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 $C$ 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 $C$) 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 $C$ encrypts its data in a way that allows further queries on the data. $C$ uploads the data at the end of this phase together with some access policy associated with the data to the cloud server $S$;
- **Delegate**: $C$ delegates the rights for word search operations to a third party $U$;
- **Query**: The client $C$ or the delegate (i.e., authorized third party $U$) prepares the query using the data encryption key and sends it to the cloud server $S$, which in turn processes the query without discovering any additional information on the query or on its result;
- **Verification:** The client $C$ or/and the delegate can make decisions on the search query based on the data sent by the cloud;
- **Revocation:** This additional step allows $C$ to revoke a delegated third party $U$; $C$ informs $S$ 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 $C$ which outsources its encrypted files $F = \{F_1, F_2, ..., F_m\}$ to an honest-but-curious cloud server $S$, and which performs word search queries using its encryption key $K$. To ensure privacy against the honest-but-curious cloud server $S$, PRISM builds upon a stateful secure encryption to prevent $S$ from deriving any information about the outsourced files, and employs an efficient construction of private information retrieval [4] to prevent $S$ from inferring any details about $C$’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 $C$ encrypts its secret key $K$ according to some access policy and stores the resulting ciphertext $C = E_{\text{ABE}}(K)$ together with the encryption of the files $F = \{F_1, F_2, ..., F_m\}$ at the cloud server $S$. 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 $S$ 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 $S$ according to the client $C$’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 $C$ picks a secret key $K$, then encrypts its files (word by word) using $K$ and a stateful cipher $E$ as follows:

  Without loss of generality, we assume that the client $C$ outsources only one file $F = \{w_1, w_2, ..., w_N\}$ to the cloud server $S$. 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 $\gamma_{w_i}$, 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:

  $$c_i = E_{\omega} = E_{\omega}(w_i)$$

  After encrypting the file $F$, $C$ uploads $E(F) = \{c_1, c_2, ..., c_N\}$ to the cloud server $S$.

  Upon receipt of the encrypted file $F$, the cloud server $S$ generates $g$ binary $(t, t)$-matrices $1 \leq j \leq q$ that are initially set to $0$. To fill the matrices $M_j$, $S$ uses two hash functions $H_1 : \{0, 1\}^* \rightarrow \{0, 1\}^{\log_2(t)} \times \{0, 1\}^{\log_2(t)}$ and $H_2 : \{0, 1\}^* \rightarrow \{0, 1\}$ as depicted below:

  For each ciphertext $c_i \in E(F)$:

  1. $S$ 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 $M_j$;

  2. then, it computes $h_i = H_2(c_i)$. If the $j$th bit of $h_i$ is equal to 1, then $S$ sets the bit at position $(x_i, y_i)$ in the matrix $M_j$ to 1, otherwise, the bit remains unchanged.

---

1. $t$ is a power of 2.
derive information about the ciphertexts composing

\( E_S U \) parties at the expense of a higher computation cost at

that it obtained from previous executions, the revoked third party can perform a dictionary attacks to gradually learn

While a revoked third party cannot check directly whether a word

\( S \)

\( S \)

\( F \)

\( F \)

\( C \)

\( C \)

\( C \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)

\( S \)
- **Query:** Assume that some authorized third party $U$ wants to check whether $F$ contains some word $\omega$. $U$ first queries $S$ to get the ciphertext $C$. Using its attributes, $U$ decrypts $C$ to get the encryption key $K$.

By having access to $K$, $U$ issues a PRISM query for the word $\omega$.

- **Response:** After receiving $U$'s query, $S$ computes first its response to the PRISM query, which is composed of $q$ vectors $(r_1, r_2, ..., r_q)$ (we recall that each vector $r_j$ corresponds to the $y$th column of the matrix $M_j$, where $(x, y) = H_1(E_K(\omega, 1)))$. Then, it picks a one-time key $K'_i \in \{0, 1\}^*$ and encrypts each element of the vectors $r_j$ separately. Let $r'_j$ denote the encryption of $r_j$ using the secret key $K'_i$.

Next, $S$ uses the access policy associated with $F$ to compute a ciphertext-policy attribute-based encryption $C'_i = E_{ABE}(K'_i)$. Finally, it returns the encryption $(r'_1, r'_2, ..., r'_q)$ of the vectors $(r_1, r_2, ..., r_q)$ and the ciphertext $C'_i$ to $U$.

- **Verification:** Upon receipt of $C'_i$ and $(r'_1, r'_2, ..., r'_q)$, the authorized third party $U$ decrypts $C'_i$ using its attributes and derives the one-time key $K'_i$. Then $U$ decrypts the $x$th element of each vector $r'_j$ using $K'_i$. Finally, $U$ carries out the verification of $S$'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.

### Acknowledgments

This work was partially supported by the Cloud Accountability project - A4Cloud, grant EC 317550.

### Bibliography


