Privacy preserving delegated word-search in the cloud

Kaoutar Elkhiyaoui, Melek Önen, and Refik Molva

EURECOM, Sophia-Antipolis, France
Email: {kaoutar.elkhiyaoui, melek.onen, refik.molva}@eurecom.fr

1 Introduction

In this work, we focus on the problem of privacy in data and computation outsourcing to the cloud and in particular on the problem of delegated word search. In the face of potentially malicious cloud providers the client or the data owner does not want to disclose any information about the outsourced data, yet they still want to take advantage of the highly parallel cloud environment. In [2], authors recently proposed a privacy preserving word search scheme for the MapReduce paradigm called PRISM which assures data confidentiality and query confidentiality. While meeting its original privacy objectives, PRISM does not allow third parties to issue word search queries without the disclosure of the data encryption key.

We suggest a new third-party word search solution based on PRISM that assures both *access delegation* and *user revocation*. The idea is to combine attribute-based encryption mechanisms and secure permutations so that only the parties having the correct credentials can retrieve the data encryption key and issue correct search queries.

2 Problem Statement

We consider a scenario where a client $\mathcal{C}$ outsources some privacy sensitive data to the cloud provider and wishes to later on perform some operations over it without revealing any details about the data. The operation we are focusing on is word search over encrypted data and in our scenario the client may wish to delegate part of the search operations to authorized third parties. For example, due to regulatory matters, some data (such as logs) still need to be searchable by third parties such as data protection commissioners. Most of existing privacy preserving word search solutions are intended to be executed by the owner of the data only, since generally the search queries have as input the encryption key used to encrypt the data. Revealing this key to other authorized users may cause some privacy/security exposures. Therefore, this paper focuses on the problem of key management whereby relevant keys should only be distributed to authorized entities and such entities can be revoked at any time after the key distribution phase by the data owner (i.e., the client).

The new privacy preserving word search solution should hence offer the same privacy guarantees as existing schemes against an honest-but-curious cloud while allowing third party lookup operations and user revocation. In accordance with the work of Blass et al. [2], the privacy requirements of delegated word search can therefore be summarized as follows:

- **data privacy against the cloud**: the cloud should not be able to discover any information about the stored data;
- **query privacy against the cloud**: the cloud should not be able to infer any details about word search queries;
- **authorized access with revocation**: only authorized entities should be able to lookup words in the cloud server database, while the data owner (i.e., the client $\mathcal{C}$) can revoke any authorized entity at any time.

Considering these privacy requirements, a new privacy preserving third-party word search mechanism consists of five phases:

- **Setup**: During which the client $\mathcal{C}$ encrypts its data in a way that allows further queries on the data. $\mathcal{C}$ uploads the data at the end of this phase together with some access policy associated with the data to the cloud server $\mathcal{S}$;
- **Delegate**: $\mathcal{C}$ delegates the rights for word search operations to a third party $\mathcal{U}$;
- **Query**: The client $\mathcal{C}$ or the delegate (i.e., authorized third party $\mathcal{U}$) prepares the query using the data encryption key and sends it to the cloud server $\mathcal{S}$, which in turn processes the query without discovering any additional information on the query or on its result;

- **Verification**: The client $\mathcal{C}$ or/and the delegate can make decisions on the search query based on the data sent by the cloud;
- **Revocation**: This additional step allows $\mathcal{C}$ to revoke a delegated third party $\mathcal{U}$; $\mathcal{C}$ informs $\mathcal{S}$ about this modification and updates the keying material accordingly. This phase should have a minimum impact on the overall computational and communication overhead of the solution.

3 Solution

Overview. The solutions we propose to address the privacy requirements described in Section 2 rely on a protocol for privacy preserving word search called PRISM [2]. PRISM involves a client $\mathcal{C}$ which outsources its encrypted files $\mathcal{F} = \{F_1, F_2, ..., F_m\}$ to an *honest-but-curious* cloud server $\mathcal{S}$, and which performs word search queries using its encryption key K. To ensure privacy against the honest-but-curious cloud server $\mathcal{S}$, PRISM builds upon a *stateful* secure encryption to prevent $\mathcal{S}$ from deriving any information about the outsourced files, and employs an efficient construction of private information retrieval [4] to prevent $\mathcal{S}$ from inferring any details about $\mathcal{C}$'s search queries and the corresponding results. Moreover, PRISM was designed to suit the MapReduce paradigm, and thus, it takes a full advantage of the parallel processing features akin to the cloud environment.

One of the building blocks of our solution is attribute-based encryption [1, 3]. The idea is that the client $\mathcal{C}$ encrypts its secret key K according to some access policy and stores the resulting ciphertext $C = E_{\mathsf{ABE}}(K)$ together with the encryption of the files $\mathcal{F} = \{F_1, F_2, ..., F_m\}$ at the cloud server $\mathcal{S}$. Now any third party possessing the attributes matching the access policy attached to the ciphertext C can decrypt the latter and derive K that further allows the third party to search the files $\{F_1, F_2, ..., F_m\}$ for any word of its choice. However, once a third party gets K, it can perform search queries even after its authorization has been revoked. A naive and an inefficient solution to this limitation of the basic scheme would be to re-encrypt the files $\{F_1, F_2, ..., F_m\}$ every time a revocation takes place. In order to come up with an efficient revocation scheme, we suggest to use a one-time permutation key K'_i that will help the honest-but-curious cloud server $\mathcal{S}$ hide its responses to the search queries from un-authorized/revoked third parties. Like K, the one-time permutation key K'_i will be encrypted by the cloud server $\mathcal{S}$ according to the client $\mathcal{C}$'s (updated) access policy using an attribute-based encryption scheme.

As a background for our solution, we first provide a brief description of PRISM.

3.1 PRISM

The PRISM protocol consists of the following operations:

- **Setup:** The client $\mathcal{C}$ picks a secret key K, then encrypts its files (word by word) using K and a stateful cipher $\mathcal{E}$ as follows:
  Without loss of generality, we assume that the client $\mathcal{C}$ outsources only one file $F = \{w_1, w_2, ..., w_N\}$ to the cloud server $\mathcal{S}$. We also assume that $L = \{\omega_1, \omega_2, ...\omega_n\}$ denotes the list of distinct words in F, and that each word $w_i \in F$ is associated with a counter γ_{w_i}, such that $\gamma_{w_i} = j$, if w_i is the j^{th} occurrence of some word $\omega \in L$.
  The encryption c_i of the word w_i is defined as:

  $$c_i = \mathcal{E}(w_i) = E_K(\omega, \gamma_{w_i})$$

  After encrypting the file F, $\mathcal{C}$ uploads $\mathcal{E}(F) = \{c_1, c_2, ..., c_N\}$ to the cloud server $\mathcal{S}$.
  Upon receipt of the encrypted file F, the cloud server $\mathcal{S}$ generates q binary (t,t)-matrices[1] $\mathcal{M}_j$ $(1 \leq j \leq q)$ that are initially set to 0. To fill the matrices $\mathcal{M}_j$, $\mathcal{S}$ uses two hash functions $H_1 : \{0,1\}^* \to \{0,1\}^{\log_2(t)} \times \{0,1\}^{\log_2(t)}$ and $H_2 : \{0,1\}^* \to \{0,1\}^q$ as depicted below:
  For each ciphertext $c_i \in \mathcal{E}(F)$:
  1.) $\mathcal{S}$ computes $H_1(c_i) = (x_i, y_i)$ which maps the ciphertext c_i to the position (x_i, y_i) in the matrices $\mathcal{M}_j$;
  2.) then, it computes $h_i = H_2(c_i)$. If the j^{th} bit of h_i is equal to 1, then $\mathcal{S}$ sets the bit at position (x_i, y_i) in the matrix $\mathcal{M}_j$ to 1, otherwise, the bit remains unchanged.

[1] t is a power of 2.

- **Query:** Assume that the client $\mathcal{C}$ wants to issue a search query for the word ω. It first computes $c = E_K(\omega, 1)$, then it computes $H_1(c) = (x, y)$ which returns the position that corresponds to c in the matrices $\mathcal{M}_j$. Next, $\mathcal{C}$ issues a PIR query as in [4] to fetch the y^{th} column of each matrix $\mathcal{M}_j$ $(1 \leq j \leq q)$.
- **Response:** $\mathcal{S}$ returns the matrix $R = (\mathbf{r}_1, \mathbf{r}_2, ..., \mathbf{r}_q)$ which consists of q PIR responses where each response $\mathbf{r}_j$ matches the y^{th} column of the matrix $\mathcal{M}_j$.
- **Verification:** After receiving $\mathcal{S}$'s response, $\mathcal{S}$ first retrieves the x^{th} element of each vector $\mathbf{r}_j$, unblinds it and gets a bit b_j. Now to check whether $\omega \in F$, $\mathcal{C}$ computes $h = H_2(\omega)$ and verifies whether the following equation holds:

$$h = h \wedge (b_1 || b_2 || ... || b_q)$$

For further details on the security and the privacy properties of PRISM, the reader may refer to [2].

3.2 Privacy preserving delegated word search

Without loss of generality, we assume that the client $\mathcal{C}$ outsources a file F to the cloud server $\mathcal{S}$. The new scheme involves the following operations:

- **Setup and delegate:** The client $\mathcal{C}$ encrypts F using its secret key K and the stateful encryption algorithm used in PRISM. Then, it encrypts K using a ciphertext-policy attribute-based encryption, let $C = E_{\mathsf{ABE}}(K)$. Finally, $\mathcal{C}$ forwards the encrypted file F together with the ciphertext C and the access policy associated with F to the cloud server $\mathcal{S}$.
  Upon receipt of F, C, and the corresponding access policy, $\mathcal{S}$ constructs q matrices $\mathcal{M}_j$ as in PRISM.
- **Query:** Assume that an authorized third party $\mathcal{U}$ wants to check whether F contains some word ω.
  First, $\mathcal{U}$ queries the cloud server $\mathcal{S}$ to retrieve the necessary secret information for the search query.
  Upon receipt of $\mathcal{U}$'s query, $\mathcal{S}$ responds with the ciphertext C that encrypts the secret key K and a second ciphertext C'_i which encrypts a one-time permutation key $K'_i \in \{0,1\}^\tau$ that was chosen randomly by $\mathcal{S}$ (τ is a security parameter). We note that both K and K'_i are encrypted according to the access policy associated with the file F.
  $\mathcal{U}$ decrypts the ciphertexts C and C'_i using its attributes and obtains the secret keys K and K'_i respectively.
  Let $\pi : \{0,1\}^\tau \times \{0,1\}^{\log_2(t)} \times \{0,1\}^{\log_2(t)} \to \{0,1\}^{\log_2(t)} \times \{0,1\}^{\log_2(t)}$ be a secure random permutation.
  After decrypting C and C'_i, $\mathcal{U}$ computes $(x, y) = H_1(E_K(\omega, 1))$ and $\pi(K'_i, (x, y)) = (x', y')$, it then issues a PIR query as in [4] to retrieve the y'^{th} column of a (t, t)-matrix.
- **Response:** After receiving $\mathcal{U}$'s PIR query, $\mathcal{S}$ permutes the q matrices $\mathcal{M}_j$ using the secure random permutation π and the secret key K'_i. We denote $\mathcal{M}'_j$ the permuted matrix $\mathcal{M}_j$. Next, $\mathcal{S}$ returns q PIR responses $(\mathbf{r}'_1, \mathbf{r}'_2, ..., \mathbf{r}'_q)$ such that each response $\mathbf{r}'_j$ matches the y'^{th} column of the matrix $\mathcal{M}'_j$.
- **Verification:** Upon receipt of $\mathcal{S}$'s response $(\mathbf{r}'_1, \mathbf{r}'_2, ..., \mathbf{r}'_q)$, the third party $\mathcal{U}$ retrieves the x'^{th} element of each vector $\mathbf{r}'_j$ and performs the verification as described in Section 3.1.
- **Revocation:** Whenever the client $\mathcal{C}$ wants to revoke some third party $\mathcal{U}$, it re-encrypts K using the new access policy, then forwards the new ciphertext and the new access policy to the cloud server $\mathcal{S}$.

Although the above solution is quite efficient on $\mathcal{U}$'s side, it reveals information about F to revoked third parties. While a revoked third party cannot check directly whether a word ω is in F by querying $\mathcal{S}$, it may still be able to derive information about the ciphertexts composing $\mathcal{E}(F)$. Using this information and the secret encryption key K that it obtained from previous executions, the revoked third party can perform a dictionary attacks to gradually learn the content of F.

To address this issue, we propose another solution that does not leak any information about F to revoked third parties at the expense of a higher computation cost at $\mathcal{U}$'s side. The idea is that instead of permuting the matrices $\mathcal{M}_j$, the cloud server $\mathcal{S}$ encrypts its response using the one-time permutation key K'_i. More precisely, the client $\mathcal{C}$ and the cloud server $\mathcal{S}$ proceed as follows to perform the search query:

- **Setup and delegate:** The client $\mathcal{C}$ encrypts F using its symmetric key K following the same encryption algorithm as in PRISM. Then, he encrypts K using a ciphertext-policy attribute-based encryption and obtains the ciphertext $C = E_{\mathsf{ABE}}(K)$. Finally, $\mathcal{C}$ forwards the encryption of F, the ciphertext C and the access policy associated with F to the cloud server $\mathcal{S}$.
  As in the previous solution and in PRISM, $\mathcal{S}$ fills q matrices $\mathcal{M}_j$ that will be used by $\mathcal{S}$ to answer the search queries.

- **Query:** Assume that some authorized third party $\mathcal{U}$ wants to check whether F contains some word ω. $\mathcal{U}$ first queries $\mathcal{S}$ to get the ciphertext C. Using its attributes, $\mathcal{U}$ decrypts C to get the encryption key K.
  By having access to $K, \mathcal{U}$ issues a PRISM query for the word ω.
- **Response:** After receiving $\mathcal{U}$'s query, $\mathcal{S}$ computes first its response to the PRISM query, which is composed of q vectors $(\mathbf{r}_1, \mathbf{r}_2, ..., \mathbf{r}_q)$ (we recall that each vector $\mathbf{r}_j$ corresponds to the y^{th} column of the matrix $\mathcal{M}_j$, where $(x, y) = H_1(E_K(\omega, 1)))$. Then, it picks a one-time key $K'_i \in \{0,1\}^\tau$ and encrypts each element of the vectors $\mathbf{r}_j$ separately. Let $\mathbf{r'}_j$ denote the encryption of $\mathbf{r}_j$ using the secret key K'_i.
  Next, $\mathcal{S}$ uses the access policy associated with F to compute a ciphertext-policy attribute-based encryption $C'_i = E_{\mathsf{ABE}}(K'_i)$. Finally, it returns the encryption $(\mathbf{r'}_1, \mathbf{r'}_2, ..., \mathbf{r'}_q)$ of the vectors $(\mathbf{r}_1, \mathbf{r}_2, ..., \mathbf{r}_q)$ and the ciphertext C'_i to $\mathcal{U}$.
- **Verification:** Upon receipt of C'_i and $(\mathbf{r'}_1, \mathbf{r'}_2, ..., \mathbf{r'}_q)$, the authorized third party $\mathcal{U}$ decrypts C'_i using its attributes and derives the one-time key K'_i. Then $\mathcal{U}$ decrypts the x^{th} element of each vector $\mathbf{r'}_j$ using K'_i. Finally, $\mathcal{U}$ carries out the verification of $\mathcal{S}$'s response as in PRISM.

4 Summary and Future Work

In this work, we introduced two solutions for privacy preserving delegated word search, i.e, solutions that enable authorized third parties to issue search queries to a cloud server database without disclosing the content of the database or the access patterns to the cloud server. The proposed solutions combine the PRISM protocol with techniques of attribute-based encryption and secure permutations to allow for both *access delegation* and *user revocation* at the expense of very little additional cost at the queriers' side.

The next step of our work is to design a protocol for privacy preserving delegated word search that do not leak any information about the content of the database other than the result of the search query to the authorized third parties. That is, at the end of the execution of the protocol, a third party only learns the value of a bit b, where $b = 1$ if the word the third party is looking for is in the cloud server's database, and $b = 0$ otherwise.

Bibliography

[1] J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-policy attribute-based encryption. In *Security and Privacy, 2007. SP '07. IEEE Symposium on*, pages 321–334, 2007.

[2] E.-O. Blass, R. di Pietro, R. Molva, and M. Önen. PRISM - Privacy-Preserving Search in MapReduce. In *Proceedings of the 12th Privacy Enhancing Technologies Symposium (PETS 2012)*. LNCS, July 2012.

[3] V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the 13th ACM conference on Computer and communications security*, CCS '06, pages 89–98. ACM, 2006. ISBN 1-59593-518-5.

[4] J. Trostle and A. Parrish. Efficient Computationally Private Information Retrieval from Anonymity or Trapdoor Groups. In *Proceedings of Conference on Information Security*, pages 114–128, Boca Raton, USA, 2010.