I.J. Computer Network and Information Security, **2011, 2, 50-56**
Published Online March 2011 in MECS (http://www.mecs-press.org/)

Efficient Proxy Re-encryption with Private Keyword Searching in Untrusted Storage

Xi Chen
Key Laboratory of Communication & Information Systems (Beijing Jiaotong University),
Beijing Municipal Commission of Education, Beijing 100044, China
Email: 09120113@bjtu.edu.cn

Yong Li
Key Laboratory of Communication & Information Systems (Beijing Jiaotong University),
Beijing Municipal Commission of Education, Beijing 100044, China
Email: li.yong9@gmail.com

***Abstract*—Cloud computing is an important trend that in many ways is beginning to fulfill the early promise of the Internet and creating unanticipated change in computing paradigm. As promising as cloud computing is, this paradigm brings forth new security and privacy challenges when operating in the untrusted cloud scenarios. Motivated by the challenging problem "Private Searching over Encrypted Data", 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. Extensive analysis shows that our scheme is efficient and semantically secure under the BDH assumption.**

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

I. 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 users. Data confidentialty against cloud providers is hence frequently desired when users 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. A data owner stores his files in an encrypted form in the untrusted cloud, and retrieves them wherever and whenever he wants. What's more, he wants to share his files with other data users. 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 cloud provider ought 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. We call such kind of problem as "Private Searching on Encrypted Data" (PSED for short).

A. 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 cipher texts 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 inputting 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. Diament et al [5] first introduced the notion of an efficient dual receiver cryptosystem, which enables a cipher text 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 cipher text, 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 exists an inherent problem. It 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 cipher text attack. Their scheme is based on the techniques for PRE in [8] and the IBE schemes in [9]. Note that the third party is trusted and this scheme improved the security level with the sacrifice of efficiency.

Note that there are further related work [10][11] and the latest work in Structured Encryption [12], which also considered the problem of private querying on encrypted data, i.e. enabling user efficiently query and retrieve the encrypted files containing specific keywords.

B. *Our contributions*

Main contributions of this paper can be summarized as follows.

1) 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.

2) The PRPS scheme enables the decrease of computing overhead for the user.

3) It reduces the modification of encrypted sharing file storage when different users accessing the cloud provider.

The rest of this paper is organized as follows. Section II discusses some preliminaries. Section III provides the Proxy Re-encryption with Private Searching model and its security definition. Section IV introduces the construction for PRPS. In Section V, we analyze the PRPS scheme in terms of its security and efficiency. We conclude this paper in Section VI.

II. 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): We call e a bilinear map if $e: G_1 \times G_1 \to G_2$ is a map with the following properties:

1) Computable: given $g, h \in G_1$, there is a polynomial time algorithms to compute $e(g,h) \in G_2$.

2) Bilinear: for any integers $x, y \in [1,q]$, we have $e(g^x, g^y) = e(g,g)^{xy}$.

3) Non-degenerate: if g is a generator of G_1, then $e(g,g)$ is a generator of G_2.

Definition 2.2 (BDH Parameter Generator): We say that a randomized algorithm IG is a BDH parameter generator if IG takes a sufficiently large security parameter $K > 0$, runs in polynomial time in K, and outputs the description of two groups G_1 and G_2 of the same prime order q and the description of a bilinear map $e: G_1 \times G_1 \to G_2$.

Definition 2.3 (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 $e(g,g)^{xyz} \in G_2$.

Definition 2.4 (BDH Assumption): If IG is a BDH parameter generator, the advantage $Adv_{IG}(\mathrm{B})$ that an algorithm B has in solving the BDH problem is defined to be the probability that B outputs $e(g,g)^{xyz}$ on inputs $G_1, G_2,\ e, g, g^x, g^y, g^z$,where (G_1, G_2, e) is the output of IG for a sufficiently large security parameter K, g is a random generator of G_1, and x, y, z are random elements of Z_q^*. The BDH assumption is that $Adv_{IG}(\mathrm{B})$ is negligible for any efficient B.

III. 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_1 as input, and produces a key pair (A_{pub}, A_{priv}) for a data owner A, where A_{pub}, A_{priv} are public key and private key respectively. We write $KG(K_1) = (A_{pub}, A_{priv})$. Let K_2 be a sufficiently large security parameter, we write $KG(K_2) = (S_{pub}, S_{priv})$ 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 $KG(K_3) = (U_{pub}, U_{priv})$ 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 $W_i (i \in Z^+)$ and message m. Correspondingly, two parts, KWEnc and EMBEnc constitutes Encryption.

  1) KWEnc: is a public key encryption algorithm that takes a public key A_{pub} and a key word $W_i (i \in Z^+)$ as inputs, and produces W_i 's cipher text $C_{W_i} \in C_w$.We write $KWEnc(A_{pub}, W_i) = \mathrm{C}_{W_i}$.

2) EMBEnc: is a public key encryption algorithm that takes public keys S_{pub} , A_{pub} and message $m \in M$ as inputs, and produces $m's$ cipher text C_m .We write $EMBEnc(S_{pub}, A_{pub}, m) = C_m$.

- Re-Encryption Key Generation (RG): A data owner takes a public key U_{pub} and private key A_{priv} as inputs, and produces the re-encryption key $rk_{A \to U}$. We write $RG(A_{priv}, U_{pub}) = rk_{A \to U}$.
- TCompute: User takes private key U_{priv} and a keyword W_j $(j \in Z^+)$ as inputs, and produces W_j 's trapdoor T_{W_j} . We write
$$TCompute(U_{priv}, W_j) = T_{W_j}$$.
- Re-Encryption(R): The cloud provider takes re-encryption key $rk_{A \to U}$, cipher text C_m and some intermediate result θ as the inputs, and produces cipher text C_m 's re-encrypted cipher text C_U . We write $\text{Re-}Encryption(\theta, rk_{A \to U}, C_m) = C_U$.
- Test: The cloud provider takes re-encryption key $rk_{A \to U}$, an encrypted keyword C_{W_i} and a trapdoor T_{W_j} as inputs, and produces "1" if $W_i = W_j$ or "0" otherwise. This algorithm is to check whether the cipher text C_{W_i} matches the trapdoor T_{W_j} .
- Decryption (D): The user takes private key U_{priv} and re-encrypted cipher text C_U as inputs, and outputs the plaintext m .

Note. RG algorithm implies that the PRPS scheme is non-interactive, which means re-encryption keys can be generated by a data owner via the user's public key. No trusted third party or interaction is required.

We define security for the PRPS scheme in the sense of semantic security. Semantic security captures the intuition that given a cipher text, the adversary learns nothing about the corresponding plaintext, thus we also say that a semantically secure scheme is IND-CPA secure [9]. We first define semantic security for KWEnc and EMBEnc, and then give the definition of semantically secure PRPS scheme.

Definition 3.2 (Semantic Security of KWEnc): Given a public key encryption algorithm KWEnc which encrypts keywords using A_{pub} , let A_1 be a polynomial time IND-CPA adversary that can adaptively ask for the trapdoor T_{W_i} for any keyword $W_i \in W$ of its choice. A_1 first chooses two keywords W_0 and W_1, which are not to be asked for trapdoors previously, and sends them to KWEnc. And then KWEnc picks a random element $b_1 \in \{0,1\}$ and gives A_1 the cipher text $C_{W_{b_1}} = KWEnc(A_{pub}, W_{b_1})$. Finally, A_1 outputs a guess $b_1^{'} \in \{0,1\}$ for b_1 . We define the advantage of A_1 in breaking KWEnc as
$$Adv_{\mathrm{A}_1}(k) = \left| \Pr[b_1 = b_1^{'}] - \frac{1}{2} \right|.$$
KWEnc is semantically secure if for any polynomial time adversary A_1 , $Adv_{\mathrm{A}_1}(k)$ 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 A_2 be a polynomial time IND-CPA adversary that can adaptively ask for the cipher text for any message $m_i \in M$ 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. A_2 first chooses two messages m_0 and m_1 , which are not to be asked for the cipher text previously, and sends them to EMBEnc. And then EMBEnc picks a random $b_2^{'} \in \{0,1\}$ and gives A_2 the cipher text
$$C_{m_{b_2}} = EMBEnc(A_{pub}, S_{pub}, m_{b_2}).$$
Finally, A_2 outputs a guess $b_2^{'} \in \{0,1\}$ for b_2 . That is, for all PPT algorithms A_k ,
$$\begin{aligned} &\Pr[(pk_T, sk_T) \leftarrow KG(1^k), \{(pk_x, sk_x) \leftarrow KG(1^k)\}, \\ &\{rk_{x \to T} \leftarrow RG(pk_x, sk_x, pk_T, sk_T^*)\}, \\ &\{(pk_h, sk_h) \leftarrow KG(1^k)\}, \\ &\{rk_{T \to h} \leftarrow RG(pk_T, sk_T, pk_h, sk_h^*)\}, \\ &\{rk_{h \to T} \leftarrow RG(pk_h, sk_h, pk_T, sk_T^*)\}, \\ &(m_0, m_1, \alpha) \leftarrow A_k(pk_T, \{(pk_x, sk_x)\}, \{pk_h\}, \{rk_{x \to T}\}, \{rk_{T \to h}\}, \{rk_{T \to D}\}), \\ &b_2 \leftarrow \{0,1\}, b_2^{'} \leftarrow A_k(\alpha, EMBEnc(pk_T, m_{b_2})): \\ &b_2 = b_2^{'}] < 1/2 + 1/poly(k) \end{aligned}$$

We define the advantage of A_2 in breaking EMBEnc as
$$Adv_{\mathrm{A}_2}(k) = \left| \Pr[b_2 = b_2^{'}] - \frac{1}{2} \right|.$$
We say that EMBEnc is semantically secure if for any polynomial time adversary A_2 , $Adv_{\mathrm{A}_2}(k)$ is negligible.

Definition 3.4 (Semantic Security of PRPS): Given an PRPS scheme consisting of KWEnc and EMBEnc, it takes a security parameter K as input and runs the key generation algorithm Keygen to generate the public/ private key pairs (A_{pub}, A_{priv}) , (S_{pub}, S_{priv}) and (U_{pub}, U_{priv}) . Given an adversary A consisting of two polynomial time algorithms A_1 and A_2 , A_1 initiates attacks on KWEnc and A_2 initiates attacks on EMBEnc. We say that the PRPS Scheme is semantically secure if for any adversary A , $Adv_{\mathrm{A}}(k) = Adv_{\mathrm{A}_1}(k) + Adv_{\mathrm{A}_2}(k)$ is negligible.

IV. CONSTRUCTION FOR PRPS

We assume that the scheme is composed of the following parties, 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 this work, 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. More specifically, we assume cloud providers are more interested in file contents and user access privilege information than other secret information. 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. In addition, each party is preloaded with a public/private key pair and the public key can be easily obtained by other parties when necessary.

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.

The details of construction are as follows:

Suppose data owner A is about to store an encrypted file with keywords $W_1,...W_l$ on a cloud storage S, where $l \in Z^+$. 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 $W_1,...W_l$ using his public key A_{pub}. The file deposited in the cloud storage S by the data owner A is as follows:

$$MSG_{U2S} = [EMBEnc(A_{pub}, S_{pub}, m), KWEnc(A_{pub}, W_1), ..., KWEnc(A_{pub}, W_l)]$$

Where $EMBEnc$, $KWEnc$ are public key encryption algorithms. Finally, A appends to the encrypted file message with all the encrypted keywords and sends MSG_{U2S} to S.

Given a sufficiently large security parameter $K \in Z^+$, it runs IG to generate a prime q, two groups G_1 and G_2 of prime order q, and a bilinear map $e: G_1 \times G_1 \to G_2$, $g, h \in G_1$, $Z = e(g,g) \in G_2$, where g is a generator of G_1. Then it chooses two hash functions H_1, H_3 : $\{0,1\}^* \to G_1^*$, hash function $H_2 : G_2 \to \{0,1\}^{\log q}$, and hash function $H_4 : G_2 \to \{0,1\}^n$ for some n, where H_1, H_2, H_3 and H_4 are random oracles. Finally, it picks three random elements $a, b, c \in Z_q^*$ and computes g^a, g^b and g^c. The plaintext space includes $M \in \{0,1\}^n$ and $W \in \{0,1\}^*$. The cipher text space includes $C_M = G_1^* \times \{0,1\}^n$ and $C_W \in G_2$.

- Key Generation (KG): The data owner A's public key is $A_{pub} = g^a$ with the corresponding private key $A_{priv} = a$; the user U 's public/private key is $U_{pub} = g^b$, $U_{priv} = b$ respectively. The cloud provider S 's public key is $S_{pub} = g^c$ with the corresponding private key $S_{priv} = c$.
- Encryption (E): This encryption algorithm consists of KWEnc and EMBEnc. The data owner first picks a random element $r \in Z_q^*$.

  1) KWEnc(E_1) : To encrypt m 's keywords $W_1,...W_k$ $(k \in Z^+)$ under a data owner's public key g^a and a random element r, it computes $H_2(e(g^a, H_1(W_i))^r)$, where $W_i \in \{W_1,...W_k\}$, sets the cipher text $C_{W_i} = H_2(e(g^a, H_1(W_i))^r)$.

  2) EMBEnc(E_2): To encrypt the file message m under data owner's public key g^a, cloud provider's public key g^c and random element r, it picks a random element $\rho \in \{0,1\}^n$, and computes

  $$u_1 = h^r,$$
  $$u_2 = \rho \oplus H_4(e(h^a, g^c)^r),$$
  $$u_3 = m \cdot e(H_3(\rho), g^a)^r,$$

  and sets the cipher text $C_m = (u_1, u_2, u_3)$.
- Re-Encryption KeyGeneration (RG): Data owner A delegates to user U by publishing the re-encryption key $rk_{A \to U} = g^{abr}$, computed with U 's public key g^b.
- Tcompute: To retrieve the file containing keyword $W_j (j \in Z^+)$, user computes the trapdoor $T_{W_j} = H_1(W_j)^{1/b}$ using his/her private key $U_{priv} = b$, then sends the trapdoor to the cloud provider.
- Re-Encryption(R): to change the cipher text $C_m = (u_1, u_2, u_3)$ for A into a cipher text $C_U = (u_3, u_4)$ for U under the re-encryption key $rk_{A \to U} = g^{abr}$, it computes

  $$u_4 = e(H_3(\rho), rk_{A \to U}) = e(H_3(\rho), g^{abr}).$$

  The cloud provider sends C_U to the user.

Note. Since

$\rho = u_2 \oplus H_4(e(h^a, g^c)^r) = u_2 \oplus H_4(e(g^a, h^r)^c)$, the cloud provider can compute the intermediate value ρ with its private key c.

- Test：To determine whether a given file contains keyword W_j, the cloud provider tests whether $C_{W_i} = H_2(e(rk_{A\to U}, T_{W_j}))$.

  If so, $Test(rk_{A\to U}, C_{W_i}, T_{W_j})$ outputs 1, and 0 otherwise.

Note. If $W_i = W_j$, since $C_{W_i} = H_2(e(g^a, H_1(W_i))^r)$, then $C_{W_i} = H_2(e(g^a, H_1(W_j))^r) = H_2(e(g^{abr}, H_1(W_j)^{1/b})) = H_2(e(rk_{A\to U}, T_{W_j}))$)

- Decryption(D) : Given the cipher text $C_U = (u_3, u_4)$, it computes $m = u_3 / (u_4)^{\frac{1}{U_{priv}}} = u_3 / (u_4)^{\frac{1}{b}}$ to recover the message m.

Note that:

$$\frac{u_3}{(u_4)^{\frac{1}{b}}} = \frac{m \cdot e(H_3(\rho), g^a)^r}{(e(H_3(\rho), g^{ab})^r)^{\frac{1}{b}}} = \frac{m \cdot e(H_3(\rho), g^a)^r}{e(H_3(\rho), g^a)^r} = m$$
.

V. ANALYSIS

A. Security Analysis

1) Privacy for Keyword

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 A_1 be an IND-CPA adversary that has the advantage ε_1 in breaking KWEnc. Suppose A_1 makes at most $q_{H_2} > 0$ hash queries to H_2 and at most $q_T > 0$ trapdoor queries. Then there is an algorithm B_1 that solves the BDH problem with the advantage at least $\varepsilon_1' = 2\varepsilon_1 / \{e \cdot q_{H_2} \cdot (1+q_T)\}$, and a running time $O(time(\mathrm{A}_1))$.

Proof. The proof is similar to Lemma 4.2 in [6].

Privacy for Keyword guarantees the user's query privacy, namely, the cloud provider learns nothing about what the user's querying for in this process. In our scheme, the file is encrypted with the data owner's public key before its storage in the untrusted cloud. A user sends a trapdoor with inputting encrypted keyword to query for a file which including the encrypted keyword. The cloud provider will have no knowledge of the file's keyword, only if it obtains the private key of the data owner.

2) Privacy for Message

Our security definition quantifies over all encryption algorithms; in this case, we have two algorithms EMBEnc(E_2)and Re-Encryption(R), where an E_2 cipher text takes the form $u_3 = m \cdot e(H_3(\rho), g^a)^r$. This construction is equivalent to R cipher text of the form $u_4 = e(H_3(\rho), g^{ar})^b$. Now, it is clear if the E_2 cipher text of the form $u_3 = m \cdot e(H_3(\rho), g^a)^r$ is secure, then so is the R version, since E_2 cipher texts reveal more information. Thus, it suffices to argue the security of the E_2 cipher texts only.

Next, we show that EMBEnc is a semantically secure public key encryption if the BDH assumption holds. It is worth noticing that the outer attackers couldn't calculate ρ if the BDH assumption holds. Without loss of generality, we suppose that an IND-CPA adversary A_2 has already known ρ and could issue H_3 queries at any time.

Lemma 5.2 (Privacy for Message) Let H_3 be a random oracle from $\{0,1\}^*$ to G_1^* and H_4 be a random oracle from G_2 to $\{0,1\}^n$. Let A_2 be an IND-CPA adversary that has the advantage ε_2 against EMBEnc. Suppose A_2 makes $q_{H_3} > 0$ hash function queries to H_3 and $q_R > 0$ queries to $Request$. Then there is an algorithm B_2 that solves the BDH problem with the advantage at least $\varepsilon_2' = 2\varepsilon_2 / q_{H_3} q_R$ and a running time $O(time(\mathrm{A}_2))$.

Proof . See Appendix A.

3) Security for PRPS

We will study the security for our PRPS scheme according to Definition 3.4. The following theorem shows that PRPS is semantically secure if the BDH problem is hard.

Theorem 5.1 (Security for PRPS). Suppose the hash functions H_1, H_2, H_3 and H_4 are random oracles. Let A be an IND-CPA adversary consisting of two polynomial time algorithms A_1 and A_2. Let A_1 be an IND-CPA adversary that has the advantage ε_1 in breaking KWEnc. Suppose A_1 makes $q_T > 0$ trapdoor queries and $q_{H_2} > 0$ hash queries to H_2. Let A_2 be an IND-CPA adversary that has the advantage ε_2 against EMBEnc. Suppose A_2 makes $q_{H_3} > 0$ hash function queries to H_3 and $q_R > 0$ queries to $Request$. Let A be an IND-CPA adversary that has the advantage $\varepsilon = \varepsilon_1 + \varepsilon_2$ against the PRPS scheme. Then there is an algorithm B that solves the BDH problem with the advantage at least: $Adv_{\mathrm{B}} \geq 2\varepsilon_1 / \{e \cdot q_{H_2} \cdot (1+q_T)\} + 2\varepsilon_2 / q_{H_3} q_R$ That means the PRPS scheme is semantically secure under the BDH problem. Here $e \approx 2.71$ is the base of the natural logarithm. The running time of B is $O(time(\mathrm{A}))$.

Proof. PRPS includes two public key encryption algorithms, i.e. EMBEnc and KWEnc. Therefore, the proof follows directly from Lemma 5.1 and Lemma 5.2.

B. *Efficiency Analysis*

This section evaluates the efficiency of the PRPS scheme in terms of the computation overhead introduced by each operation. We use computation time to denote the computation overhead of the algorithm operated by different roles (for example, the data owner, the user). Encryption (KWEnc, EMBEnc) and Re-Encryption Key Generation are operated by the data owner; Re-Encryption and Test are operated by the cloud provider and the user's operation are Tcompute and Decryption.

Suppose the runtime of exponent arithmetic (EXP) is T_e, the runtime of hash arithmetic (Hash) is T_h and the runtime of arithmetic of bilinear pairings (Pairing) is T_b.

TABLE I.

COMPUTATION EFFICIENCY OF PRPS

		EXP	**Hash**	**Pairing**	**Total**
Data owner	***Encryption***	$5T_e$	$4T_h$	$3T_b$	$6T_e + 4T_h + 3T_b$
	Re-Encryption KeyGeneration	T_e			
Cloud Provider	***Re-Encryption***	$2T_e$	$2T_h$	$2T_b$	$2T_e + 3T_h + 3T_b$
	Test		T_h	T_b	
User	***Tcompute***		T_h		$T_e + T_h$
	Decryption	T_e			

The comparison in the runtimes for the cryptographic operations in PRPS scheme is given in TABLE I. These results indicate that the runtimes of hash arithmetic and exponent arithmetic operated by a user are much less than the ones of cloud provider's and data owner's operations. The scheme transfers most computation cost from the user to the cloud provider decrease the computation overhead and enhance the efficiency of the user. That makes sense to the application of cloud computing with thin clients.

Note. In our scheme, a data owner takes his own private key, the user's public key and a random element as the inputs, and produces re-encryption key $r_{A\to U} = (g^b)^{ar} = g^{abr}$. Thus, there is no need to deliver the user's private key to the data owner or interact with the third party for the re-encryption key, which implies that our PRPS scheme is *non-interactive*.

VI. 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 storage 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. We indicate that the challenging "Private Searching on Encrypted Data" problem is of independent interest and deserved further study.

APPENDIX A PROOF OF LEMMA 5.2

Proof. B_2 Is given $\rho \in \{0,1\}^n$, $\mu_0 = g, \mu_1 = g^{\alpha_2}$, $\mu_2 = g^{\beta_2}$, $\mu_3 = g^{\gamma_2} \in G_1$, where $\alpha_2, \beta_2, \gamma_2$ are random elements in Z_q^*. Its goal is to output $D_2 = e(g,g)^{\alpha_2\beta_2\gamma_2} \in G_2$. B_2 finds D_2 by interacting with A_2 as follows:

Keygen: B_2 sends (μ_0, μ_1) as the public key to A_2.

$H_3 - Queries$: B_2 maintains a list of tuples called $H_3 - List$, in which each entry is a tuple of the form $\langle \rho_j, f_j \rangle$. The list is initially empty. When A_2 issues a query to H_3, B_2 checks if ρ_i is already on $H_3 - List$ in the form of $\langle \rho_j, f_j \rangle$. If so B_2 responds to A_2 with $H_3(\rho_i) = f_i$. Otherwise, B_2 picks a random $d \in Z_q^*$, computes $f_i = \mu_2 . g^d = g^{\beta_2} . g^d \in G_1^*$ adds the tuple $\langle \rho_i, f_i \rangle$ to $H_3 - List$, and responds to A_2 with $H_3(\rho_i) = f_i$.

$H_4 - Queries$: B_2 maintains a list of tuples called $H_4 - List$, in which each entry is a tuple of the form $\langle r_j, l_j \rangle$. The list is initially empty. When A_2 issues a query to H_4, B_2 checks if r_i is already on $H_4 - List$ in the form of $\langle r_i, l_i \rangle$. If so, B_2 responds to A_2 with $H_4(r_i) = l_i$. Otherwise, B_2 picks a random string $l_i \in \{0,1\}^n$, adds the tuple $\langle r_i, l_i \rangle$ to $H_4 - List$, and responds to A_2 with $H_4(r_i) = l_i$.

$Request$: Next, for $i = 1$ up to $poly(k)$, A_2 can request:

a. $rk_{x\to T}$, a delegation to T from a party corrupted by A_2. A_2 can generate these delegations for as many corrupted users as it likes internally by running $(pk_x, sk_x) \leftarrow KG(1^k)$ and computing $rk_{x\to T} = (g^{\alpha_2})^{sk_x}$.

b. $rk_{T\to h}$, a delegation from T to an honest party h. The simulator randomly selects one values $r_h \leftarrow Z_q$, sets $rk_{T\to h} = (\mu_0)^{r_h} = g^{r_h}$ And $pk_h = g^{r_h}$, and sends $(pk_h, rk_{T\to h})$ to A_2. The corresponding secret key is $sk_h = r_h$.

c. $rk_{h\to T}$, a delegation to T from an honest party h. The simulator uses either the recorded value r_h from the previous step if the honest party already exists, or generates fresh random values for a new party, and computes $rk_{h\to T} = (g^{\alpha_2})^{r_h}$.

Challenge. A_2 outputs two messages m_0 and m_1 on which it wishes to be challenged. B_2 randomly picks $b_2 \in \{0,1\}$ and a random string $S_2 \in \{0,1\}^n$, and gives the cipher text $C_2 = (\mu_3, S_2)$ to A_2. Note that the decryption of the cipher text is:

$$\mu_3 = m \cdot (e(H_3(\rho), \mu_1)^{\gamma_2}) = m \cdot (e(H_3(\rho), g^{\alpha_2})^{\gamma_2})$$
$$= m \cdot (e(g^{\beta_2}.g^{d}, g^{\alpha_2})^{\gamma_2}) = m \cdot (e(g,g)^{\alpha_2\gamma_2(\beta_2+d)})$$

Hence, C_2 is a valid cipher text for m_{b_2} as required.

Guess: A_2 outputs its guess $b_2^{'} \in \{0,1\}$ for b_2, B_2 picks a random pair $\langle \rho_j, f_j \rangle$ from $H_3 - List$ and outputs f_j as the solution to the given instance of BDH.

Let Q_2 be the event that A_2 issues a query for f. From proof of Lemma 5.1, we know that $\Pr[Q_2] \geq 2\varepsilon_2$. That means A_2 will issue a query for f with the probability at least $2\varepsilon_2$. B_2 will choose the correct pair with the probability at least $1/q_{H_3}$ and succeed in $Request$ with the probability at least $1/q_R$, thus B_2 produces the correct answer with the probability at least $\varepsilon_2^{'} = 2\varepsilon_2 / q_{H_3}q_R$ as required. If A_2 has the advantage ε_2 against EMBEnc, then B_2 succeeds with probability $\varepsilon_2^{'} = 2\varepsilon_2 / q_{H_3}q_R$ .This contradicts the BDH assumption.

ACKNOWLEDGMENT

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

[1] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A View of Cloud Computing," *Communications of the ACM,* Vol. 53, pp.50-58,April 2010.

[2] Shucheng Yu, Cong Wang, Kui Ren and Wenjing Lou, "Achieving Secure, Scalable, and Fine-grained Data Access Control in Cloud Computing," *in Proceedings of INFOCOM 2010. IEEE,* 2010.

[3] G. Ateniese, K. Fu, M. Green, S. Hohenberger, "Improved proxy re-encryption schemes with applications to secure distributed storage," *ACM Transactions on Information and System Security (TISSEC).* 9 (1) (2006) 1–30.

[4] D. Boneh, G. Crescenzo, R. Ostrovsky, and G. Persiano. "Public Key Encryption with Keyword Search," *Proceedings of Eurocrypt 2004, Lecture Notes in Computer Science 3027, Springer-Verlag*. 2004. pp. 506-522.

[5] T. Diament, H. K. Lee, A. D. Keromytis and M. Yung, "The Dual Receiver Cryptosystem and its Application," *Proceedings of the ACM CCS 2004*, pp. 330-343.

[6] Qin Liu, Guojun Wang, Jie Wu, "An Efficient Privacy Preserving Keyword Search Scheme in Cloud Computing,"*in Computational Scinece and Engineering, CSE'09*, vol.2, 2009 , pp. 715 - 720.

[7] J. Shao, Z. Cao, X. Liang, H. Lin, "Proxy re-encryption with keyword search," *Information Science.*vol.180, pp. 2576–2587, 2010.

[8] R. Canetti, S. Hohenberger, "Chosen-cipher text secure proxy re-encryption," in: ACM CCS 2007, 2007. Full version: Cryptology ePrint Archieve: Report 2007/171.

[9] D. Boneh and M. Franklin. "Identity Based Encryption from the Weil Pairing," *SIAM J. of Computing*, 32 (3): 586-615, 2003.

[10] D. Song, D. Wagner, A. Perrig, "Practical techniques for searching on encrypted data". *IEEE Symp. On Research in Security and Privacy 2000*, IEEE. 2000. pp.44–55

[11] Y. Chang, M. Mitzenmacher, "Privacy preserving keyword searches on remote encrypted data". *Proceedings of ACNS2005*. *Lecture Notes in Computer Science* 3531, *Springer-Verlag*. 2005. pp.442–455.

[12] Melissa Chase, Seny Kamara, "Structured Encryption and Controlled Disclosure". *Proceedings of ASIACRYPT 2010, Lecture Notes in Computer Science 6477, Springer-Verlag*. 2010. pp. 577–594.

Xi Chen is currently pursuing the M.S degree with Department of Information and Communication Engineering in Beijing Jiaotong University. Her research interests include cryptographic protocols and information security.

Yong Li received his M.S. degree in Computer Science from Wuhan University in 2003, and the Ph.D. degree from State Key Laboratory of Information Security, Graduate University of Chinese Academy of Sciences in 2007.

Currently, he is an associate professor at the School of Electronic and Information Engineering, Beijing Jiaotong University. He has over 20 publications and filed 6 patents. His research interests include cryptographic protocols and information security.

Dr. Li is a member of the International Association for Cryptologic Research (IACR), Chinese Association for Cryptologic Research (CACR), Association for Computing Machinery (ACM), China Computer Federation (CCF). He served as organizing committee chair of the international conference on cloud computing (CloudCom 2009). He was the program committee member for the international conferences CloudCom 2009, CloudCom 2010, NCIS'11. He also served as peer reviewers for seveal international conferences and academic journals, such as Asiacrypt, PKC, FSE, ACNS, Journal of Systems and Software, High Technology Letters, etc.