypts the password while exposes user's full name, login domain, creation time and so on. What is more serious, NeedMyPassword and RoboForm even do not provide basic password encryption protection, they store all private information of users as plaintext. In conclusion the private identity information of users managed by existing password managers is easily compromised by malicious attackers or exploited by the honest-but-curious password service providers.

5.2 Privacy Comparison with Proposed Approaches

A comparison of our approach with related others, as shown in Fig. 12, demonstrates that our solution can solve

the three identified security requirements, whereas the others proposed in the traditional data outsourcing scenario cannot.

5.3 Key Privacy and Search Privacy

Key privacy. The key attributes in Key attribute group $\vec{k}_l$ are encrypted two times. One is the hiding encryption under partial parameters (y_1, y_2), the other is the hidden vector encryption on both the blinded key k' and identity attributes in Search attribute group $\vec{x}_l$ under pk_o. Therefore even if the attacker evaluates the intercepted $QuT_{o,\vec{w}_{l'}}$ against CT_l in EDB correctly, he still does not derive $\vec{k}_l$ from rek_l because it is hard to compromise sk_{du} of the delegated user.

Search privacy. Query token $QuT_{o,\vec{w}_l}$ is generated by using hidden vector encryption on vector $\vec{w}_l \subset W_*$ under private key sk_o. $QuT_{o,\vec{w}_l}$ hides sk_o and $\vec{w}_l$ by using random chosen parameters $A_1, A_2, r_i, d_i, \eta_i, \tau_i$. Therefore even if an attacker intercepts $QuT_{o,\vec{w}_l}$ from communication channel between the Trusted client and the CloudKeyBank provider, he still does not derive any private information from $QuT_{o,\vec{w}_l}$.

5.4 Key Owner Controllable Authorization Enforcement

The owner controllable authorization is enforced as follows:

- Corresponding to Query token, there exists the first level access control enforcement by the CloudKeyBank provider. Only the delegated user du who satisfies the access control policy ACP_{o2} of key owner can submit key owner's $QuT_{o,\vec{w}_l}$ to the CloudKeyBank provider. Only after $QuT_{o,\vec{w}_{l'}}$ is evaluated from CT_l correctly, does the second level access control enforcement be performed under the appropriate Delegation token.
- Corresponding to Delegation token, there exists the second level access control enforcement by the CloudKeyBank provider. Only the Delegation token $dk_{o\to du}$ for the authorized delegated user is delegated and stored in anonymous authorization table AuT, the encrypted key ek_i of key owner from search algorithm can be transformed to the re-encrypted key rek_i correctly.
- Only after both the two access control enforcements above are performed correctly, does the authorized delegated user du derive original $\vec{k}_i$ from rek_i under his own private key sk_{du}.

From above knowing that an attacker is difficult to derive the shared keys of key owner unless he attacks the following three at the same time: Query token $QuT_{o,\vec{w}_l}$ authorized by the key owner, Delegation token in anonymous authorization table AuT generated by the key owner and stored at the CloudKeyBank provider and private key sk_{du} of the authorized delegated user du.

6 Performance Evaluation

In this section, we describe the performance evaluation from three aspects: communication overhead, computation efficiency and query efficiency. We give experiment analysis based on the following development environment: Mac air with operating system OS X Version 10.9.2, hardware with 2.9 GHz Intel Core i7 and 8 GB 1,600 MHz DDR3, software with Python 2.7 under the support of different cryptographic and multiple precision arithmetic libraries such as Pyparsing 2.0.1, Pbc 0.5.14, Openssl 1.0 and Gmp 5.1.3.

6.1 Communication Overhead

Assume n and n' are the number of identity attributes in vector $\vec{x}_i$ and key attributes in vector $\vec{k}_i$ respectively. N is the number of all identity attributes $W = \bigcup_{i=1}^{d} W_i$, $N = |W|$. Here $|G| = 224$ bit and $|G_T| = 521$ bit denote element size on multiplication group G and G_T respectively. $p = 191$ is the prime number. $DB = (W_i, K_i)_{i=1}^{d}$, d is the number of key tuples in key DB, $d = |DB|$. $AuT = \{(tid_i)_{i=1}^{d}, (dk_{o\to du_j})_{j=1}^{d'}\}$, d' is the number of delegated users of the key owner, $d' = |AuT|$. d'' is the number of authorized Query tokens to the delegated user du. In our experiment we only consider the following three communication overheads directly related to users (including key owner).

1) *Between the key owner and trusted client in DepositKey protocol (O-T)*. The public key size is $(5n+3)|G| + 2|G_T|$ and the private key size is $2|G| + (2n+2)p$. The key DB size is $8d(15n + 4n')$ where each search attribute size is 15 Bytes and each key attribute size is 4 Bytes. Query token size outputted from the $QuTGen$ algorithm is $5|G|$. Therefore the total overload size is as follows:

$$\Big((5n+5)|G| + 2|G_T| + d(5|G| + 8(15n+4n')) + (2n+2)p\Big),$$

where d is a variable depending on key DB of the key owner, n and n' are constant for specified applications. For instance, LastPass has basic fields user name, login domain, creation time and password and so on.

2) *Between the delegated user and trusted client in WithdrawKey protocol (Du-T)*. Each Query token size is $5|G|$. Each re-encryption ciphertext size from the ReEnc algorithm is $2|G_T|$. The private key size is $2|G| + (2n+2)p$. Therefore the total overload size is:

$$d''(5|G| + 2|G_T|) + 2|G| + (2n+2)p,$$

where d'' is a variable depending the number of authorized Query tokens to the delegated user du.

3) *Between the key owner and delegated user (O-Du)*. The authorized Query tokens are the main overload between the key owner and the delegated user which has the size $5d''|G|$.

In conclusion, each overload size above is a linear function of d or d'' for the fixed number $(n + n')$ of attributes in each key tuple shown in Fig. 13. This is true for most of the practical applications such as LastPass and PasswordBox. Here we assume the shared keys take up 20 percent of the whole d database tuples, i.e. $d'' = 20\% \times d$. As seen in Fig. 13, the communication overload between the key owner and trusted client in the DepositKey protocol is higher than the other two, but it does not influence the user experience

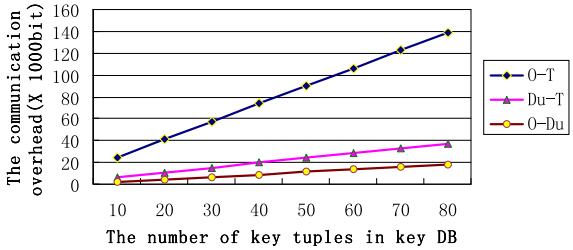


Fig. 13. Communication overheads related to users.

in the WithdrawKey protocol because most of the operations between them can be performed offline. The communication overhead between the key owner and delegated user is almost constant due to the constant Query token with size of $5|G|$. For example, if $|G| = 224\,\text{bit}$, so each Query token size is a constant $5|G| = 1120\,\text{bit}$.

6.2 Computation Efficiency

In terms of the secure pairing computation proposed in [39], the security of $224\,\text{bit}$ pairing computation is equal to that of $2{,}048\,\text{bit}$ RSA, $256\,\text{bit}$ security is equal to that of $3{,}072\,\text{bit}$ RSA. As shown in Fig. 14, the cost time of $521\,\text{bit}$ group encryption in our solution is much more efficient than that of $3{,}072\,\text{bit}$ RSA given different number of identity attributes, but the security strength is equal to that of $15{,}360\,\text{bit}$ RSA. From Fig. 14 we can see that in our solution the time cost grows slowly with the increasing number of identity attributes in Search attribute group, while the time cost increases faster when the identity attributes are over the number of 70. Therefore our solution can be efficiently extended to support large scale documents with many query keywords.

6.3 Query Efficiency

The query response time mainly includes two parts: time of Query tokens from the key owner to the delegated user, time of accessing and deriving keys from the CloudKeyBank provider. Fig. 15a shows the response time for outsourced passwords in CloudKeyBank and Fig. 15b shows the response time for outsourced data encryption keys in CloudKeyBank. From them we know that the response time is almost linear with the number of outsourced key tuples in key DB. The response time is increasing with the growth of authorized Query tokens from $20\% * d$, $50\% * d$ to d. However, it is under the limit of user acceptance. For example querying on an outsourced EDB with 200 key tuples and $20\% * d$ authorized Query tokens needs 1.89 seconds. The query response is more efficient because the CloudKeyBank provider does not need to return the whole encrypted key

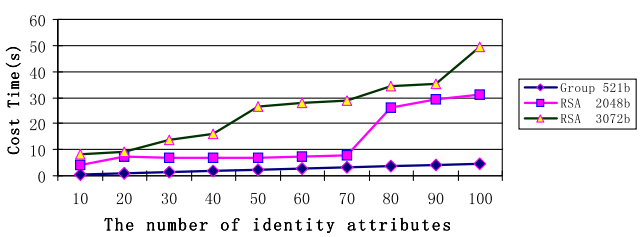


Fig. 14. Time cost for secure RSA and group computation.

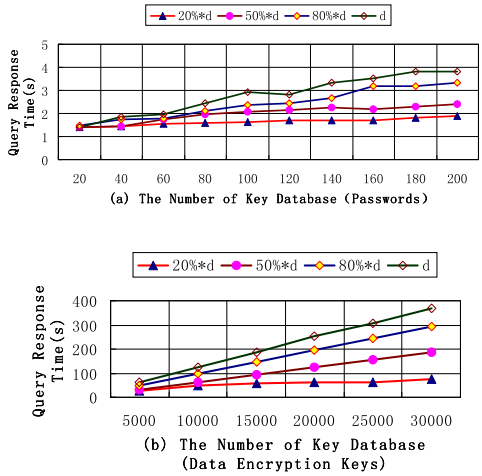


Fig. 15. Query response time.

tuple like that in most of the proposed approaches [13], [22], [20] Our solution only needs to return the re-encrypted key ciphertexts ReK_l which are decrypted at the Trusted client under private key sk_{du} of du.

7 Related Work

In recent years academia and industry have paid more attention to privacy problems in outsourcing scenario. In the following sections, we just describe the related work based on which our solution is developed.

7.1 Encryption Based Privacy and Authorization in Database as a Service (DaaS)

Hacigumus et al. [12] first introduced the concept of database as a service for outsourced database management. To guarantee privacy and access authorization of outsourced data, data owners employ different cryptographic techniques to encrypt data so as to implement different goals of privacy protection. For example bucketization encryption [13], order preserving encryption [22] and B+ based encryption [20] were proposed to support efficient queries on encrypted data with less privacy leakage. The approaches in [12], [13], [21] mainly guarantee the confidentiality and privacy of data by encrypting data tuples in an all or nothing way. Policy based encryption [17], [21], [23], [24], [25] were proposed to encrypt data in such a way that only users who were permitted to access data in terms of the access policies could be able to do so. Therefore efficient key management determined by the access control policy becomes the primary factor of policy based encryption. The approaches in [17], [25] focus on achieving the identity and authorization privacy of users by encrypting identity attributes and related identity based conditions in the access control policy. Bertino et al. [19] proposed an unified framework for an outsourced medical database to enforce the privacy and copyright protection by combining techniques of binning and digital watermarking. However, to the best of our knowledge, there is no related research on how to provide the privacy and owner authorization enforced key management for large number of web based applications and outsourced data storage. We tackle the problem in this paper.

7.2 Conjunctive Keywords and Search on Encrypted Data

Searchable encryption consists of searchable symmetric encryption (SSE) introduced by Song et al. [2] and public encryption with keyword search (PKES) introduced by Boneh et al. [3]. Based on SSE, the approaches proposed in [14], [15] mainly achieves the conjunctive search privacy on data tuples of scalable encrypted database. Based on PKES, the concept of predicate encryption [3], [4], allowing searches on encrypted data without a private key that corresponds to a public key, was created. In predicate encryption scheme, a service provider is given a token, instead of the full private key, for evaluating one or more predicates on the encrypted data [7]. Hidden vector encryption [6] is one kind of predicate encryption where two vectors over attributes are associated with a ciphertext and a token respectively. HVE supports conjunctive search queries over encrypted data. HVE schemes proposed in [6], [5], [7] are constructed on bilinear groups with a composite order. Katz [5] extended the predicate encryption to support disjunction and inner product. But due to the high cost of computation on exponentiation and bilinear pairings, Lovino and Persiano [8] proposed a scheme based on prime group which requires a token size of $O(n)$ and pairing computation of $O(n)$, when n is the number of attribute conjunctions. It still results in the huge query cost because each of the ciphertexts could be a valid ciphertext that matches the token. Park proposed a low cost HVE [10], [11] scheme which is also based on pairing group but only requires a $O(1)$ token size and $O(1)$ pairing computation. Therefore following predicate privacy in [9] and scheme proposed in Park [10], [11], based on search attribute group $\vec{x_i}$ of key tuple t_i we introduce two vectors to guarantee the search privacy on conjunctive identity attributes.

7.3 Proxy Re-Encryption with Keyword Search

In 1998, Blaze et al. [27] introduced a concept of proxy re-encryption (PRE) where a proxy can re-encrypt a ciphertext from the sender to a re-ciphertext decrypted under the private key of the receiver. There are two types of PRE, one is based on the re-encryption direction including bidirectional and unidirectional, the other is based on the number of hops including single hop and multi-hop. In 2005 Ateniese et al. [28] presented different unidirectional schemes using bilinear maps and showed how to prevent proxies from colluding with delegatees to derive the delegator's secret key. From then on, Many different type of PRE schemes are proposed [29], [30], [31]. Proxy re-encryption with keyword search introduced by Shao and Cao [33] enables a proxy to transfer the re-ciphertext to a delegatee in a way that the keyword search can be performed on the ciphertext by using a keyword token. PRES scheme proposed in [36] can guarantee the privacy of keyword. Conditional proxy re-encryption introduced by Weng et al. [34], [35] provided a solution to enforce the conditional delegation capabilities of the delegator where only ciphertexts satisfying a certain keyword condition set by the delegator can be transferred by the proxy. To further enforce keyword privacy and fine-grained conditional delegation capability of the delegator, Fang et al. introduced anonymous PRE [32] and hierarchical proxy re-encryption (H-PRE) [37] respectively. H-PRE extends the single keyword based delegation to conjunctive keywords based delegation. However, to date, the major work proposed has focused on single keyword search and no proposed C-PRE schemes can support hidden multi-keyword based search on ciphertext.

8 Conclusion and Future Work

To solve the identified critical security requirements for keys outsourcing, we present CloudKeyBank, the first unified privacy and owner authorization enforced key management framework. To implement CloudKeyBank, we propose a new cryptographic primitive SC-PRE and the corresponding concrete SC-PRE scheme. The security comparison and analysis prove that our solution is sufficient to support the identified three security requirements which are not be solved in traditional outsourcing scenario. From the performance analysis, we can see that our solution is not so efficient because it requires several seconds to answer a query on a database only 200 passwords. The main reason for inefficiency is that SC-PRE belongs to one kind of public encryption which is inefficient in common by comparing to the symmetric encryption. That is what we want to solve in our future work where we will introduce searchable symmetric encryption, bloom filter based index in one server, and access policy enforcement in another server to support scalable operations on encrypted key database.

Acknowledgments

First, the authors would like to thank the anonymous reviewers for their helpful comments and suggestions for this paper. Second, the authors are very appreciative of the contribution of Jun Shao, who first proposed proxy re-encryption with keyword search (PRES) in 2009, which gives us much more professional technology support on how to combine HVE and PRE seamlessly from the cryptography theory and security. This work was supported by NSFC grants (No. 61202020), Natural Science Foundation of Shanghai City grant (No. 12ZR1411900) and 973 project (No. 2010CB328106).

References

[1] J. Bonneau, C. Herley, P. C. van Oorschot, and F. Stajano, "The quest to replace passwords: A framework for comparative evaluation of web authentication schemes," in *Proc 33th IEEE Symp. Security Privacy*, 2012, pp. 553–567.

[2] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proc. IEEE Symp. Security Privacy*, Oakland, California, USA, May 2000, pp. 44–55.

[3] D. Boneh, G. D. Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *Proc. Adv. Cryptography*, May 2004, pp. 506–522.

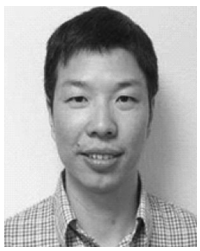
[4] X. Boyen and B. Waters, "Anonymous hierarchical Identity-based encryption (without random oracles)," in *Proc. 26th Annu. Int. Cryptol. Conf.*, 2006, pp. 290–307.

[5] J. Katz, A. Sahai, and B. Waters, "Predicate encryption supporting disjunctions, polynomial equations, and inner products," in *Proc. 27th Annu. Int. Conf. Adv. Cryptol. Theory Appl. Cryptograph. Techn.*, vol. 4965, pp. 146–162, 2008.

[6] D. Boneh and B. Waters, "Conjunctive, subset, and range queries on encryted data," in *Proc. Conf. Theory Cryptography*, 2007, vol. 4392, pp. 535–554.

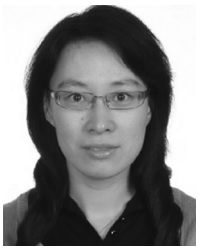
[7] E. Shi and B. Waters, "Delegating capabilites in predicate encryption systems," in *Proc. Int. Colloq. Automata, Languages Program.*, 2008, vol. 5126, pp. 560–578.
[8] V. Iovino and G. Persiano, "Hidden-vector encryption with groups of prime order," in *Proc. Int. Conf. Pairing-Based Cryptography*, 2008, vol. 5209, pp. 75–88.
[9] E. Shen, E. Shi, and B. Waters, "Predicate privacy in encryption systems," in *Proc. Theory Cryptography Conf.*, 2009, vol. 5444, pp. 457–473.
[10] J. H. Park, "Efficient hidden vector encryption for conjunctive queries on encrypted data," *IEEE Trans. Knowl. Data Eng.*, vol. 23, no. 10, pp. 1483–1497, Oct. 2011.
[11] J. Hwan Park, K. Lee, W. Susilo, and D. Hoon Lee, "Fully secure hidden vector encryption under standard assumptions," *Inf. Sci.*, vol. 232, pp. 188–207, 2013.
[12] H. Hacigumus, B. Iyer, C. Li, and S. Mehrotra, "Executing SQL over encrypted data in the database-service-provider model," in *Proc. 18th Int. Conf. Data Eng.*, 2002, pp. 216–227.
[13] H. Hacigumus, B. Iyer, C. Li, and S. Mehrotra, "Executing SQL over encrypted data in the database-service-provider model," in *Proc. 18th Int. Conf. Data Eng.*, Jun. 2002, pp. 216–227.
[14] D. Cash, Stanislaw, C. Jutla, H. Krawczyk, M.-C. Rosu, and M. Steiner, "Highly-scalable searchable symmetric encryption with support for boolean queries," in *Proc 33th Int. Conf. Crypography Conf.*, 2013, pp. 353–373.
[15] D. Cash, J. Jaeger, S. Jarecki, C. S. Jutla, H. Krawczyk, M. Catalin Rosu, and M. Steiner, "Dynamic searchable encryption in very-large databases: Data structures and implementation," presented at the 21th Network and Distributed System Security, San Diego, CA, USA, Feb. 2014.
[16] V. Pappas, F. Krell, B. Vo, V. Kolesnikov, T. Malkin, S. Geol Choi, W. George, A. D. Keromytis, and S. M. Bellovin, "Blind Seer: A scalable private DBMS," in *Proc. 35th IEEE Symp. Security Privacy*, San Jose, CA, USA, May 2014, pp. 359–374.
[17] N. Shang, F. Paci, M. Nabeel, and E. Bertino, "A privacy-preserving approach to policy-based content dissemination," in *Proc 26th Int. Conf. Data Eng.*, 2010, pp. 944–955.
[18] M. Nabeel and E. Bertino, "Privacy preserving delegated access control in public clouds," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 9, pp. 2268–2280, Sep. 2014.
[19] E. Bertino, B. C. Ooi, Y. Yang, and R. H. Deng, "Privacy and ownership preserving of outsourced medical data," in *Proc 21th Int. Conf. Data Eng.*, 2005, pp. 521–532.
[20] F. Li, M. Hadjieleftheriou, G. Kollios, and L. Reyzin, "Dynamic authenticated index structures for outsourced databases," in *Proc ACM SIGMOD Int. Conf. Manag. Data*, 2006, pp. 121–132.
[21] X. Tian, X. Wang, and A. Zhou, "DSP re-encryption a flexible mechanism for access control enforcement management in DaaS, in *Proc. IEEE Int. Conf. Cloud Comput.*, 2009, pp. 25–32.
[22] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu, "Order preserving encryption for numeric data," in *Proc ACM SIGMOD Int. Conf. Manag. Data*, 2004, pp. 563–574.
[23] S. D. C. di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati, "Over-encryption: Management of access control evolution on outsourced data," in *Proc 29th Int. Conf. Very Large Data Bases*, 2007, pp. 123–134.
[24] X. X. Tian, X. L. Wang, and A. Y. Zhou, "DSP Re-encryption based access control enforcement management mechanism in DaaS," *Int. J. Netw. Security*, vol. 15, no. 1, pp. 28–41, 2013.
[25] X. X. Tian, L. Huang, Y. Wang, C. F. Sha, and X. L. Wang, "DualAcE: Fine-grained dual access control enforcement with Multi-privacy guarantee in DaaS," *Secure Commun. Netw.*, vol. 8, no. 8, pp. 1494–1508, 2015
[26] M. Li, S. C. Yu, N. Cao, and W. Liu, "Authorized private keyword search over encrypted data in cloud computing," in *Proc. 31st Int. Conf. Distrib. Comput. Syst.*, 2011, pp. 383–394.
[27] M. Blaze, G. Bleumer, and M. Strauss, "Divertible protocols and atomic proxy cryptography," in *Proc. Int. Conf. Theory Appl. Cryptograph. Techn.*, 1998, pp. 127–144.
[28] G. Ateniese, K. Fu, M. Green, and S. Hohenberger, "Improved proxy re-encryption schemes with applications to secure distributed storage," in *Proc. 12th Annu. Netw. Distrib. Syst. Security Symp.*, 2005, pp. 29–44.
[29] M. Green and G. Ateniese, "Identity-based proxy re-encryption," in *Proc. 5th Int. Conf. Appl. Cryptography Netw. Security*, 2007, pp. 288–306.
[30] R. Canetti and S. Hohenberger, "Chosen-ciphertext secure proxy Re-encryption," in *Proc. 14th ACM Conf. Comput. Commun. Security*, 2007, pp. 185–194.
[31] B. Libert and D. Vergnaud, "Unidirectional chosen-ciphertext secure proxy reencryption," in *Proc. 11th Int. Workshop Practice Theory Public-Key Cryptography*, 2008, pp. 360–379.
[32] L. Fang, W. Susilo, and J. Wang, "Anonymous conditional proxy re-encryption without random oracle," in *Proc. 3rd Int. Conf. Provable Security*, 2009, pp. 47–60.
[33] J. Shao and Z. Cao, "CCA-secure proxy re-encryption without pairings," in *Proc. 12th Int. Conf. Practice Theory Public Key Cryptography*, 2009, pp. 357–376.
[34] J. Weng, R. H. Deng, C. Chu, X. Ding, and J. Lai, "Conditional proxy re-encryption secure against chosen-ciphertext attack," in *Proc. 4th ACM Int. Symp. Inf., Comput. Commun. Security*, 2009, pp. 322–332.
[35] J. Weng, M. Chen, Y. Yang, R. Deng, K. Chen, and F. Bao, "CCA-secure unidirectional proxy re-encryption in the adaptive corruption model without random oracles," *Science China Inf. Sci.*, Express vol. 53, no. 3, pp. 593–606, 2010.
[36] H. S. Rhee, W. Susilo, and H.-J. Kim, "Secure searchable public key encryption scheme against keyword guessing attacks," *IEICE Electronics Express*, vol. 6, no. 5, pp. 237–243, 2009.
[37] L. M. Fang, W. Susilo, C. P. Ge, and J. D. Wang, "Hierarchical conditional proxy re-encryption," *Comput. Standards Interfaces*, vol. 34, pp. 380–389, 2012.
[38] D. Boneh, X. Boyen, and H. Shacham, "Short group signature," in *Proc. 24th Annu. Int. Cryptol. Conf.*, 2004, pp. 41–55.
[39] B. Lynn, "On the implementation of pairing-based cryptography," A dissertation of computer science at Stanford University, 2007.

Xiuxia Tian received the MS degree in applied cryptography-based information security from Shanghai Jiaotong University in 2005, and the PhD degree in database security and privacy preserving in cloud computing from Fudan University in 2011. She is currently a professor in the School of Computer Science and Technology, Shanghai University of Electric Power. She is a visiting scholar of two years at UC Berkeley working with groups of SCRUB and SecML. She has published more than 40 papers and some papers are published in international conferences and journals such as DASFAA, ICWS, CLOUD, and SCN. Her main research interests include database security, privacy preserving (large data and cloud computing), applied cryptography, and secure machine learning.



Ling Huang received the PhD degree in computer science from the University of California at Berkeley in 2007. He is a research scientist at Intel Labs, and is currently a member of the Intel Science and Technology Center on Secure Computing at UC Berkeley. His research interests include machine learning, distributed systems, and security analytics. Between 2007 and 2011, he was a research scientist at Intel Labs Berkeley.

Tony Wu is currently an third year undergraduate working toward the BS degree in electrical engineering and computer science at UC Berkeley. He is also an undergraduate researcher at the UC Berkeley SCRUB lab. His main research interests are machine learning, security, and mobile technologies.



Xiaoling Wang received the bachelor's, master's, and doctoral degrees from Southeastern University in 1997, 2000, and 2003, respectively. She is currently a professor, vice dean in Software Engineering Institute, East China Normal University. She was an assistant professor and an associate professor at Fudan University from 2003 to 2008, and joined East China Normal University in 2008. She achieved the Programs of New-Century Talent of Ministry of Education of China and Innovation of Science and Technology of Department of Education of Shanghai. She is the associate editor of *Frontier of Computer Science*. She acted as a program committee cochair of BDMS 2013, a member of program committee of VLDB2014ER 2013, SCC 2013, SCC2014, APWeb 2013, and is a member of China Computer Federation Technical Committee on Databases. She has published more than 60 papers and some papers were published in international conferences and journals such as SIGMOD, WWW, SIGIR, DASFAA, WAIM, ICWS, and *JWSR*. Her research interests mainly include web data management, data service technology, and applications. She is a member of the IEEE.

Aoying Zhou received the bachelor's and master's degrees in computer science from Sichuan University, in 1985 and 1988, respectively, and the PhD degree from Fudan University in 1993. He is currently a professor of computer science at East China Normal University (ECNU), where he is heading the Institute for Data Science and Engineering. He received the National Science Fund for Distinguished Young Scholars supported by NSFC and the professorship appointment under Changjiang Scholars Program of Ministry of Education. He is currently acting as a vice-director of ACM SIGMOD China and the Database Technology Committee of the China Computer Federation. He is serving as a member of the editorial boards of the *VLDB Journal*, *WWW Journal*, etc. His research interests include web data management, data management for data-intensive computing, memory cluster computing, benchmarking for big data, and performance. He is a member of the IEEE.