Efficient Proxy Re-encryption with Private Searching in the Untrusted Cloud

Xi Chen, Yong Li

Key Laboratory of Communication & Information Systems (Beijing Jiaotong University), Beijing Municipal Commission of Education, Beijing 100044, China

Abstract

As promising as cloud computing is, this paradigm brings forth new security and privacy challenges when operating in the untrusted cloud scenarios. In this paper, we propose a new cryptographic primitive Proxy Re-encryption with Private Searching (PRPS for short). The PRPS scheme enables the data users and owners efficiently query and access files stored in untrusted cloud, while keeping query privacy and data privacy from the cloud providers. The concrete construction is based on proxy re-encryption, public key encryption with keyword search and the dual receiver cryptosystem. The scheme is semantically secure under the BDH assumption.

Index Terms: public key encryption with keyword search; proxy re-encryption; untrusted cloud; private searching

1. Introduction

Cloud computing is an important trend which is beginning to fulfill the early promise of the Internet and creating unanticipated change in computing paradigm. However, a significant barrier to the adoption of cloud computing is that data owners fear of confidential data leakage and lose of privacy in the cloud [1]. These concerns originate from the fact that cloud providers are usually operated by commercial providers which are very likely to be outside of the trusted domain of the data owners or users. Data confidentiality against cloud providers is hence frequently desired when data owners outsource data for storage in the cloud [2].

Our work is motivated by the following scenario. Data owners, cloud storage providers and data users are separated geographically. Data owner stores his files in an encrypted form in the untrusted cloud, and retrieves them wherever and whenever he wants. The user sends a query for files containing certain keywords to the cloud provider. The desired requirements are: 1) The user can decrypt the files uploaded by the data owner with his private key; 2) The cloud provider can search whether the encrypted files contain some keywords; 3) The

Corresponding author:
E-mail address: 09120113@bjtu.edu.cn, li.yong9@gmail.com
cloud provider oughts to keep blind to the files content and the query keywords of the user; 4) The user could finish query and decryption with a thin client which demands computing overhead as small as possible.

1.1. Related work

**Proxy Re-Encryption (PRE).** PRE is a cryptographic primitive, where a (potentially untrusted) proxy is given a re-encryption key $rk_{1 \to 2}$ that allows it to translate a message $m$ encrypted under public key $pk_1$ into a ciphertexts under a public key $pk_2$, without being able to see anything about the encrypted messages. In [3], Ateniese et al. proposed a single-use, unidirectional, but not transparent Proxy Re-Encryption schemes based on bilinear maps.

**Public key encryption with keyword search (PEKS).** In PEKS scheme, Alice creates a trapdoor with her private key and a keyword, and sends it to S. S uses a test algorithm with inputing encrypted keyword, trapdoor and user’s public key. If matches, it outputs 1 and 0 otherwise. PEKS supports that a user could search for some files containing certain keywords in untrusted storage servers, and at the same time, the servers keep blind to the privacy of file and the keyword. In [4], Boneh et al. proposed a public key encryption with keyword search scheme.

**Dual receiver cryptosystem.** Diamet et al. [5] first introduced the notion of an efficient dual receiver cryptosystem, which enables a ciphertext to be decrypted by two independent receivers. The main disadvantage of the dual receiver cryptosystem is that the server needs to send an auxiliary private key to a client for decrypting a partial ciphertext, which is insecure in the real environment [6]. Liu et al. [6] improved the PEKS by inspiring the idea of dual receiver cryptosystem, and proposed an efficient privacy preserving keyword search scheme. However, this scheme is one specific case applicable in the setting that the data owner and data user is the same one. Shao et al. [7] introduced the concept of proxy re-encryption with keyword search (PRES), in particular the concept of bidirectional PRES, against the chosen ciphertext attack. Note that the third party is trusted, and this scheme improved the security level with the sacrifice of efficiency.

1.2. Our contributions

We proposed a new cryptographic primitive, Proxy Re-encryption with Private Searching (PRPS), and the new PRPS construction combines technologies from PRE, PEKS and dual receiver cryptosystem. The PRPS scheme is able to protect the data privacy and the users’ queries privacy simultaneously during the search process. And it is provably secure under the BDH assumption in random oracle model. In addition, the PRPS scheme enables the decrease of computing overhead for the user and reduces the modification of encrypted storage when different users accessing the cloud provider.

2. Preliminaries

Let $G_1$ and $G_2$ be two cyclic groups of some large prime order $q$. We view $G_1$ as an additive group and $G_2$ as a multiplicative group.

**Definition 2.1 (Bilinear Maps):** A bilinear map $e: G_1 \times G_1 \rightarrow G_2$ is a map with the following properties: (1) Bilinearity: for any integers $x, y \in [1, q]$, we have $e(g^x, g^y) = e(g,g)^{xy}$. (2) Computability: given $g, h \in G_1$, there is a polynomial time algorithms to compute $e(g,h) \in G_2$. (3) Non-degeneracy: if $g$ is a generator of $G_1$, then $e(g,g)$ is a generator of $G_2$.

**Definition 2.2 (BDH Problem):** Given a random element $g \in G_1$, as well as $g^x, g^y$ and $g^z$, for some $x, y, z \in Z_q^*$, compute $e(g,g)^{xy} \in G_2$. 
3. Proxy Re-encryption with Private Searching

Definition 3.1 Proxy Re-encryption with Private Searching (PRPS) scheme consists of seven randomized polynomial time algorithms as follows:

- **Key Generation (KG):** takes a sufficiently large security parameter \( K_i \) as input, and produces a public/private key pair \( (A_{pub}, A_{priv}) \) for a data owner \( A \). We write \( KG(K_i) \square (A_{pub}, A_{priv}) \). Let \( K_2 \) be a sufficiently large security parameter, we write \( KG(K_2) \square (S_{pub}, S_{priv}) \) for the cloud provider \( S \), where \( S_{pub}, S_{priv} \) are public/private key respectively. Let \( K_3 \) be a sufficiently large security parameter, we write \( KG(K_3) \square (U_{pub}, U_{priv}) \) for the data user \( U \), where \( U_{pub}, U_{priv} \) are public/private key respectively.

- **Encryption (E):** this algorithm is performed by data owner \( A \) to encrypt the keyword \( W_i(z', z^*) \) and message \( m \). Correspondingly, two parts, KWEnc and EMBEnc constitute Encryption.
  1. \( KWEnc \): is a public key encryption algorithm that takes a public key \( A_{pub} \) and a key word \( W_i(z', z^*) \) as inputs, and produces ciphertext \( C_W \). We write \( KWEnc(A_{pub}, W_i) \square C_W \).
  2. \( EMBEnc \): is a public key encryption algorithm that takes public keys \( S_{pub}, A_{pub} \) and message \( m \square M \) as inputs, and produces \( m' \) ciphertext \( C_m \). We write \( EMBEnc(S_{pub}, A_{pub}, m) \square C_m \).

- **Re-Encryption Key Generation (RG):** A data owner takes private key \( A_{priv} \) and user’s public key \( U_{pub} \) as inputs, and produces the re-encryption key \( rk_{A,U} \). We write \( RG(A_{priv}, U_{pub}) \square rk_{A,U} \).

- **TCompute:** User takes private key \( U_{priv} \) and a keyword \( W_j(z') \) as inputs, and produces \( W_j \) ciphertext \( C_W \). We write \( TCompute(U_{priv}, W_j) \square C_W \).

- **Re-Encryption (R):** The cloud provider takes re-encryption key \( rk_{A,U} \), ciphertext \( C_m \) and some intermediate result \( U \) as the inputs, and produces ciphertext \( C_m \) re-encrypted ciphertext \( C_U \). We write \( ReEncryption(U, rk_{A,U}, C_m) \square C_U \).

- **Test:** The cloud provider takes re-encryption key \( rk_{A,U} \), an encrypted keyword \( C_m \) and a trapdoor \( T_m \) as inputs, and produces “1” if \( W_j \square W_i \) or “0” otherwise. This algorithm is to check whether the ciphertext \( C_m \) matches the trapdoor \( T_m \).

- **Decryption (D):** The user takes private key \( U_{priv} \) and re-encrypted ciphertext \( C_U \) as inputs, and outputs the plaintext \( m \).

Definition 3.2 (Semantic Security of KWEnc): Given a public key encryption algorithm KWEnc which encrypts keywords using \( A_{pub} \), let \( \square \) be a polynomial time IND-CPA adversary that can adaptively ask for the trapdoor \( T_m \) for any keyword \( W \) of its choice. \( \square \) first chooses two keywords \( W_0 \) and \( W_1 \), which are not to be asked for trapdoors previously, and sends them to KWEnc. Then KWEnc picks a random element \( h \{0,1\} \) and gives \( \square \) the ciphertext \( C_w \). Finally, \( \square \) outputs a guess \( \hat{h} \{0,1\} \) for \( h \). We define the advantage of \( \square \) in breaking KWEnc as \( Adv_{kw}(k) = \Pr[h \hat{h}] \cdot \frac{1}{2} \). KWEnc is semantically secure if for any polynomial time adversary \( \square \), \( Adv_{kw}(k) \) is negligible.

Definition 3.3 (Semantic Security of EMBEnc): Given a public key encryption algorithm EMBEnc which encrypts the message using \( A_{pub} \) and \( S_{pub} \). Let \( \square \) be a polynomial time IND-CPA adversary that can adaptively ask for the ciphertext for any message \( m \) of its choice. We use subscript \( T \) to denote the target user, \( x \) to denote the adversarial users, and \( h \) to denote the honest users (other than \( T \)). The input marked with a \(*\) is optional. \( \square \) first chooses two messages \( m_h \) and \( m_x \), which are not to be asked for the ciphertext previously, and
The file deposited in the cloud storage is protected by public key encryption algorithms. Finally, an adversary outputs a guess \( b_2 \in \{0,1\} \) and gives \( b_2 \) the cloud storage’s public key. Keywords may be words in headline or stored date, and \( l \) may be the length of the keyword. That is, for all PPT algorithms \( A_h \),

\[
\Pr[(pk_r, sk_r) \rightarrow KG(t), ((pk_r, sk_r) \rightarrow KG(t))],
\]

\[
\{rk_{s, t} \rightarrow RG(pk_r, sk_r, pk_r, sk_r^t)\},
\]

\[
\{rk_{s, t} \rightarrow RG(pk_r, sk_r, pk_r, sk_r^t)\},
\]

\[
\{rk_{s, t} \rightarrow RG(pk_r, sk_r, pk_r, sk_r^t)\},
\]

\[
(m_{i, m_{i-1}}, \ldots, A_i (pk_r, ((pk_r, sk_r)), \{pk_r\}, \{rk_{s, t}\}, \{rk_{s, t}\}, \{rk_{s, t}\}),
\]

\[
b_2 \in \{0,1\}, b_2 \rightarrow A_i (REMEnc (pk_r, m_{b_i})): \quad b_2 \in b_2 < 1/2 \in 1/poly(k)
\]

We define the advantage of \( b_2 \) in breaking \( REMEnc \) as \( Adv_h (k) \rightarrow \Pr[b_2 \rightarrow b_2 > 1/2] \). We say that \( REMEnc \) is semantically secure if for any polynomial time adversary \( b_2 \), \( Adv_h (k) \) is negligible.

Definition 3.4 (Semantic Security of PRPS): Given an PRPS scheme consisting of \( KWEnc \) and \( REMEnc \), it takes a security parameter \( K \) as input and runs the key generation algorithm \( KG \) to generate the public/private key pairs \( (A_{pub}, A_{priv}) \), \( (S_{pub}, S_{priv}) \) and \( (U_{pub}, U_{priv}) \). Given an adversary \( U \) consisting of two polynomial time algorithms \( U_1 \) and \( U_2 \), \( U \) initiates attacks on \( KWEnc \) and \( U \) initiates attacks on \( REMEnc \). We say that the PRPS Scheme is semantically secure if for any adversary \( U \), \( Adv_h (k) \rightarrow Adv_h (k) \rightarrow Adv_h (k) \) is negligible.

4. Construction for PRPS

We assume that the scheme is composed of the following entities, the data owner, data users, and cloud providers. To access data files shared by the data owner, data users download data files of their interest from cloud providers and then decrypt. The users are assumed to have the only access privilege of data file reading. The cloud providers are assumed to have abundant storage capacity and computation power.

In our scheme, cloud providers are viewed as “honest but curious”, which means they follow the proposed protocol in general, but try to find out as much secret information as possible. Cloud providers might collude with malicious users for the purpose of harvesting file contents when it is highly beneficial. Communication channel between the data owner/users and cloud providers are assumed to be secured. Users may work independently or cooperatively.

The main design goal is to help the data users achieve efficient private querying and downloading the encrypted files stored in cloud providers. The data owner won’t need to re-encrypt the files in cloud provider for different users. We also want to prevent cloud providers from being able to learn both the data file contents and user queries information.

Suppose data owner \( A \) is about to store an encrypted file with keywords \( W_1, \ldots, W_l \) on a cloud storage \( S \), where \( I \in \mathbb{Z}^* \). Keywords may be words in headline or stored date, and are relatively small. \( A \) encrypts the file message using his public key \( A_{pub} \), the cloud storage’s public key \( S_{pub} \). And then \( A \) encrypts keywords \( W_1, \ldots, W_l \) using his public key \( A_{pub} \). The file deposited in the cloud storage \( S \) by the data owner \( A \) is as follows:

\[
MSG_{U35} \[ REMEnc (A_{pub}, S_{pub}, m), KWEnc (A_{pub}, W_1), \ldots, KWEnc (A_{pub}, W_l) \]
\]

where \( REMEnc \), \( KWEnc \) are public key encryption algorithms. Finally, \( A \) appends to the encrypted file message with all the encrypted keywords and sends \( MSG_{U35} \) to \( S \).
Given a sufficiently large security parameter $K \in \mathbb{Z}^*$, two groups $G_1$ and $G_2$ of prime order $q$, and a bilinear map $e: G_1 \times G_2 \rightarrow \mathbb{G}$, where $g$ is a generator of $G_1$. Then it chooses two hash functions $H_1, H_2: \mathbb{Z}_q \rightarrow G_1^*$, hash function $H_1^*: G_2 \rightarrow \mathbb{Z}_q$ for some $n$, where $H_1, H_2, H_3$, and $H_4$ are random oracles. Finally, it picks three random elements $a, b, c \in \mathbb{Z}_q^*$ and computes $g^a, g^b$ and $g^c$. The plaintext space includes $M \in \{0,1\}^*$ and $W \in \{0,1\}$. The ciphertext space includes $C_M \in G_1^* \cdot \mathbb{Z}^*$ and $C_W \in G_2$.

- **Key Generation (KG):** The data owner $A$'s public key is $A_{pub} \equiv g^a$ with the corresponding private key $A_{priv} \equiv a$; the user $U$'s public/private key is $U_{pub} \equiv g^b$, $U_{priv} \equiv b$ respectively. The cloud provider's public key is $S_{pub} \equiv g^c$ with the corresponding private key $S_{priv} \equiv c$.

- **Encryption (E):** This encryption algorithm consists of $KWEnc$ and $EMEnc$. The data owner first picks a random element $r \in \mathbb{Z}_q^*$.

  1. $KWEnc(E_i) \colon$ To encrypt $m$'s keywords $W_{i_1}, \ldots, W_{i_t} (k \in \mathbb{Z}^*)$ under a data owner’s public key $g^a$ and a random element $r$, it computes $H_i(e(g^a, H_i(W_i)))$, where $W_i \in \{W_{i_1}, \ldots, W_{i_n}\}$, sets the ciphertext $C_W = H_i(e(g^a, H_i(W_i)))$.

  2. $EMEnc(E_i) \colon$ To encrypt the file message $m$ under data owner’s public key $g^a$, cloud provider’s public key $g^b$ and random element $r$, it picks a random element $r \in \mathbb{Z}_q^*$, and computes $u \equiv r' \cdot e(H_i(h^r, g^{r'}), u \equiv m \cdot e(H_i(\cdot), g^{r'})$, and sets the ciphertext $C_u \equiv (u, u_1, u_2)$.  

- **Re-Encryption KeyGeneration (RG):** Data owner $A$ delegates to user $U$ by publishing the re-encryption key $rk_{A \rightarrow U} \equiv g^{abr}$, computed with $U$'s public key $g^b$.

- **Tcompute:** To retrieve the file containing keyword $W_j(j \in \mathbb{Z}^*)$, user computes the trapdoor $T_{W_j} \equiv H_j(W_j)^{y_j}$ using his/her private key $U_{priv} \equiv b$, then sends the trapdoor to the cloud provider.

- **Re-Encryption (R):** to change the ciphertext $C_u \equiv (u, u_1, u_2)$ for $A$ into a ciphertext $C_v \equiv (u_3, u_4)$ for $U$ under the re-encryption key $rk_{A \rightarrow U} \equiv g^{abr}$, it computes $u_3 \equiv e(H_i(\cdot), rk_{A \rightarrow U}) \equiv e(H_i(\cdot), g^{abr})$. The cloud provider sends $C_v \equiv (u_3, u_4)$ to the user.

Note. Since $u_3 \equiv e(H_i(g^a, h^r), u_4 \equiv H_i(e(g^a, h^r))$, the cloud provider can compute the intermediate value $c$ with its private key $c$.

- **Test:** To determine whether a given file contains keyword $W_j$, the cloud provider tests whether $C_W \equiv H_i(e(rk_{A \rightarrow U}, T_{W_j})).$ If so, $Test(rk_{A \rightarrow U}, C_W, T_{W_j})$ outputs 1, and 0 otherwise.

Note. If $W \equiv W_j$, since $C_W \equiv H_j(e(g^a, H_j(W_j)))$, then $C_W \equiv H_j(e(g^a, H_j(W_j))) \equiv H_j(e(g^{abr}, H_j(W_j)^{y_j})) \equiv H_j(e(rk_{A \rightarrow U}, T_{W_j}))$.

- **Decryption (D):** Given the ciphertext $C_v \equiv (u_3, u_4)$, it computes $m \equiv u_5 \cdot (u_4)^{y_{pub}} \equiv u_5 / (u_4)^{y_{priv}}$ to recover the message $m$.

  

Note that: $\frac{u_5}{(u_4)^{y_{pub}}} \equiv \frac{m \cdot e(H_i(\cdot), g^{r'})}{e(H_i(\cdot), g^{r'})} \equiv m$.
5. Security Analysis

Lemma 5.1 (Privacy for Keyword) Let $H_1$ be a random oracle from $\{0,1\}^*$ to $G_1^*$ and $H_2$ be a random oracle from $G_2$ to $\{0,1\}^{\log{q}}$. Suppose $\square_1$ be an IND-CPA adversary that has the advantage $\mathcal{A}_1$ in breaking KWEnc. Suppose $\square_1$ makes at most $q_h > 0$ hash queries to $H_2$ and at most $q_r > 0$ trapdoor queries. Then there is an algorithm $B_1$ that solves the BDH problem with the advantage at least $\mathcal{A}_1 \times 2^{\mathcal{A}_1} / \left\{ e^{\mathcal{A}_1} q_h, \mathcal{A}_1 q_r \right\}$, and a running time $O(\text{time}(\square_1))$.

Lemma 5.2 (Privacy for Message) Let $H_3$ be a random oracle from $\{0,1\}^*$ to $G_1^*$. Let $\square_2$ be an IND-CPA adversary that has the advantage $\mathcal{A}_2$ against EMBEnc. Suppose $\square_2$ makes $q_h > 0$ hash function queries to $H_1$ and $q_r > 0$ queries to Request. Then there is an algorithm $B_2$ that solves the BDH problem with the advantage at least $\mathcal{A}_2 \times 2^{\mathcal{A}_2} / q_h q_r$ and a running time $O(\text{time}(\square_2))$.

Theorem 5.1 (Security for PRPS) Suppose the hash functions $H_1, H_2, H_3$ and $H_4$ are random oracles. Let $\square$ be an IND-CPA adversary consisting of two polynomial time algorithms $\square_1$ and $\square_2$. Let $\square_1$ be an IND-CPA adversary that has the advantage $\mathcal{A}_1$ in breaking KWEnc. Suppose $\square_1$ makes $q_r > 0$ trapdoor queries and $q_h > 0$ hash queries to $H_2$. Let $\square_2$ be an IND-CPA adversary that has the advantage $\mathcal{A}_2$ against EMBEnc. Suppose $\square_2$ makes $q_h > 0$ hash function queries to $H_1$ and $q_r > 0$ queries to Request. Let $\square$ be an IND-CPA adversary that has the advantage $\mathcal{A}_1 \mathcal{A}_2$ against the PRPS scheme. Then there is an algorithm $\square$ that solves the BDH problem with the advantage at least:

$$\text{Adv}_{\square} = 2^{\mathcal{A}_1} / \left\{ e^{\mathcal{A}_1} q_h, \mathcal{A}_1 q_r \right\} \times 2^{\mathcal{A}_2} / q_h q_r$$

Here $e \approx 2.71$ is the base of the natural logarithm. The running time of $\square$ is $O(\text{time}(\square))$.

Due to page limitation, the details of formal security proof and some remarks are provided in the full version.

6. Conclusions

In this paper, we propose an efficient proxy re-encryption with private searching (PRPS) scheme in the untrusted cloud. We exploit proxy re-encryption and uniquely combining it with techniques of public key encryption with keyword search and dual receiver cryptosystem. PRPS allows users and data owners to query and access files stored in untrusted cloud provider, while maintaining query privacy and data privacy. It allows user to decrypt the files efficiently. The PRPS scheme is proven semantically secure in the random oracle model.

Acknowledgements

This work is partially supported by National High Technology Research and Development Program of China (863 Program) under Grant No. 2009AA01Z423 and the Fundamental Research Funds for the Central Universities under Grant No. 2009JBM004.

References


