Nothing is for Free: Security in Searching Shared & Encrypted Data

Qiang Tang

Abstract—**Most existing symmetric searchable encryption schemes aim at allowing a user to outsource her encrypted data to a cloud server and delegate the latter to search on her behalf. These schemes do not qualify as a secure and scalable solution for the multi-party setting, where users outsource their encrypted data to a cloud server and selectively authorize each other to search. Due to the possibility that the cloud server may collude with some malicious users, it is a challenge to have a secure and scalable multi-party searchable encryption (MPSE) scheme. This is shown by our analysis on the Popa-Zeldovich scheme, which says that an honest user may leak all her search patterns even if she shares only one of her documents with another malicious user. Based on our analysis, we present a new security model for MPSE by considering the worst-case and average-case scenarios, which capture different server-user collusion possibilities. We then propose a MPSE scheme by employing the bilinear property of Type-3 pairings, and prove its security based on the Bilinear Diffie-Hellman Variant (BDHV) and Symmetric eXternal Diffie-Hellman (SXDH) assumptions in the random oracle model.**

Index Terms—**Multi-party Searchable Encryption (MPSE), Data Privacy, Trapdoor Privacy, Pairing.**

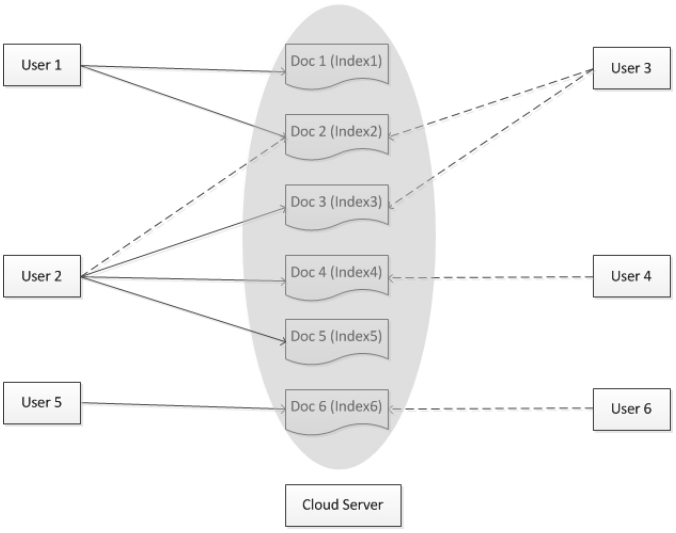

Fig. 1. Search Shared & Encrypted Data

I. Introduction

With the advancement of information technologies, particularly cloud storage services, information outsourcing and sharing have become ubiquitous in our life. For example, a user Alice may store her data at Dropbox and share them with her friends, in the mean time she may also have access to her friends' data. Due to the private nature of personal data, there is an inherent need for a user to selectively share her data with different recipients. In practice, what a user can do is to set some access control policies and then rely on the cloud server (e.g. Dropbox) to enforce them. Unfortunately, this approach is not realistic due to two reasons. One is that the users have no means to prevent the server from accessing their data. The other is that, even if the server is benign, it may also be forced to share users' data with other parties (e.g. by the USA PATRIOT Act).

A toy example is shown in Fig. 1. The real line from a user to index means that the user owns the index, while a dashed line means that the user is authorized by the index owner to search.

A. Problem Statement

The concept of searchable encryption provides a promising direction in solving the privacy problem when outsourching data to the cloud. Such schemes allow users to store their data in encrypted form at an untrusted server, and then delegate the server to search on their behalf by issuing a trapdoor (i.e. encrypted keyword). A detailed survey of searchable encryption schemes can be found in [27].

As to the specific setting where multiple users store and share their data with each other in the cloud, we need a new primitive, namely multi-party searchable encryption (MPSE) schemes in the symmetric setting. MPSE can be regarded as a multi-party version of the symmetric searchable encryption proposed by Song et al. [25]. Briefly, a MPSE scheme allows every user to build an encrypted index for each of her documents and store it at a cloud server. The index contains a list of encrypted keywords, as well as some authorization information which selectively authorizes other users to search over this index.

Our objective is first to formulate MPSE and its security properties and then to provide a scalable and secure construction. Informally, the scalability and security criteria in our mind are the following:

- As to scalability, the size of trapdoors and indexes should be of constant size. If a user Alice wants to allow another X users to search over her index, then the size of this index should not be linear in X. If Alice has been authorized by another Y users to

The author is affiliated with APSIA group, SnT, University of Luxembourg, Luxembourg. e-mail: qiang.tang@uni.lu

search their indexes, when Alice wants to search for a keyword w then the size of her trapdoor should not be linear in Y.
- As to security, there should not be any additional trusted third party (TTP) involved, and the information leaked to the server and malicious users should be minimized. Referring to Fig. 1, the cloud server will inevitably know which indexes match which search queries, and the cloud server and User 1 together will potentially be able to recover what User 3 is searching for (since Index2 is generated by User 1 and the cloud server is delegated by User 3 to search over Index2). But, other leakages should be prevented.

B. Related Work

The investigation of searchable encryption schemes in the symmetric setting was initiated by Song et al. [25] and followed up by many others (e.g. [8], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [23], [24]). This setting typically assumes one user and one server, where the user can generate searchable contents and stores them at the server, and later delegate the server to search on her behalf. To address our problem with any of these schemes, one needs to independently generate a key to protect each document (i.e. instantiate the scheme once for every document) and then shares the key with the users who will be authorized to search this document. As a result, the number of trapdoors required to search for a keyword will equal to the number of documents (or, indexes), so that the solution obviously does not satisfy our scalability criteria. Some other schemes are intended for more complex settings (e.g. [2], [4], [5], [13], [14], [15], [26], [22]) which are analyzed in more details below.

1) Some Existing Multi-User Schemes: Curtmola et al. [13] proposed the concept of multi-user searchable encryption schemes, where a user can authorize multiple other users to search her encrypted data. However, the proposed primitive does not take into account the fact that the same user may also be authorized to search other users' data and the corresponding security issues. As a result, the primitive from [13] offers a solution for a much more simplified problem than ours, and it seems not trivial to construct a scalable solution for our problem based on their scheme. In the work of Bao et al. [4], a new party, namely user manager, is introduced into the system, to manage multiple users' search capabilities (e.g. enable them to search each other's data). In this extension, the user manager needs to be fully trusted since it is capable of submitting search queries and decrypting encrypted data. This conflicts with our security criteria (i.e. there should not be additional TTP involved). The schemes of Dong, Russello and Dulay [14], [15] have similar issues. In the work from [2], [5], [26], the authors have investigated order preserving encryption, where the ciphertexts preserve the order the plaintexts so that every entity can perform an equality comparison. Clearly, these schemes also conflict with our security criteria (i.e. leak minimal information to the server).

2) Popa-Zeldovich Scheme: The Popa-Zeldovich scheme [22] is the only available work that addresses a similar problem to ours. Let λ be the security parameter, their scheme consists of the following algorithms.

- $\mathsf{MK.Setup}(\lambda)$: run $\mathcal{G}_{\mathcal{P}}$ to generate $\Gamma = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, g_2)$, and return the public parameters $params = (\Gamma, \mathsf{H}_1, \mathsf{H}_2)$, where $\mathsf{H}_1 : \{0,1\}^* \rightarrow \mathbb{G}_1$ and $\mathsf{H}_2 : \mathbb{G}_T \times \mathbb{G}_T \rightarrow \{0,1\}^*$ are two hash functions. The bilinear pairing parameter generation algorithm $\mathcal{G}_{\mathcal{P}}$ is defined in Section II.
- $\mathsf{MK.KeyGen}(params)$: randomly select k from $\mathbb{Z}_p$ and return k. Note that this algorithm is used by a user to generate her long-term private key (denoted as uk) and also generate an independent secret key (denoted as fk) for protecting each of her documents.
- $\mathsf{MK.Delta}(uk, fk)$: return $\Delta = g_2^{\frac{fk}{uk}}$.
- $\mathsf{MK.Token}(uk, w)$: return $tk = \mathsf{H}_1(w)^{uk}$.
- $\mathsf{MK.Enc}(fk, w)$: return $c = (r, \mathsf{H}_2(r, \hat{e}(\mathsf{H}_1(w), g_2)^{fk}))$, where $r \in_R \mathbb{G}_T$.
- $\mathsf{MK.Adjust}(tk, \Delta)$: return $tk' = \hat{e}(tk, \Delta)$.
- $\mathsf{MK.Match}(tk', c)$: parse c as (r, h), return $\mathsf{H}_2(r, tk') \stackrel{?}{=} h$.

Next, we show that this scheme leaks unnecessary information to the server, so that it does not satisfy our security criteria. Informally, the granularity of authorization is very coarse, i.e. an honest user may leak the keywords in all her search queries to the server if she authorizes a malicious user to search her index.

Let the long-term user keys of U_i and U_j be denoted as uk_i and uk_j respectively. Suppose that U_i wants to build an index $index0$ and authorizes U_j to search, she performs as follows:

1) U_i runs $\mathsf{MK.KeyGen}$ to generate a secret key fk_0, and runs $\mathsf{MK.Enc}$ to generate $index0 = \mathsf{MK.Enc}(fk_0, w_0)$. For simplicity, we assume the index encodes only one keyword w_0. U_i stores $index0$ and $\Delta_i = \mathsf{MK.Delta}(uk_i, fk_0) = g_2^{\frac{fk_0}{uk_i}}$ at the server. Here we assume U_i wants to enable herself to search over $index0$.
2) U_i and U_j magically computes $\Delta_j = \mathsf{MK.Delta}(uk_j, fk_0) = g_2^{\frac{fk_0}{uk_j}}$ and stores it at the server. Note that the scheme does not specify how Δ_j should be computed (see Remark 1 below), but this does not affect our security analysis.

Now suppose that U_j colludes with the server, then the latter can trivially recover $g_2^{fk_0}$. Afterwards, for any token $tk = \mathsf{H}_1(w)^{uk_i}$ from U_i, the server can first compute $\hat{e}(tk, \Delta_i) = \hat{e}(\mathsf{H}_1(w), g_2^{fk_0})$, and then identify w by computing $\hat{e}(\mathsf{H}_1(w'), g_2^{fk_0})$ for all w' given that the keyword set is polynomial size. Note that, even if w is from a set of super-polynomial size, the server can exclude a lot of possibilities based on its try-and-error. Ideally, the server and U_j should only be able to tell whether w equals to

w_0 or not. This is captured in the the worst-case trapdoor privacy property of our security model in Section IV-B.

Remark 1: Besides the security issue, Popa and Zeldovich did not explicitly specify how a user U_i can authorize another user U_j to search his index. Take the above analysis as example. The first possibility is that U_i may want to send the secret key fk_0 to U_j through a secure channel, so that U_j can generate Δ_j. In this case, if U_i would like to authorize a large number of users to search her index, she needs to communicate with all of them for sharing her secret key. Moreover, such communications are required for every index, so the solution is clearly unscalable. The second possibility is that U_i may want to run an interactive protocol with U_j for the latter to generate Δ_j without revealing fk_0 in clear. This may be even worse than in the first possibility.

C. Solution Intuition

Conceptually, in the context of MPSE, an individual user can act as a data owner and/or a follower.

- Data owner: in this case, the user will generate the contents and indexes and allow other users to search.
- Follower: in this case, the user will be authorized by others to search their data.

When a user Alice acts a follower and searches another user's data (e.g. Bob's), it is easy to say that Bob can figure out the keyword in Alice's query if he colludes with the server. As such, Alice may not want a random user Eve to authorize her to search his data. It is crucial to have a secure and efficient procedure for Alice to allow another user Bob to authorize her to search his data. In this case, how to prevent Bob from colluding with the server seems to be a difficult problem, and we leave it as a future work.

In contrast, as a data owner, a user Alice has more control over how the indexes are constructed. As a result, when Alice searches her own data (which might be authorized to other users as well), it is desirable to have more privacy protection to the keywords in the search queries. The security model and scheme from [22] fall short in this aspect, while we will address the issue in this paper.

D. Our Contribution

We first present a new formulation for MPSE. Compared with [22], we explicitly introduce a Follow algorithm, which enables a user Alice to assign a token to other users through a public channel. With this token, other users can authorize Alice to search their indexes without any further interaction with Alice. Since the communication of token is once for all, we solve the inherent scalability problem in the Popa-Zeldovich scheme [22], as indicated in Remark 1.

We then present a security model for MPSE by considering the worst-case and average-case scenarios. In the worst-case scenario, we assume that every user except the honest user will collude with the cloud server, and we define the worst-case data and trapdoor privacy properties. It is worth noting that the worst-case trapdoor privacy property eliminates the aforementioned information leakage in the Popa-Zeldovich scheme. In the average-case scenario, we assume that a group of honest users, who do not share data from any one outside of the group, will not collude with the cloud server, and we define a hybrid privacy property.

We finally propose a MPSE scheme by employing the bilinear property of Type-3 pairings, and prove its security based on the Bilinear Diffie-Hellman Variant (BDHV) and Symmetric eXternal Diffie-Hellman (SXDH) assumptions in the random oracle model. Similar to the Popa-Zeldovich scheme, the proposed scheme is also motivated by the PEKS construction of [6]. But, it is different in the sense that an index contains separate searchable contents for the data owner and other authorized users, in order for the scheme to be provably secure in our security model. As a side effect, we do not need the eXternal Diffie-Hellman Variant (XDHV) assumption (required in [22]) which is stronger than the SXDH assumption.

E. Organization

The rest of this paper is organized as follows. In Section II, we present the preliminary knowledge. In Section III, we present the algorithmic definition for MPSE. In Section IV, we present the security model for MPSE. In Section V, we present our main construction for MPSE. In Section VI, we conclude the paper.

II. Preliminary

Throughout the paper, we use the following notation. $x\|y$ means the concatenation of x and y, P.P.T. means probabilistic polynomial time, and $x \xleftarrow{\$} \mathcal{A}^{O_1,O_2,\cdots}(m_1,m_2,\cdots)$ means that x is the output of the algorithm $\mathcal{A}$ which runs with the input $m_1,m_2,\cdots$ and access to oracles $O_1,O_2,\cdots$. When X is a set, $x \in_R X$ means that x is chosen from X uniformly at random, and $|X|$ means the size of X. For $b \in \{0,1\}$, $\bar{b}$ represents $1-b$.

A function $P(\lambda) : \mathbb{Z} \to \mathbb{R}$ is said to be *negligible* with respect to λ if, for every polynomial $f(\lambda)$, there exists an integer N_f such that $P(\lambda) < \frac{1}{f(\lambda)}$ for all $\lambda \geq N_f$. If $P(\lambda)$ is negligible, then we say $1-P(\lambda)$ is *overwhelming*.

A. Pairing and Hardness Assumptions

A prime-order bilinear group generator is an algorithm $\mathcal{G}_{\mathcal{P}}$ that takes as input a security parameter λ and outputs a description $\Gamma = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, g_1, g_2)$ where:

- $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ are groups of prime-order p with efficiently-computable group laws.
- g_1 is a randomly-chosen generator of $\mathbb{G}_1$ and g_2 is a randomly-chosen generator of $\mathbb{G}_2$.

- $\hat{e}$ is an efficiently-computable bilinear pairing $\hat{e}: \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$, i.e., a map satisfying two properties:
  - Bilinearity: $\hat{e}(g_1^a, g_2^b) = \hat{e}(g_1, g_2)^{ab}$ for all $a, b \in \mathbb{Z}_p$;
  - Non-degeneracy: $\hat{e}(g_1, g_2) \neq 1$.

According to [9], a pairing is Type-3 if there is no efficiently computable isomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$. The following assumption requires a Type-3 pairing.

Given $\Gamma = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, g_1, g_2)$, the Symmetric eXternal Diffie-Hellman (SXDH) assumption [3] holds if, for every P.P.T. attacker $\mathcal{A}$, the advantage $\epsilon_{sxdh} = \max\{\epsilon_1, \epsilon_2\}$ is negligible for $x_1, x_2, x_3, x_4, r_1, r_2 \in_R \mathbb{Z}_p$.

$$\epsilon_1 = |\Pr[\mathcal{A}(\Gamma, g_1^{x_1}, g_1^{x_2}, g_1^{x_1 x_2}) = 1] - \Pr[\mathcal{A}(\Gamma, g_1^{x_1}, g_1^{x_2}, g_1^{r_1}) = 1]|$$

$$\epsilon_2 = |\Pr[\mathcal{A}(\Gamma, g_2^{x_3}, g_2^{x_4}, g_2^{x_3 x_4}) = 1] - \Pr[\mathcal{A}(\Gamma, g_2^{x_3}, g_2^{x_4}, g_2^{r_2}) = 1]|$$

This essentially says that the Decisional Diffie-Hellman (DDH) assumption holds for both $\mathbb{G}_1$ and $\mathbb{G}_2$.

Given $\Gamma = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, g_1, g_2)$, the Bilinear Diffie-Hellman Variant (BDHV) assumption [22] holds if, for $x_1, x_2, x_3 \in_R \mathbb{Z}_p$ and $R \in_R \mathbb{G}_T$, every P.P.T. attacker $\mathcal{A}$'s advantage $\epsilon_{bdhv} = |\Pr[\mathcal{A}(\Gamma, X_0) = 1] - \Pr[\mathcal{A}(\Gamma, X_1) = 1]|$ is negligible.

$$X_0 = (g_1^{x_1}, g_2^{\frac{1}{x_1}}, g_2^{\frac{x_2}{x_1}}, g_1^{x_3}, \hat{e}(g_1, g_2)^{x_2 x_3}), X_1 = (g_1^{x_1}, g_2^{\frac{1}{x_1}}, g_2^{\frac{x_2}{x_1}}, g_1^{x_3}, R).$$

Lemma 1: Suppose an attacker has the advantages ϵ_{bdhv} and ϵ_{sxdh} in solving the BDHV and SXDH problems, then its advantage in distinguishing Y_0 and Y_1 is at most $\epsilon_{bdhv} + \epsilon_{sxdh}$ where $x_1, x_2, x_3 \in_R \mathbb{Z}_p$ and $R_0, R_1 \in_R \mathbb{G}_T$.

$$Y_0 = (g_1^{x_1}, g_2^{\frac{1}{x_1}}, g_2^{\frac{x_2}{x_1}}, g_1^{x_3}, g_1^{x_4}, \hat{e}(g_1, g_2)^{x_2 x_3}, \hat{e}(g_1, g_2)^{x_2 x_4})$$

$$Y_1 = (g_1^{x_1}, g_2^{\frac{1}{x_1}}, g_2^{\frac{x_2}{x_1}}, g_1^{x_3}, g_1^{x_4}, R_0, R_1)$$

Proof. For the attacker, we consider another input Y_2, where $Y_2 = (g_1^{x_1}, g_2^{\frac{1}{x_1}}, g_2^{\frac{x_2}{x_1}}, g_1^{x_3}, g_1^{x_4}, R_0, R_0^{\frac{x_4}{x_3}})$

From Y_0 and Y_2 we can unanimously obtain X_0 and X_1 (defined in the BDHV assumption) by getting rid of two elements (e.g. the last and third last ones), therefore $|\Pr[\mathcal{A}(\Gamma, Y_0) = 1] - \Pr[\mathcal{A}(\Gamma, Y_2) = 1]| \geq \epsilon_{bdhv}$. On the other hand, from X_0 and X_1 we can unanimously obtain Y_0 and Y_2 by adding two elements (e.g. choose $r \in_R \mathbb{Z}_p$ and set $g_1^{x_4} = g_1^{x_3 r}$, and generate a new element by rasing the last element of X_0 or X_1 to the power r), therefore $|\Pr[\mathcal{A}(\Gamma, Y_0) = 1] - \Pr[\mathcal{A}(\Gamma, Y_2) = 1]| \leq \epsilon_{bdhv}$. As a result, we have $|\Pr[\mathcal{A}(\Gamma, Y_0) = 1] - \Pr[\mathcal{A}(\Gamma, Y_2) = 1]| = \epsilon_{bdhv}$.

As to distinguishing Y_1 and Y_2, note the fact that $g_1^{x_1}, g_2^{\frac{1}{x_1}}, g_2^{\frac{x_2}{x_1}}$ are irrelevant to the rest of Y_1 and Y_2, we have $|\Pr[\mathcal{A}(\Gamma, Y_1) = 1] - \Pr[\mathcal{A}(\Gamma, Y_2) = 1]|$ equals to $|\Pr[\mathcal{A}(\Gamma, \alpha_0) = 1] - \Pr[\mathcal{A}(\Gamma, \alpha_1) = 1]|$, where

$$\alpha_0 = (g_1^{x_3}, g_1^{x_4}, R_0, R_1), \ \alpha_1 = (g_1^{x_3}, g_1^{x_4}, R_0, R_0^{\frac{x_4}{x_3}})$$

Furthermore, it is obvious that $|\Pr[\mathcal{A}(\Gamma, \alpha_0) = 1] - \Pr[\mathcal{A}(\Gamma, \alpha_1) = 1]| \leq |\Pr[\mathcal{A}(\Gamma, \beta_0) = 1] - \Pr[\mathcal{A}(\Gamma, \beta_1) = 1]|$, where

$$x_5, x_6 \in_R \mathbb{Z}_p, \beta_0 = (g_1^{x_3}, g_1^{x_4}, g_1^{x_5}, g_1^{x_6}), \ \beta_1 = (g_1^{x_3}, g_1^{x_4}, g_1^{x_5}, g_1^{\frac{x_4 x_5}{x_3}}).$$

Finally, $|\Pr[\mathcal{A}(\Gamma, \beta_0) = 1] - \Pr[\mathcal{A}(\Gamma, \beta_1) = 1]| \leq \epsilon_{sxdh}$ so that $|\Pr[\mathcal{A}(\Gamma, Y_1) = 1] - \Pr[\mathcal{A}(\Gamma, Y_2) = 1]| \leq \epsilon_{sxdh}$.

Based on $|\Pr[\mathcal{A}(\Gamma, Y_0) = 1] - \Pr[\mathcal{A}(\Gamma, Y_2) = 1]| = \epsilon_{bdhv}$ and $|\Pr[\mathcal{A}(\Gamma, Y_1) = 1] - \Pr[\mathcal{A}(\Gamma, Y_2) = 1]| \leq \epsilon_{sxdh}$, we have $|\Pr[\mathcal{A}(\Gamma, Y_0) = 1] - \Pr[\mathcal{A}(\Gamma, Y_1) = 1]| \leq \epsilon_{bdhv} + \epsilon_{sxdh}$. The lemma now follows. ∎

Given $\Gamma = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, g_1, g_2)$, the n-Parallel Decisional Diffie-Hellman (PDDH$_n$) assumption [1] holds in $\mathbb{G}_1$ if, for $x_1, x_2, \cdots, x_n \in_R \mathbb{Z}_p$ and $R_1, R_2, \cdots, R_n \in_R \mathbb{G}_1$, every P.P.T. attacker $\mathcal{A}$'s advantage $\epsilon_{pddh_n} = |\Pr[\mathcal{A}(\Gamma, Z_0) = 1] - \Pr[\mathcal{A}(\Gamma, Z_1) = 1]|$ is negligible.

$$Z_0 = (g_1^{x_1}, g_1^{x_2}, \cdots, g_1^{x_n}, g_1^{x_1 x_2}, g_1^{x_2 x_3}, \cdots, g_1^{x_n x_1}),$$

$$Z_1 = (g_1^{x_1}, g_1^{x_2}, \cdots, g_1^{x_n}, R_1, R_2, \cdots, R_n).$$

It has been proven that this assumption is equivalent to DDH assumption in $\mathbb{G}_1$ [1], namely $\epsilon_{ddh} \leq \epsilon_{pddh_n} \leq n\epsilon_{ddh} \leq n\epsilon_{sxdh}$. Next, we review a more generalized assumption.

Given $\Gamma = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, g_1, g_2)$, the n-Multi Decisional Diffie-Hellman (M-DDH$_n$) assumption [10] holds in $\mathbb{G}_1$ if, for $x_1, x_2, \cdots, x_n \in_R \mathbb{Z}_p$ and $R_1, R_2, \cdots, R_{\frac{n(n-1)}{2}} \in_R \mathbb{G}_1$, every P.P.T. attacker $\mathcal{A}$'s advantage $\epsilon_{m-ddh_n} = |\Pr[\mathcal{A}(\Gamma, U_0) = 1] - \Pr[\mathcal{A}(\Gamma, U_1) = 1]|$ is negligible.

$$U_0 = (g_1^{x_1}, g_1^{x_2}, \cdots, g_1^{x_n}, g_1^{x_i x_j} (1 \leq i < j \leq n))$$

$$U_1 = (g_1^{x_1}, g_1^{x_2}, \cdots, g_1^{x_n}, R_1, R_2, \cdots, R_{\frac{n(n-1)}{2}}).$$

The M-DDH$_n$ assumption can be reduced to the DDH assumption in $\mathbb{G}_1$ [10]. More precisely, we have $\epsilon_{ddh} \leq \epsilon_{m-ddh_n} \leq n^2 \epsilon_{ddh} \leq n^2 \epsilon_{sxdh}$.

In a similar way, both PDDH$_n$ and M-DDH$_n$ assumptions can be defined for $\mathbb{G}_2$.

III. New Definition for MPSE

A MPSE scheme involves two types of entities: a cloud server and a set of users U_i ($1 \leq i \leq N$) where N can be any integer and should be a polynomial in the security parameter. Every user can generate encrypted indexes and authorize others to search, while the cloud server stores all encrypted contents and search on the users' behalf. We envision the following workflow for MPSE.

1) The users together set up some global public parameters.
2) Every user U_i, for $1 \leq i \leq N$, generates a master public/private key pair. The private key is mainly used to for three purposes.
   - Generate tokens to follow other users who can then authorize U_i to search their encrypted indexes.
   - Derive document-specific secret keys to generate encrypted indexes.
   - Generate trapdoors to search over encrypted indexes.
3) For every her document, U_i first extracts a list of keywords and then derives a document-specific key from her master private key to generate the encrypted index, which contains an encryption of

the list of keywords and some authorization information indicating who can search. The index is outsourced to the cloud server.
4) To search for a specific keyword, U_i issues the cloud server a trapdoor, generated based on her master private key and the keyword.
5) With the trapdoor from U_i, the cloud server first selects the encrypted indexes (from both U_i and other users) whose attached authorization information indicates that U_i is authorized to search. Then, for each selected index, the cloud server runs a match algorithm to decide whether the index contains the same keyword as that in the trapdoor.

Intuitively, we wish U_i's master private key to establish a link between her trapdoors and all indexes that have been authorized to her (generated by herself and others). As a result, U_i is able to search all indexes with only one trapdoor.

Remark 2: The authorization information in each index gives the server an indication who can search which index. This is crucial for the server's operational efficiency in practice. Otherwise, with a search request, the server needs to try all indexes in its database. Consider the server may have many users and each of them may store a lot of indexes, the absence of explicit authorization information will make it impossible to have a practical solution.

A. Algorithmic Definition for MPSE

Let λ be the security parameter. A MPSE scheme consists of the following algorithms.

- $\mathsf{Setup}(\lambda)$: Taking the security parameter λ as input, it sets up the global public parameter *params*. It is required that *params* includes a definition for a keyword space $\mathcal{W}$.
- $\mathsf{KeyGen}(params)$: Run by user U_i, for every $1 \leq i \leq N$, it generates a master public/private key pair (MPK_i, MSK_i).
- $\mathsf{Follow}(MSK_i, MPK_j)$: Run by U_i, it generates a token $TK_{i\to j}$. We use $i \to j$ to denote U_i follows U_j.
- $\mathsf{Formindex}(MSK_i, doc, \mathcal{TK})$: The *doc* parameter is a keyword set $\{w_1, w_2, \cdots, w_n\}$ where no keyword should repeat, and $\mathcal{TK}$ contains tokens of the form $TK_{t\to i}$. If U_i wants to authorize herself to search, instead of $TK_{i\to i}$, we assume MPK_i should be included in $\mathcal{TK}$. U_i performs as follows.
  1) Select a unique identifier *id* for the index, and generate a document-specific secret key FK_{id}.
  2) Based on FK_{id}, run the Enc algorithm to generate $TAG_{id} = \{Tag_{w_1}, Tag_{w_2}, \cdots, Tag_{w_n}\}$.
  3) Based on MSK_i and $\mathcal{TK}$, generate authorization information Δ_{id} that allows the users affiliated with the tokens from $\mathcal{TK}$ to search the index.
  4) Return $index_{id} = (id, TAG_{id}, \Delta_{id})$.
- $\mathsf{Enc}(FK_{id}, w)$: It generates a tag Tag_w for the keyword w based on the secret key FK_{id}.
- $\mathsf{Trapdoor}(MSK_i, w)$: Run by U_i, this algorithm outputs a trapdoor $Trap_w$ for keyword w.
- $\mathsf{Match}(Trap_w, index_{id})$: Let $index_{id} = (id, TAG_{id}, \Delta_{id})$ and $TAG_{id} = \{Tag_{w_1}, Tag_{w_2}, \cdots, Tag_{w_n}\}$. Run by the cloud server, when $Trap_w$ issuer's token has been included in generating Δ_{id}, it returns 1 if $w = w_i$ form some $1 \leq i \leq n$ and 0 otherwise.

Remark 3: Even though U_i needs to authorize herself to search her own data, for both security and efficiency reasons, we assume that she will not explicitly issue a token $TK_{i\to i}$ and stores it locally. Instead, she should authorize herself on the fly in the $\mathsf{Formindex}$ algorithm.

B. Soundness Property

Given $(MPK_i, MSK_i) = \mathsf{KeyGen}(params)$ and $(MPK_j, MSK_j) = \mathsf{KeyGen}(params)$, a MPSE scheme is sound if the followings are true.

- If $w \in doc$, $\mathsf{Match}(\mathsf{Trapdoor}(MSK_j, w), \mathsf{Formindex}(MSK_i, doc, \mathcal{TK})) = 1$ holds with overwhelming probability when $TK_{j\to i}$ is in $\mathcal{TK}$ or MPK_j is in $\mathcal{TK}$ (in which case $j = i$).
- If $w \notin doc$, $\mathsf{Match}(\mathsf{Trapdoor}(MSK_j, w), \mathsf{Formindex}(MSK_i, doc, \mathcal{TK})) = 0$ holds with overwhelming probability when $TK_{j\to i}$ is in $\mathcal{TK}$ or MPK_j is in $\mathcal{TK}$ (in which case $j = i$).

IV. Security Model for MPSE

In the context of searchable encryption, it is realistic to assume that the de facto size of the keyword space $\mathcal{W}$ is polynomial with respect to the security parameter λ, even though we can specify a space of any size in our theoretical cryptographic scheme. This implies that a scheme can be in the danger of dictionary attacks, as shown in Section I-B2. We further make the following assumptions.

- We assume that the cloud server will honestly perform search on every user's behalf (e.g. if an index is matched by a search query, the cloud server will not say no.), but it can be malicious in trying to obtaining user's private information by colluding with dishonest users or simply forging new users in the system.
- We assume that there is a secure channel between every user and the cloud server to transmit trapdoors. This leads us to consider the cloud server as the primary attacker since a dishonest user alone will not learn anything due to the secure channel.

In the context of MPSE, it is infeasible for a user to be absolutely confident about which other users will be honest in the long run (or, will not be compromised in the future). As such, we choose to investigate the security properties for MPSE in two scenarios.

- In the worst-case scenario, an honest user assumes all other users may be dishonest or compromised

by the cloud server. Corrspondingly, we define the worst-case data and trapdoor privacy properties in Section IV-A. In this scenario, a user who only follows other users has no privacy (e.g. User 3, User 4 and User 6 in the example shown in Fig. 1 of Section I-A).
- In the average-case scenario, an honest user assumes all other users who have authorized her to search are honest. For example, in the example shown in Fig. 1 of Section I-A, User 3 needs to assume User 1 and User 2 are honest. In this scenario, we define the average-case hybrid privacy property in Section IV-B.

It is worth noting that the security properties in the two scenarios do not imply each other, so that it is essential for a MPSE scheme to achieve them all. In the following, we define the security properties through the standard challenger-attacker attack games.

A. Security in the Worst-case Scenario

Without loss of generality, we present the definitions by assuming U_1 is honest. In the attack games, the challenger simulates U_1 with key pair (MPK_1, MSK_1) and interacts with the attacker $\mathcal{A}$, which plays the role of the cloud server and all colluded users U_i ($2 \leq i \leq N$). For simplicity, we assume the attacker generates (MPK_i, MSK_i) for $2 \leq i \leq N$. The attacker may issue the following types of oracle queries to the challenger: Follow, Formindex, Trapdoor.

- For a Follow oracle query, the attacker has MPK_i for some $2 \leq i \leq N$ as input, and receives a token $TK_{1\to i} = \mathsf{Follow}(MSK_1, MPK_i)$.
- For a Formindex oracle query, the attacker has a document parameter *doc* and a set of tokens $\mathcal{TK}$ as input, and receives an index $index_{id} = \mathsf{Formindex}(MSK_1, doc, \mathcal{TK})$. If the attacker wants U_1 to search her own index, it includes MPK_1 in $\mathcal{TK}$. Note that $\mathcal{TK}$ mainly contains tokens of the form $TK_{i\to 1}$.
- For a Tarpdoor oracle query, the attacker has a keyword w as input, and receives a trapdoor $Trap_w = \mathsf{Trapdoor}(MSK_1, w)$.

Remark 4: In answering the oracle queries, the challenger should check the inputs from the attacker are well-formed. For instance, in a Follow oracle query the MPK_i should be in the right form, and in a Formindex oracle query the parameter *doc* contain keywords from the right domain. For the sake of simplicity, we skip enumerating all these checks.

Intuitively, the worst-case data privacy property says that, if U_1 has not authorized an index to any other user (e.g. Index1 for Doc 1 in the toy example, shown in Fig. 1 of Section I-A), then the attacker will not learned any useful information about the keywords in this index unless U_1 reveals it by issuing the relevant trapdoors. This property is an analog to the semantic security for PEKS in [6].

Definition 1: A MPSE scheme achieves worst-case data privacy if any P.P.T. attacker $\mathcal{A}$'s advantage (i.e. $|Pr[b' = b] - \frac{1}{2}|$) is negligible in the attack game shown in Fig. 2. In the game, there are two restrictions.

- In the Formindex oracle query to generate $index_{id^*}$, it is required that $\mathcal{TK}$ only contains MPK_1 and neither w_0 nor w_1 is in the *doc* parameter.
- Neither w_0 nor w_1 has been queried to the Trapdoor oracle.

1) $params \xleftarrow{\$} \mathsf{Setup}(\lambda)$
   $(MPK_1, MSK_1) \xleftarrow{\$} \mathsf{KeyGen}(params)$
2) $(w_0, w_1, index_{id^*}, state) \xleftarrow{\$} \mathcal{A}^{\mathsf{Follow,Formindex,Trapdoor}}(params, MPK_1)$
3) $b \in_R \{0,1\}$
   $Tag_{w_b} = \mathsf{Enc}(FK_{id^*}, w_b),\ \ Tag_{w_{\bar{b}}} = \mathsf{Enc}(FK_{id^*}, w_{\bar{b}})$
4) $b' \xleftarrow{\$} \mathcal{A}^{\mathsf{Follow,Formindex,Trapdoor}}(state, Tag_{w_b}, Tag_{w_{\bar{b}}})$

Fig. 2. Worst-case Data Privacy Game

When generating the challenge $(Tag_{w_b}, Tag_{w_{\bar{b}}})$ in Step 3, FK_{id^*} is the secret key the challenger has used in generating $index_{id^*}$. In the above definition, the first restriction is to guarantee that no keyword should repeat in the index and the index has not been authorized to the attacker. Note that the challenger will return Tag_{w_b} and $Tag_{w_{\bar{b}}}$ generated under FK_{id^*}, if the attacker queries Formindex with w_0 or w_1 then the keyword will repeat in the index. The second restriction is obviously essential. Given $Trap_{w_0}$ or $Trap_{w_1}$ (the attacker knows the corresponding keyword w_0 or w_1), the attacker can trivially recover b by running the Match algorithm.

Remark 5: In the definition, we have set the challenge to be $(Tag_{w_b}, Tag_{w_{\bar{b}}})$, which equals to (Tag_{w_0}, Tag_{w_1}) if $b = 0$ and equals to (Tag_{w_1}, Tag_{w_0}) otherwise. The attacker's task is to tell b, namely which tag is for w_0 or w_1. This is in contrast to the usual definitions, e.g. those from [6], [22], where the challenge only contains Tag_{w_b}. For searchable encryption schemes in the asymmetric setting, e.g. [6], this feature does not make much difference because the attacker can generate tags on its own, but it seems to result in a strictly stronger definition for symmetric searchable encryption schemes (including MPSE) because $(Tag_{w_b}, Tag_{w_{\bar{b}}})$ reveals more information than Tag_{w_b} alone to the attacker.

Intuitively, the worst-case trapdoor privacy property says that, if U_1 does not follow any other user, then the attacker will not learn any useful information about the keywords in the U_1's trapdoors unless these trapdoors match an index (e.g. Index 3 or Index4 in the toy example, shown in Fig. 1 of Section I-A) that has been shared with the attacker.

Definition 2: A MPSE scheme achieves worst-case trapdoor privacy if any P.P.T. attacker

$\mathcal{A}$'s advantage (i.e. $|Pr[b' = b] - \frac{1}{2}|$) is negligible in the attack game shown in Fig. 3. In the game, there are three restrictions.

- For any Formindex oracle query with the input $(doc, \mathcal{TK})$, the following two situations should not happen at the same time: (1) w_0 or w_1 is in the doc parameter; (2) $\mathcal{TK}$ contains $TK_{i\to 1}$ for some i.
- For any Formindex oracle query with the input $(doc, \mathcal{TK})$, if $\mathcal{TK}$ contains only MPK_1 then w_0 and w_1 can only be in the doc parameter at the same time.
- Neither w_0 nor w_1 has been queried to the Trapdoor oracle.

1) $params \xleftarrow{\$} \mathsf{Setup}(\lambda)$
   $(MPK_1, MSK_1) \xleftarrow{\$} \mathsf{KeyGen}(params)$
2) $(w_0, w_1, state) \xleftarrow{\$} \mathcal{A}^{\mathsf{Formindex},\mathsf{Trapdoor}}(params, MPK_1)$
3) $b \in_R \{0,1\}$, $Trap_{w_b} = \mathsf{Trapdoor}(MSK_1, w_b)$
   $Trap_{w_{\bar{b}}} = \mathsf{Trapdoor}(MSK_1, w_{\bar{b}})$
4) $b' \xleftarrow{\$} \mathcal{A}^{\mathsf{Formindex},\mathsf{Trapdoor}}(state, Trap_{w_b}, Trap_{w_{\bar{b}}})$

Fig. 3. Worst-case Trapdoor Privacy Game

In the above definition, the first restriction says that an index should not simultaneously be authorized to the attacker and contain the keyword w_0 or w_1. Otherwise, the attacker can recover b by comparing search results of $Trap_{w_b}$ to the results of its own trapdoors for w_0 and w_1 (this is possible because the attacker is authorized to search). The second restriction is essential. If only w_0 or w_1 is in an index (the attacker knows which one), then the attacker can recover b in $Trap_{w_b}$ by running the Match algorithm on $Trap_{w_b}$ and the index. The third restriction is essential, because, given $Trap_{w_0}$ or $Trap_{w_1}$ (the attacker knows the corresponding keyword w_0 or w_1), the attacker can trivially recover b in $Trap_{w_b}$ by running the Match algorithm based on these trapdoors and comparing their results.

Remark 6: It is worth noting that we allow the attacker to access tags for w_0 and w_1 in the game through the Formindex oracle. This is in the vein of *enhanced* function privacy for IBE and enhanced keyword privacy for PEKS [7]. In contrast to Definition 1, the attacker has no access to the Follow oracle in Definition 2. As a result, it seems difficult to have a hybrid definition to cover both properties.

B. Security in the Average-case Scenario

Unlike in the worst-case scenario, where the challenger simulates only one honest user U_1, in the average-case scenario the challenger simulates a group of honest users who do not follow any user from outside of the group. For example, User 1, User 2, and User 3 form such a group, as shown in Fig. 1 of Section I-A. Without loss of generality, we assume that U_t $(1 \le j \le t)$ are honest and t is an integer determined by the attacker. The attacker may query the following oracles: Follow, Formindex, Trapdoor.

- For a Follow oracle query, the attacker has x, y where $1 \le x \ne y \le t$ as input and receives $TK_{x\to y} = \mathsf{Follow}(MSK_x, MPK_y)$.
- For a Formindex oracle query, the attacker has MPK_x where $1 \le x \le t$, a document parameter doc and a set of tokens $\mathcal{TK}$ as input, and receives an index $index_{id} = \mathsf{Formindex}(MSK_x, doc, \mathcal{TK})$.
- For a Tarpdoor oracle query, the attacker has x where $1 \le x \le t$ and a keyword w as input, and receives a trapdoor $Trap_w = \mathsf{Trapdoor}(MSK_x, w)$.

Intuitively, the average-case hybrid privacy property implies that the attacker cannot learn more information than what can be inferred from the search results on their indexes. The following definition can be considered as a hybrid version of the worst-case data and trapdoor privacy properties, as we ask the attacker to distinguish two (tag, trapdoor) pairs for (v_0, w_0) and (v_1, w_1) respectively, which are chosen by the attacker.

Definition 3: A MPSE scheme achieves average-case hybrid privacy if any P.P.T. attacker $\mathcal{A}$'s advantage (i.e. $|Pr[b' = b] - \frac{1}{2}|$) is negligible in the attack game shown in Fig. 4. Let the set $\mathcal{S} = \{MPK_1, \cdots, MPK_t, TK_{x\to y}\ (1 \le x \ne y \le t)\}$. The set $\mathcal{TAG}_{v_b}$ contains $\mathsf{Enc}(FK_{id}, v_b)$ for every FK_{id} that is generated in answering the Formindex oracle query with an input $(MPK_x, doc, \mathcal{TK})$ where $\mathcal{TK} \subseteq \mathcal{S}$ (i.e. the index has not been authorized to the attacker). The set $\mathcal{TRAP}_{w_b}$ contains $\mathsf{Trapdoor}(MSK_x, w_b)$ for every $1 \le x \le t$. $\mathcal{TAG}_{v_{\bar{b}}}$ and $\mathcal{TRAP}_{w_{\bar{b}}}$ are defined similarly. There are the following restrictions.

- $v_0 = w_0$ iff $v_1 = w_1$, and $v_0 = w_1$ iff $v_1 = w_0$.
- Any of v_0, w_0, v_1, w_1 should not be in the input to the Trapdoor oracle.
- For any Formindex oracle query with the input $(MPK_x, doc, \mathcal{TK})$, if $\mathcal{TK} \cap \mathcal{S} \ne \emptyset$ and $\mathcal{TK} \setminus \mathcal{S} \ne \emptyset$ (the index has been authorized to both the attacker and some honest users), then none of v_0, w_0, v_1, w_1 is included in the doc parameter.
- For any Formindex oracle query with the input $(MPK_x, doc, \mathcal{TK})$, if $\mathcal{TK} \subseteq \mathcal{S}$ (i.e. the index has not been authorized to the attacker), then none of v_0, w_0, v_1, w_1 should be in the doc parameter.

1) $params \xleftarrow{\$} \mathsf{Setup}(\lambda)$
2) $(t, state) \xleftarrow{\$} \mathcal{A}(params)$
3) $(MPK_1, MSK_1, \cdots, MPK_t, MSK_t) \xleftarrow{\$} \mathsf{KeyGen}(params)$
   $\mathcal{MPK}^* = (MPK_1, \cdots, MPK_t)$
4) $(v_0, w_0, v_1, w_1, state) \xleftarrow{\$}$
   $\mathcal{A}^{\mathsf{Follow},\mathsf{Formindex},\mathsf{Trapdoor}}(state, \mathcal{MPK}^*)$
5) $b \in_R \{0,1\}$, $\mathcal{TAG}_{v_b}$, $\mathcal{TRAP}_{w_b}$, $\mathcal{TAG}_{v_{\bar{b}}}$, $\mathcal{TRAP}_{w_{\bar{b}}}$
6) $b' \xleftarrow{\$} \mathcal{A}^{\mathsf{Follow},\mathsf{Formindex},\mathsf{Trapdoor}}(state, \mathcal{TAG}_{v_b}, \mathcal{TRAP}_{w_b}, \mathcal{TAG}_{v_{\bar{b}}}, \mathcal{TRAP}_{w_{\bar{b}}})$

Fig. 4. Average-case Hybrid Privacy Game

The first and second restrictions are essential because otherwise the attacker can trivially recover b by matching

the trapdoors and the tags in the challenge. The third restriction is necessary since it means that if an index is authorized to the attacker then none of v_0, w_0, v_1, w_1 should be included in it. The fourth restriction is to guarantee that no keyword will repeat in any index.

Consider the example shown in Fig. 1 of Section I-A, if the attacker chooses $t = 3$ in the attack (i.e. User 1, User 2, and User 3 are not compromised) then $\mathcal{TAG}_{v_b}$, $\mathcal{TRAP}_{w_b}$, $\mathcal{TAG}_{v_{\bar{b}}}$, $\mathcal{TRAP}_{w_{\bar{b}}}$ are defined as follows. $\mathcal{TAG}_{v_b}$ and $\mathcal{TAG}_{v_{\bar{b}}}$ contain the tags for v_b and $v_{\bar{b}}$ under the document-specific secret keys of index1, index2, index3 and index5. $\mathcal{TRAP}_{w_b}$ and $\mathcal{TRAP}_{w_{\bar{b}}}$ contain trapdoors for w_b and $w_{\bar{b}}$ under the long-term private keys of User 1, User 2 and User 3.

C. Remarks on the Security Model

In [22], Popa and Zeldovich defined two security properties for their multi-key searchable encryption scheme.

- One is data hiding property. In the attack game, the challenger generates the key materials after the attacker has chosen its parameters. This property is similar to worst-case data privacy property for MPSE, but seems to be weaker (see Definition 1 and Remark 5 in Section IV-A).
- The other is trapdoor hiding property (called token hiding property in [22]), which is similar to the worst-case trapdoor privacy property (Definition 2 in Section III) for MPSE. The trapdoor hiding property definition is weaker because it does not allow the attacker to query any trapdoor from the honest user. In contrast, in our worst-case trapdoor privacy property definition, we allow the attacker to query any trapdoors except for the two challenged keywords w_0, w_1. This explains why the Popa-Zeldovich scheme does not achieve our worst-case trapdoor privacy property.

V. Main MPSE Construction

In the literature, searchable encryption schemes in the symmetric setting are usually based on lightweight cryptographic primitives such as block ciphers and pseudo-random functions. As a result, they are much more efficient than schemes in the asymmetric setting where relatively expensive tools (e.g. bilinear pairings) are required. Unfortunately, for MPSE, it seems difficult to have an instantiation only based on these lightweight primitives due to the fact that a user needs to selectively authorize other users to search her indexes. This fact has lead Popa and Zeldovich to design their scheme based on bilinear pairings [22], by employing the bilinear property of pairings.

In our construction, we follow a very similar approach to that of Popa and Zeldovich, by adapting the PEKS construction of [6] to the symmetric setting. However, our construction is different in two aspects.

- One is that an index contains separate searchable contents for the owner and other authorized users, so that we can avoid the undesirable property of the Popa-Zeldovich scheme.
- The other is that we add parameters for users to use a static variant of ElGamal encryption to protect their credential (i.e. $g_2^{\frac{1}{y_i}}$ for user U_i) in running the Follow algorithm. As a side effect, we get rid of the eXternal Diffie-Hellman Variant (XDHV) assumption required in [22]. Note that other encryption can also be used, but they will be less efficient in space and size.

A. The Proposed MPSE Scheme

Let λ be the security parameter, the proposed MPSE scheme consists of the following algorithms.

- $\mathsf{Setup}(\lambda)$: run $\mathcal{G}_{\mathcal{P}}$ to generate $\Gamma = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, , g_1, g_2)$, and return $params = (\Gamma, \mathsf{H}, \mathsf{H}_1)$, where $\mathsf{H} : \{0,1\}^* \rightarrow \mathbb{G}_1$ and $\mathsf{H}_1 : \{0,1\}^* \rightarrow \mathbb{G}_2$ are two hash functions.
- $\mathsf{KeyGen}(params)$: Run by U_i, for every $1 \leq i \leq N$, it returns (MPK_i, MSK_i), where $MPK_i = g_2^{x_i}$ and $MSK_i = (MPK_i, x_i, y_i)$ for $x_i, y_i \in_R \mathbb{Z}_p$.
- $\mathsf{Follow}(MSK_i, MPK_j)$: parse MSK_i as (MPK_i, x_i, y_i) and MPK_j as $g_2^{x_j}$, and return $TK_{i \rightarrow j} = (MPK_i, MPK_j, \mathsf{H}_1(g_2^{x_i x_j} \| i \| j) \cdot g_2^{\frac{1}{y_i}})$.
- $\mathsf{Formindex}(MSK_i, doc, \mathcal{TK})$: parse MSK_i as (MPK_i, x_i, y_i) and doc as $(w_1, w_2, \cdots, w_n)$, and then do the following.
  1) Select a unique identifier $id \in_R \{0,1\}^\lambda$ for the index, and generate a document-specific secret key $FK_{id} = (k_1, k_2)$ where $k_1, k_2 \in_R \mathbb{Z}_p$.
  2) Generate $TAG_{id} = \{\mathsf{Enc}(FK_{id}, w_1), \cdots, \mathsf{Enc}(FK_{id}, w_n)\}_{\mathsf{Permute}}$, which is a randomly permutated set of the ciphertexts.
  3) For every $TK_{t \rightarrow i} = (MPK_t, MPK_i, \mathsf{H}_1(g_2^{x_t x_i} \| t \| i) \cdot g_2^{\frac{1}{y_t}})$ from $\mathcal{TK}$, first recover $g_2^{\frac{1}{y_t}}$ and then generate $\Theta_{MPK_t} = g_2^{\frac{k_2}{y_t}}$.
  4) If $MPK_i \in \mathcal{TK}$, generate $\Theta_{MPK_i} = g_2^{\frac{k_1}{y_i}}$ and set $\Delta_{id} = (\Theta_{MPK_t}, \cdots; \Theta_{MPK_i})$, otherwise set $\Delta_{id} = (\Theta_{MPK_t}, \cdots)$.
  5) Return $index_{id} = (id, TAG_{id}, \Delta_{id})$.
- $\mathsf{Enc}(FK_{id}, w)$: parse FK_{id} as k_1, k_2 and return $c = (c_1, c_2)$, where $c_1 = \hat{e}(\mathsf{H}(w), g_2)^{k_1}, c_2 = \hat{e}(\mathsf{H}(w), g_2)^{k_2}$.
- $\mathsf{Trapdoor}(MSK_i, w)$: parse MSK_i as (MPK_i, x_i, y_i), return $Trap_w = (MPK_i, \mathsf{H}(w)^{y_i})$.
- $\mathsf{Match}(Trap_w, index_{id})$: parse $Trap_w$ as (α, β) and parse $index_{id}$ as $(id, \{\mathsf{Enc}(FK_{id}, w_1), \mathsf{Enc}(FK_{id}, w_2), \cdots, \mathsf{Enc}(FK_{id}, w_n)\}_{\mathsf{Permute}}, \Delta_{id})$ and proceed as follows:
  1) If $\Theta_\alpha \notin \Delta_{id}$, return 0.
  2) If id is generated by the issuer of $Trap_w$, then if $\mathsf{Test}_1(\mathsf{Enc}(FK_{id}, w_j), \beta) = 1$ for some $1 \leq j \leq n$ return 1 otherwise return 0. Let $\mathsf{Enc}(FK_{id}, w_j) =$

(c_1, c_2), we define $\mathsf{Test}_1(\mathsf{Enc}(FK_{id}, w_j), \beta) = 1$ iff $c_1 = \hat{e}(\beta, \Theta_\alpha)$.

3) If $\mathsf{Test}_2(\mathsf{Enc}(FK_{id}, w_j), \beta) = 1$ for some $1 \le j \le n$ return 1, otherwise return 0. Let $\mathsf{Enc}(FK_{id}, w_j) = (c_1, c_2)$, we define $\mathsf{Test}_2(\mathsf{Enc}(FK_{id}, w_j), \beta) = 1$ iff $c_2 = \hat{e}(\beta, \Theta_\alpha)$.

B. Analysis of the Proposed Scheme

Let $MPK_i = g_2^{x_i}$, $MSK_i = (MPK_i, x_i, y_i)$ and $MPK_j = g_2^{x_j}$, $MSK_j = (MPK_j, x_j, y_j)$. Given any $FK_{id} = (k_1, k_2)$ and $w, w' \in \mathcal{W}$, it is straightforward to verify that the followings are true.

- Given $(c_1, c_2) = \mathsf{Enc}(FK_{id}, w)$, the equalities $c_1 = \hat{e}(\mathsf{H}(w)^{y_i}, g_2^{\frac{k_1}{y_i}})$ and $c_2 = \hat{e}(\mathsf{H}(w)^{y_j}, g_2^{\frac{k_2}{y_j}})$ hold with probability 1.
- Given $(c_1, c_2) = \mathsf{Enc}(FK_{id}, w)$ and $w' \neq w$, the inequalities $c_1 \neq \hat{e}(\mathsf{H}(w')^{y_i}, g_2^{\frac{k_1}{y_i}})$ and $c_2 \neq \hat{e}(\mathsf{H}(w')^{y_j}, g_2^{\frac{k_2}{y_j}})$ hold with overwhelming probability. In fact, these inequalities do not hold only if $\mathsf{H}(w') = \mathsf{H}(w)$ which has a negligible probability.

As a result, the value of $\mathsf{Match}(\mathsf{Trapdoor}(MSK_j, w), \mathsf{Formindex}(MSK_i, doc, \mathcal{TK}))$ is the same as required in the soundness definition in Section III-B. The proposed scheme is sound.

With the following three theorems, we show that the proposed scheme achieves all three properties defined in our security model, defined in Section IV. The proofs make use of the standard game-hopping technique.

Theorem 1: The proposed MPSE scheme achieves worst-case data privacy property under Definition 1, based on the BDHV and SXDH assumptions in the random oracle model.

Proof. This proof is very straightforward. The challenger makes use of the property of random oracle, and tries to guess the attacker's output in step 2 of the attack game. This strategy reduces the tightness of the reduction, but makes the proof easier.

Suppose that the number of queries to the $\mathsf{Formindex}$ oracle is bounded by q_1 and the number of queries to the random oracle H is bounded by q_2. For a P.P.T. attacker, q_1 and q_2 should be polynomials in the security parameter λ.

Game 0: The challenger faithfully simulates everything. Let the attacker's advantage be ϵ.

Game 1: The challenger tries to guess id^*, w_0 and w_1 in the attacker's output in Step 2 of the attack game. If the guess is correct, the challenger continues, and aborts otherwise. The challenger's success probability is $\frac{1}{q_1 q_2 (q_2 - 1)}$. Let the attacker's advantage be ϵ_1, which equals to $\frac{\epsilon}{q_1 q_2 (q_2 - 1)}$.

Game 2: The challenger performs the same as in Game 1, except for the following. For a query to H with the input w where $w \notin \{w_0, w_1\}$, the challenger first check whether the same query has been made. If so, return the existing hash value, otherwise chooses $r \in_R \mathbb{Z}_p$ and return g_1^r as the hash value. The challenger also send the exponent r to the attacker. If w_0 or w_1 is queried to H, then the challenger randomly choose a value from $\mathbb{G}_1$ as the hash value. This game is identical to Game 1. Let the attacker's advantage be ϵ_2 in this game, and we have $\epsilon_2 = \epsilon_1$.

Game 3: The challenger performs the same as in Game 2, except for the following. Let $MPK_1 = g_2^{x^*}$, $MSK_1 = (MPK_1, x^*, y^*)$, $FK_{id^*} = (k_1^*, k_2^*)$ and $k_2^* = k_1^* z^*$ for $z^* \in_R \mathbb{Z}_p$. The challenger gives x^*, $g_1^{y^*}$, $g_2^{\frac{1}{y^*}}$, $g_2^{\frac{k_1^*}{y^*}}$ and z^* to the attacker. With the given information, the attacker can answer $\mathsf{Formindex}$, Follow and $\mathsf{Trapdoor}$ oracles on its own. Let the attacker's advantage be ϵ_3 in this game, and it is clear $\epsilon_3 \ge \epsilon_2$ because of the additional information given to the attacker.

In Game 3, the attacker is given the following information and is asked to guess b.

$$(g_1^{y^*}, g_2^{\frac{1}{y^*}}, g_2^{\frac{k_1^*}{y^*}}, \mathsf{H}(w_0), \mathsf{H}(w_1), \hat{e}(\mathsf{H}(w_b), g_2)^{k_1^*}, \hat{e}(\mathsf{H}(w_{\bar{b}}), g_2)^{k_1^*})$$

$\hat{e}(\mathsf{H}(w_b), g_2)^{k_1^*}$ and $\hat{e}(\mathsf{H}(w_{\bar{b}}), g_2)^{k_1^*}$ are from the challenge, and we have ignored $(\hat{e}(\mathsf{H}(w_b), g_2)^{k_1^*})^{z^*}$ and $(\hat{e}(\mathsf{H}(w_{\bar{b}}), g_2)^{k_1^*})^{z^*}$ since they are redundant. It is worth mentioning that we deliberately ignore other redundant information such as x^* because they are not relevant to the rest of our proof.

Game 4: The challenger performs the same as in Game 3, except for randomly setting the challenge. The attacker will obtain $(g_1^{y^*}, g_2^{\frac{1}{y^*}}, g_2^{\frac{k_1^*}{y^*}}, \mathsf{H}(w_0), \mathsf{H}(w_1), R_0, R_1)$, where $R_0, R_1 \in_R \mathbb{G}_T$. Let the attacker's advantage be ϵ_4 in this game, and we have $|\epsilon_4 - \epsilon_3| \le \epsilon_{bdhv} + \epsilon_{sxdh}$ based on Lemma 1.

In Game 4, it is clear that $\epsilon_4 = 0$ because the challenge is independent from b. To sum up, we have $\frac{\epsilon}{q_1 q_2 (q_2 - 1)} \le \epsilon_{bdhv} + \epsilon_{sxdh}$. Since $q_1 q_2 (q_2 - 1)$ is a polynomial in the security parameter, therefore, if $\epsilon_{bdhv} + \epsilon_{sxdh}$ is negligible then ϵ is also negligible. The theorem now follows. ∎

Theorem 2: The proposed MPSE scheme achieves worst-case trapdoor privacy property under Definition 2, based on the SXDH assumption in the random oracle model.

Proof. This proof makes use the same strategy as in the proof of Theorem 1.

Suppose that the number of queries to the random oracle H is bounded by q_2. For a P.P.T. attacker, q_2 should be a polynomial in the security parameter λ.

Game 0: The challenger faithfully simulates everything. Let the attacker's advantage be ϵ.

Game 1: The challenger tries to guess w_0 and w_1 in the attacker's output in Step 2 of the attack game. If the guess is correct, the challenger continues, and aborts otherwise. The challenger's success probability is $\frac{1}{q_2 (q_2 - 1)}$. Let the attacker's advantage be ϵ_1, which equals to $\frac{\epsilon}{q_2 (q_2 - 1)}$.

Game 2: The challenger performs the same as in Game 1, except for the following. For a query to H with the

input w where $w \notin \{w_0, w_1\}$, the challenger first check whether the same query has been made. If so, return the existing hash value, otherwise chooses $r \in_R \mathbb{Z}_p$ and return g_1^r as the hash value. The challenger also send the exponent r to the attacker. If w_0 or w_1 is queried to H, then the challenger randomly choose a value from $\mathbb{G}_1$ as the hash value. This game is identical to Game 1. Let the attacker's advantage be ϵ_2 in this game, and we have $\epsilon_2 = \epsilon_1$.

Game 3: The challenger performs the same as in Game 2, except for the following. Let $MPK_1 = g_2^{x^*}$ and $MSK_1 = (MPK_1, x^*, y^*)$. The challenger gives x^* and $g_1^{y^*}$ to the attacker. With the given information, the attacker can answer the Trapdoor oracle on its own. Let the attacker's advantage be ϵ_3 in this game, and it is clear $\epsilon_3 \geq \epsilon_2$ because of the additional information given to the attacker.

Game 4: The challenger performs the same as in Game 3, except for the following. The challenger selects $k_1^* \in_R \mathbb{Z}_p$ to compute $g_2^{\frac{k_1^*}{y^*}}$, and also computes $\mathsf{H}(w_b)^{y^*}, \mathsf{H}(w_{\bar{b}})^{y^*}$. The challenger then answers any Formindex oracle query with the input $(doc, \mathcal{TK})$ as follows.

- If $MPK_1 \notin \mathcal{TK}$, reject the query and let the attacker answer on its own.
- If $MPK_1 \in \mathcal{TK}$ and $\mathcal{TK} \setminus \{MPK_1\} \neq \emptyset$, do the following. In this case, $w_0, w_1 \notin doc$.
  1) Select a unique identifier $id \in_R \{0,1\}^\lambda$ for the index.
  2) Select $k_2, z \in_R \mathbb{Z}_p$.
  3) For every $w \in doc$, suppose $\mathsf{H}(w) = g_1^r$ then set its ciphertext $(\hat{e}(g_1^{y^*}, g_2^{\frac{k_1^*}{y^*}})^{rz}, \hat{e}(g_1^r, g_2)^{k_2})$. Shuffle all the ciphertexts to generate TAG_{id}.
  4) Set $\Delta_{MPK_1} = g_2^{\frac{k_1^* z}{y^*}}$, use k_2 to generate Δ_{MPK_t} for all $TK_{t\to 1} \in \mathcal{TK}$, and then set Δ_{id}.
  5) Return $(id, TAG_{id}, \Delta_{id})$.
- If $\mathcal{TK} = \{MPK_1\}$, do the following.
  1) Select a unique identifier $id \in_R \{0,1\}^\lambda$ for the index.
  2) Select $k_2, z \in_R \mathbb{Z}_p$.
  3) If $w_0, w_1 \in doc$, set their ciphertexts to be $(\hat{e}(\mathsf{H}(w_b)^{y^*}, g_2^{\frac{k_1^*}{y^*}})^z, \hat{e}(\mathsf{H}(w_b)^{y^*}, g_2^{\frac{k_1^*}{y^*}})^{zk_2})$ and $(\hat{e}(\mathsf{H}(w_{\bar{b}})^{y^*}, g_2^{\frac{k_1^*}{y^*}})^z, \hat{e}(\mathsf{H}(w_{\bar{b}})^{y^*}, g_2^{\frac{k_1^*}{y^*}})^{zk_2})$. For every $w \in doc$ and $w \notin \{w_0, w_1\}$, suppose that $\mathsf{H}(w) = g_1^r$ then set its ciphertext $(\hat{e}(g_1^{y^*}, g_2^{\frac{k_1^*}{y^*}})^{rz}, \hat{e}(g_1^{y^*}, g_2^{\frac{k_1^*}{y^*}})^{rzk_2})$. Shuffle all the ciphertexts to generate TAG_{id}.
  4) Set $\Delta_{MPK_1} = g_2^{\frac{k_1^* z}{y^*}}$ and then set $\Delta_{id} = \{\Delta_{MPK_1}\}$.
  5) Return $(id, TAG_{id}, \Delta_{id})$.

This game is indeed identical to Game 3. Let the attacker's advantage be ϵ_4 in this game, and we have $\epsilon_4 = \epsilon_3$.

By looking at the simulation, it is clear that the challenger only needs $(g_1^{y^*}, g_2^{\frac{k_1^*}{y^*}}, \mathsf{H}(w_0), \mathsf{H}(w_1), \mathsf{H}(w_b)^{y^*}, \mathsf{H}(w_{\bar{b}})^{y^*})$ to faithfully answer all the oracle queries from the attacker. In particular, the challenger does not need to know k_1^* and the value of b. Since the random number k_1^* only appears in $g_2^{\frac{k_1^*}{y^*}}$, the challenger indeed only needs $(g_1^{y^*}, \mathsf{H}(w_0), \mathsf{H}(w_1), \mathsf{H}(w_b)^{y^*}, \mathsf{H}(w_{\bar{b}})^{y^*})$ to faithfully answer all the oracle queries from the attacker in the game.

Game 5: The challenger performs the same as in Game 4, except for replacing $\mathsf{H}(w_b)^{y^*}$ with $R_1 \in_R \mathbb{G}_1$ and replacing $\mathsf{H}(w_{\bar{b}})^{y^*}$ with $R_2 \in_R \mathbb{G}_1$. Let the attacker's advantage be ϵ_5 in this game. It is clear that, in game 4 and Game 5, the attacker can be regarded as a distinguisher for M-DDH$_3$ problem in $\mathbb{G}_1$, and $\epsilon_{m-ddh_3} \leq 9\epsilon_{ddh} \leq 9\epsilon_{sxdh}$. Therefore, we have $|\epsilon_5 - \epsilon_4| \leq 9\epsilon_{sxdh}$.

In Game 5, it is clear that $\epsilon_4 = 0$ because the challenge is independent from b. To sum up, we have $\frac{\epsilon}{q_2(q_2-1)} \leq 9\epsilon_{sxdh}$. Since $q_2(q_2-1)$ is a polynomial in the security parameter, therefore, if ϵ_{sxdh} is negligible then ϵ is also negligible. The theorem now follows. ∎

Theorem 3: The proposed MPSE scheme achieves average-case hybrid privacy property under Definition 3, based on the SXDH assumption in the random oracle model.

Proof. In the attack game, there are five general cases for the values v_0, w_0, v_1, w_1.

1) $v_0 = w_0$, $v_1 = w_1$, and $v_0 \neq v_1$ (or equivalently, $v_0 = w_1$, $v_1 = w_0$ and $v_0 \neq v_1$)
2) $v_0 \neq v_1 \neq w_0 \neq w_1$
3) $v_0 = v_1 = w_0 = w_1$
4) $v_0 = v_1$ and $w_0 \neq w_1 \neq v_0$
5) $w_0 = w_1$ and $v_0 \neq v_1 \neq w_0$

It is clear that the attacker's advantage is 0 in the third case (i.e. $v_0 = v_1 = w_0 = w_1$) and the fourth and fifth cases is covered in the first case.

Suppose that the number of queries to the random oracle H is bounded by q_2. For a P.P.T. attacker, q_2 should be a polynomial in the security parameter λ. Next, we consider the first and second cases only. The proof is carried out in a similar manner to that of Theorem 2, and also makes use the same strategy as in the proof of Theorem 1.

Proof for Case 1: $v_0 = w_0$, $v_1 = w_1$, and $v_0 \neq v_1$

Game 0: The challenger faithfully simulates everything. Let the attacker's advantage be ϵ.

Game 1: The challenger tries to guess v_0, w_0, v_1, w_1 in the attacker's output in Step 4 of the attack game (in fact, the challenger only needs to guess v_0 and v_1). If the guess is correct, the challenger continues, and aborts otherwise. The challenger's success probability is $\frac{1}{q_2(q_2-1)}$. Let the attacker's advantage be ϵ_1, which equals to $\frac{\epsilon}{q_2(q_2-1)}$.

Game 2: The challenger performs the same as in Game 1, except for the following. For a query to H with the input w where $w \notin \{v_0, w_0, v_1, w_1\}$, the challenger first

check whether the same query has been made. If so, return the existing hash value, otherwise chooses $r \in_R \mathbb{Z}_p$ and return g_1^r as the hash value. The challenger also send the exponent r to the attacker. If one of $\{v_0, w_0, v_1, w_1\}$ is queried to H, then the challenger randomly choose a value from $\mathbb{G}_1$ as the hash value. This game is identical to Game 1. Let the attacker's advantage be ϵ_2 in this game, and we have $\epsilon_2 = \epsilon_1$.

Game 3: The challenger performs the same as in Game 2, except for the following. For each Follow oracle with the input (i, j), the challenger returns $TK_{i\to j} = (MPK_i, MPK_j, R)$ where $R \in_R \mathbb{G}_2$. Let the attacker's advantage be ϵ_3 in this game. Game 3 is identical to Game 2 unless the attacker queries the H_1 oracle with $g_2^{x_i x_j} \| i \| j$ for some i, j (referred to as an event EVT). Straightforwardly, the attacker has advantage $\Pr[EVT]$ in solving the M-DDH$_t$ problem in $\mathbb{G}_2$. As a result, $|\epsilon_3 - \epsilon_2| \le \Pr[EVT] \le t^2 \cdot \epsilon_{sxdh}$.

Game 4: The challenger performs the same as in Game 3, except for the following. Let $MPK_1 = g_2^{x_1}$ and $MSK_1 = (MPK_1, x_1, y_1)$. For $2 \le i \le t$, set $MPK_i = g_2^{x_i}$ and $MSK_i = (MPK_i, x_i, y_1 r_i)$ for $r_i \in_R \mathbb{Z}_p$. Set $r_1 = 1$. The challenger gives $g_1^{y_1}$ and r_i $(2 \le i \le t)$ to the attacker. With the given information, the attacker can answer the $\mathsf{Trapdoor}$ oracle on its own. Let the attacker's advantage be ϵ_4 in this game, and it is clear $\epsilon_4 \ge \epsilon_3$ because of the additional information given to the attacker.

Game 5: The challenger performs the same as in Game 4, except for the following. Let $\mathcal{S} = \mathcal{MPK}^* \cup \{TK_{u\to v} (1 \le u \ne v \le t)\}$. The challenger selects $k_1^* \in_R \mathbb{Z}_p$ to compute $g_2^{\frac{k_1^*}{y^*}}$ and also compute $\mathsf{H}(w_b)^{y^*}, \mathsf{H}(w_{\bar{b}})^{y^*}$ for $b \in_R \{0,1\}$. In the following, we implicitly set $y_1 = y^*$.

- The challenger answers any $\mathsf{Formindex}$ oracle query with the input $(MPK_j, doc, \mathcal{TK})$ for some $1 \le j \le t$ as follows.
  1) If $\mathcal{S} \cap \mathcal{TK} = \emptyset$, do the following.
     a) Select a unique identifier $id \in_R \{0,1\}^\lambda$ for the index.
     b) Select $k_2, k_1 \in_R \mathbb{Z}_p$.
     c) For every $w \in doc$, set its ciphertext $(\hat{e}(\mathsf{H}(w), g_2)^{k_1}, \hat{e}(\mathsf{H}(w), g_2)^{k_2})$. Shuffle all the ciphertexts to generate TAG_{id}.
     d) Use k_2 to generate Δ_{MPK_u} for every token $TK_{u\to j} \in \mathcal{TK}$ and then set Δ_{id}.
     e) Return $(id, TAG_{id}, \Delta_{id})$.
  2) If $\mathcal{TK} \cap \mathcal{S} \ne \emptyset$ and $\mathcal{TK} \setminus \mathcal{S} \ne \emptyset$, do the following. In this case, $v_0, w_0, v_1, w_1 \notin doc$.
     a) Select a unique identifier $id \in_R \{0,1\}^\lambda$ for the index.
     b) Select $k_2, z \in_R \mathbb{Z}_p$.
     c) For every $w \in doc$, suppose $\mathsf{H}(w) = g_1^r$, set its ciphertext $(\hat{e}(g_1^{y^*}, g_2^{\frac{k_1^*}{y^*}})^{rz}, \hat{e}(g_1^r, g_2)^{k_2})$. Shuffle all the ciphertexts to generate TAG_{id}.
     d) For every $MPK_i \in \mathcal{TK}$ and $TK_{i\to j} \in \mathcal{TK}$ for $1 \le i \le t$, set $\Delta_{MPK_i} = g_2^{\frac{k_1^* z}{y^* r_i}}$, use k_2 to generate Δ_{MPK} for every other token $TK_{u\to j} \in \mathcal{TK}$, and then set Δ_{id}.
     e) Return $(id, TAG_{id}, \Delta_{id})$.
  3) If $\mathcal{TK} \subseteq \mathcal{S}$, do the following.
     a) Select a unique identifier $id \in_R \{0,1\}^\lambda$ for the index.
     b) Select $k_2, z \in_R \mathbb{Z}_p$.
     c) For every $w \in doc$, suppose $\mathsf{H}(w) = g_1^r$, set its ciphertext $(\hat{e}(g_1^{y^*}, g_2^{\frac{k_1^*}{y^*}})^{rz}, \hat{e}(g_1^{y^*}, g_2^{\frac{k_1^*}{y^*}})^{rzk_2})$. Shuffle all the ciphertexts to generate TAG_{id}.
     d) For every $MPK_i \in \mathcal{TK}$ and $TK_{i\to j} \in \mathcal{TK}$, set $\Delta_{MPK_i} = g_2^{\frac{k_1^* z}{y^* r_i}}$, and then set Δ_{id}.
     e) Return $(id, TAG_{id}, \Delta_{id})$.
- In generating the challenge, the challenger proceeds as follows.
  - For every k_2, z in the third case of answering the $\mathsf{Formindex}$ oracle, set $\mathsf{Enc}(FK_{id}, v_b) = (\hat{e}(\mathsf{H}(w_b)^{y^*}, g_2^{\frac{k_1^*}{y^*}})^z, \hat{e}(\mathsf{H}(w_b)^{y^*}, g_2^{\frac{k_1^*}{y^*}})^{zk_2})$. Set $\mathcal{TAG}_{v_b}$ as the set of all these ciphertexts.
  - Set $\mathcal{TRAP}_{w_b} = (\mathsf{H}(w_b)^{y^* r_2}, \cdots, \mathsf{H}(w_b)^{y^* r_t})$.
  - For every k_2, z in the third case of answering the $\mathsf{Formindex}$ oracle, set $\mathsf{Enc}(FK_{id}, v_{\bar{b}}) = (\hat{e}(\mathsf{H}(w_{\bar{b}})^{y^*}, g_2^{\frac{k_1^*}{y^*}})^z, \hat{e}(\mathsf{H}(w_{\bar{b}})^{y^*}, g_2^{\frac{k_1^*}{y^*}})^{zk_2})$. Set $\mathcal{TAG}_{v_{\bar{b}}}$ as the set of all these ciphertexts.
  - Set $\mathcal{TRAP}_{w_{\bar{b}}} = (\mathsf{H}(w_{\bar{b}})^{y^* r_2}, \cdots, \mathsf{H}(w_{\bar{b}})^{y^* r_t})$.

This game is indeed identical to Game 4. Let the attacker's advantage be ϵ_5 in this game, and we have $\epsilon_5 = \epsilon_4$.

By looking at the simulation, it is clear that the challenger only needs $(g_1^{y^*}, g_2^{\frac{k_1^*}{y^*}}, \mathsf{H}(w_0), \mathsf{H}(w_1), \mathsf{H}(w_b)^{y^*}, \mathsf{H}(w_{\bar{b}})^{y^*})$ to faithfully answer all the oracle queries from the attacker. Recall that $v_0 = w_0$ and $v_1 = w_1$. In particular, the challenger does not need to know k_1^* and the value of b. Since the random number k_1^* only appears in $g_2^{\frac{k_1^*}{y^*}}$, the challenger indeed only needs $(g_1^{y^*}, \mathsf{H}(w_0), \mathsf{H}(w_1), \mathsf{H}(w_b)^{y^*}, \mathsf{H}(w_{\bar{b}})^{y^*})$ to faithfully answer all the oracle queries from the attacker in the game.

Game 6: The challenger performs the same as in Game 5, except for replacing $\mathsf{H}(w_b)^{y^*}$ with $R_1 \in_R \mathbb{G}_1$ and replacing $\mathsf{H}(w_{\bar{b}})^{y^*}$ with $R_2 \in_R \mathbb{G}_1$. Let the attacker's advantage be ϵ_6 in this game. It is clear that, in game 5 and Game 6, the attacker can be regarded as a distinguisher for the M-DDH$_3$ problem in $\mathbb{G}_1$, and $\epsilon_{m-ddh_3} \le 9\epsilon_{ddh} \le 9\epsilon_{sxdh}$. Therefore, we have $|\epsilon_6 - \epsilon_5| \le 9\epsilon_{sxdh}$.

In Game 6, it is clear that $\epsilon_6 = 0$ because the challenge is independent from b. To sum up, we have $\frac{\epsilon}{q_2(q_2-1)} \le (t^2+9)\epsilon_{sxdh}$. Since $q_2(q_2-1)$ and t are polynomials in the security parameter, therefore, if ϵ_{sxdh} is negligible then ϵ is also negligible.

Proof for Case 2: $v_0 \ne v_1 \ne w_0 \ne w_1$

Game 0: The challenger faithfully simulates everything. Let the attacker's advantage be ϵ.

Game 1: The challenger tries to guess v_0, w_0, v_1, w_1 in the attacker's output in Step 4 of the attack game. If the guess is correct, the challenger continues, and aborts otherwise. The challenger's success probability is $\frac{1}{q_2(q_2-1)(q_2-2)(q_2-3)}$. Let the attacker's advantage be ϵ_1, which equals to $\frac{\epsilon}{q_2(q_2-1)(q_2-2)(q_2-3)}$.

Game 2: The challenger performs the same as in Game 1, except for the following. For a query to H with the input w where $w \notin \{v_0, w_0, v_1, w_1\}$, the challenger first check whether the same query has been made. If so, return the existing hash value, otherwise chooses $r \in_R \mathbb{Z}_p$ and return g_1^r as the hash value. The challenger also send the exponent r to the attacker. If one of $\{v_0, w_0, v_1, w_1\}$ is queried to H, then the challenger randomly choose a value from $\mathbb{G}_1$ as the hash value. This game is identical to Game 1. Let the attacker's advantage be ϵ_2 in this game, and we have $\epsilon_2 = \epsilon_1$.

Game 3: The challenger performs the same as in Game 2, except for the following. For each Follow oracle with the input (i, j), the challenger returns $TK_{i\to j} = (MPK_i, MPK_j, R)$ where $R \in_R \mathbb{G}_2$. Let the attacker's advantage be ϵ_3 in this game. Game 3 is identical to Game 2 unless the attacker queries the H_1 oracle with $g_2^{x_i x_j} \| i \| j$ for some i, j (referred to as an event EVT). Straightforwardly, the attacker has advantage $\Pr[EVT]$ in solving the M-DDH$_t$ problem in $\mathbb{G}_2$. As a result, $|\epsilon_3 - \epsilon_2| \le \Pr[EVT] \le t^2 \cdot \epsilon_{sxdh}$.

Game 4: The challenger performs the same as in Game 3, except for the following. Let $MPK_1 = g_2^{x_1}$ and $MSK_1 = (MPK_1, x_1, y_1)$. For $2 \le i \le t$, set $MPK_i = g_2^{x_i}$ and $MSK_i = (MPK_i, x_i, y_1 r_i)$ for $r_i \in_R \mathbb{Z}_p$. Set $r_1 = 1$. The challenger gives $g_1^{y_1}$ and r_i $(2 \le i \le t)$ to the attacker. With the given information, the attacker can answer the $\mathsf{Trapdoor}$ oracle on its own. Let the attacker's advantage be ϵ_4 in this game, and it is clear $\epsilon_4 \ge \epsilon_3$ because of the additional information given to the attacker.

Game 5: The challenger performs the same as in Game 4, except for the following. Let $\mathcal{S} = \mathcal{MPK}^* \cup \{TK_{u\to v} (1 \le u \ne v \le t)\}$. The challenger selects $k_1^* \in_R \mathbb{Z}_p$ to compute $g_2^{\frac{k_1^*}{y^*}}$ and also compute $\mathsf{H}(v_b)^{y^*}, \mathsf{H}(v_{\bar{b}})^{y^*}, \mathsf{H}(w_d)^{y^*}, \mathsf{H}(w_{\bar{d}})^{y^*}$ for $b, d \in_R \{0,1\}$. In the following, we implicitly set $y_1 = y^*$.

- The challenger answers any $\mathsf{Formindex}$ oracle query with the input $(MPK_j, doc, \mathcal{TK})$ for some $1 \le j \le t$ as follows.
  1) If $\mathcal{S} \cap \mathcal{TK} = \emptyset$, do the following.
     a) Select a unique identifier $id \in_R \{0,1\}^\lambda$ for the index.
     b) Select $k_2, k_1 \in_R \mathbb{Z}_p$.
     c) For every $w \in doc$, set its ciphertext $(\hat{e}(\mathsf{H}(w), g_2)^{k_1}, \hat{e}(\mathsf{H}(w), g_2)^{k_2})$. Shuffle all the ciphertexts to generate TAG_{id}.
     d) Use k_2 to generate Δ_{MPK_u} for every token $TK_{u\to j} \in \mathcal{TK}$ and then set Δ_{id}.
     e) Return $(id, TAG_{id}, \Delta_{id})$.
  2) If $\mathcal{TK} \cap \mathcal{S} \ne \emptyset$ and $\mathcal{TK} \setminus \mathcal{S} \ne \emptyset$, do the following. In this case, $v_0, w_0, v_1, w_1 \notin doc$.
     a) Select a unique identifier $id \in_R \{0,1\}^\lambda$ for the index.
     b) Select $k_2, z \in_R \mathbb{Z}_p$.
     c) For every $w \in doc$, suppose $\mathsf{H}(w) = g_1^r$, set its ciphertext $(\hat{e}(g_1^{y^*}, g_2^{\frac{k_1^*}{y^*}})^{rz}, \hat{e}(g_1^r, g_2)^{k_2})$. Shuffle all the ciphertexts to generate TAG_{id}.
     d) For every $MPK_i \in \mathcal{TK}$ and $TK_{i\to j} \in \mathcal{TK}$ for $1 \le i \le t$, set $\Delta_{MPK_i} = g_2^{\frac{k_1^* z}{y^* r_i}}$, use k_2 to generate Δ_{MPK_u} for every other token $TK_{u\to j} \in \mathcal{TK}$, and then set Δ_{id}.
     e) Return $(id, TAG_{id}, \Delta_{id})$.
  3) If $\mathcal{TK} \subseteq \mathcal{S}$, do the following.
     a) Select a unique identifier $id \in_R \{0,1\}^\lambda$ for the index.
     b) Select $k_2, z \in_R \mathbb{Z}_p$.
     c) For every $w \in doc$, suppose $\mathsf{H}(w) = g_1^r$, set its ciphertext $(\hat{e}(g_1^{y^*}, g_2^{\frac{k_1^*}{y^*}})^{rz}, \hat{e}(g_1^{y^*}, g_2^{\frac{k_1^*}{y^*}})^{rzk_2})$. Shuffle all the ciphertexts to generate TAG_{id}.
     d) For every $MPK_i \in \mathcal{TK}$ and $TK_{MPK_i \to MPK_j} \in \mathcal{TK}$, set $\Delta_{MPK_i} = g_2^{\frac{k_1^* z}{y^* r_i}}$, and then set Δ_{id}.
     e) Return $(id, TAG_{id}, \Delta_{id})$.
- In generating the challenge, the challenger proceeds as follows.
  - For every k_2, z in the third case of answering the $\mathsf{Formindex}$ oracle, set $\mathsf{Enc}(FK_{id}, v_b) = (\hat{e}(\mathsf{H}(v_b)^{y^*}, g_2^{\frac{k_1^*}{y^*}})^z, \hat{e}(\mathsf{H}(v_b)^{y^*}, g_2^{\frac{k_1^*}{y^*}})^{zk_2})$. Set $\mathcal{TAG}_{v_b}$ as the set of all these ciphertexts.
  - Set $\mathcal{TRAP}_{w_d} = (\mathsf{H}(w_d)^{y^* r_2}, \cdots, \mathsf{H}(w_d)^{y^* r_t})$.
  - For every k_2, z in the third case of answering the $\mathsf{Formindex}$ oracle, set $\mathsf{Enc}(FK_{id}, v_{\bar{b}}) = (\hat{e}(\mathsf{H}(v_{\bar{b}})^{y^*}, g_2^{\frac{k_1^*}{y^*}})^z, \hat{e}(\mathsf{H}(v_{\bar{b}})^{y^*}, g_2^{\frac{k_1^*}{y^*}})^{zk_2})$. Set $\mathcal{TAG}_{v_{\bar{b}}}$ as the set of all these ciphertexts.
  - Set $\mathcal{TRAP}_{w_{\bar{d}}} = (\mathsf{H}(w_{\bar{d}})^{y^* r_2}, \cdots, \mathsf{H}(w_{\bar{d}})^{y^* r_t})$.

When $b = d$, this game is indeed identical to Game 4. Let the attacker's advantage be ϵ_5 in this game, and we have $\epsilon_5 \ge \frac{\epsilon_4}{2}$.

By looking at the simulation, it is clear that the challenger only needs the following information in order to faithfully answer all the oracle queries from the attacker. Recall that $v_0 = w_0$ and $v_1 = w_1$.

$$(g_1^{y^*}, g_2^{\frac{k_1^*}{y^*}}, \mathsf{H}(v_0), \mathsf{H}(v_1), \mathsf{H}(w_0), \mathsf{H}(w_1), \mathsf{H}(v_b)^{y^*}, \mathsf{H}(v_{\bar{b}})^{y^*}, \mathsf{H}(w_d)^{y^*}, \mathsf{H}(w_{\bar{d}})^{y^*})$$

In particular, the challenger does not need to know k_1^* and the value of b. Since the random number k_1^* only appears in $g_2^{\frac{k_1^*}{y^*}}$, the challenger indeed only needs the following to faithfully answer all the oracle queries from the attacker in the game.

$$(g_1^{y^*}, \mathsf{H}(v_0), \mathsf{H}(v_1), \mathsf{H}(w_0), \mathsf{H}(w_1), \mathsf{H}(v_b)^{y^*}, \mathsf{H}(v_{\bar{b}})^{y^*}, \mathsf{H}(w_d)^{y^*}, \mathsf{H}(w_{\bar{d}})^{y^*})$$

Game 6: The challenger performs the same as in Game 5, except for replacing $\mathsf{H}(v_b)^{y^*}$ with $R_1 \in_R \mathbb{G}_1$, replacing

$\mathsf{H}(v_{\bar{b}})^{y^*}$ with $R_2 \in_R \mathbb{G}_1$, replacing $\mathsf{H}(w_d)^{y^*}$ with $R_3 \in_R \mathbb{G}_1$, replacing $\mathsf{H}(w_{\bar{d}})^{y^*}$ with $R_4 \in_R \mathbb{G}_1$. Let the attacker's advantage be ϵ_6 in this game. It is clear that, in game 5 and Game 6, the attacker can be regarded as a distinguisher for the M-DDH$_5$ problem in $\mathbb{G}_1$, and $\epsilon_{m-ddh_5} \leq 25\epsilon_{ddh} \leq 25\epsilon_{sxdh}$. Therefore, we have $|\epsilon_6 - \epsilon_5| \leq 25\epsilon_{sxdh}$.

In Game 6, it is clear that $\epsilon_6 = 0$ because the challenge is independent from b. To sum up, we have $\frac{\epsilon}{q_2(q_2-1)(q_2-2)(q_2-3)} \leq (t^2+25)\epsilon_{sxdh}$. Since $q_2(q_2-1)(q_2-2)(q_2-3)$ and t are a polynomials in the security parameter, therefore, if ϵ_{sxdh} is negligible then ϵ is also negligible.

Based on the proofs for Case 1 and Case 2, the theorem now follows. ∎

VI. Concluding Remarks

Motivated by the work by Popa and Zeldovich [22], we have formulated a new primitive, namely multi-party searchable encryption (MPSE), for enabling users to selectively authorize each other to search in their encrypted data. Due to the user status dynamics, we presented a security model by considering the worst-case and average-case collusion scenarios simultaneously, and also proposed a new scheme with provable security. Even though our security model for MPSE provides stronger security guarantee than that from [22], a lot of interesting extensions remain possible. We list some of them below.

- In the formulation of MPSE, we assume that authorization is granted on index level, namely for each of her indexes Alice can decide whether Bob can search or not (if authorized Bob can try all keywords). It is interesting to have a formulation that supports authorizations on keyword level, namely Alice can authorize Bob to search for a subset of keywords in her indexes.
- As mentioned in Sections I-A and IV, if Alice is authorized by Bob to search his index, then the cloud server colluded with Bob can recover the keyword in all Alice's search queries. This is inevitable in our formulation because we require that Alice should be able to issue a single trapdoor to search all indexes that have been authorized to her by potentially many users (Alice (herself), Bob, Charlie, etc). If we allow Alice to have multiple key pairs and use them with different peers (this will mean that Alice needs to issue multiple trapdoors to search the indexes), then the above "inevitable" information leakage may be avoided. This deserves further investigation.
- In the formulation of MPSE, we assume that a forward index is generated for each document. One drawback of this kind of indexes is that it leaks some uncessary information, e.g. how many keywords are in the index. In contrast, inverted index structure may not have this problem, as shown in [13]. It is worth investigating a new formulation for MPSE by assuming an inverted index structure.

Acknowledgement.

The author would like to thank Afonso Arriaga (SnT, University of Luxembourg) and Liqun Chen (HP labs, Bristol) for their helpful discussions.

References

[1] M. Abdalla, E. Bresson, O. Chevassut, and D. Pointcheval. Password-based group key exchange in a constant number of rounds. In M. Yung, Y. Dodis, A. Kiayias, and T. Malkin, editors, *Public Key Cryptography — PKC 2006*, volume 3958 of *LNCS*, pages 427–442. Springer, 2006.

[2] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Order preserving encryption for numeric data. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 563–574. ACM, 2004.

[3] G. Ateniese, J. Camenisch, S. Hohenberger, and B. de Medeiros. Practical group signatures without random oracles. http://eprint.iacr.org/2005/385, 2005.

[4] F. Bao, R. H. Deng, X. Ding, and Y. Yang. Private query on encrypted data in multi-user settings. In L. Chen, Y. Mu, and W. Susilo, editors, *Proceedings of the 4th international conference on Information security practice and experience*, volume 4991 of *LNCS*, pages 71–85. Springer, 2008.

[5] A. Boldyreva, N. Chenette, Y. Lee, and A. O'Neill. Order-preserving symmetric encryption. In A. Joux, editor, *Advances in Cryptology — EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 224–241. Springer, 2009.

[6] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano. Public Key Encryption with Keyword Search. In C. Cachin and J. Camenisch, editors, *Advances in Cryptology — EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 506–522. Springer, 2004.

[7] D. Boneh, A. Raghunathan, and G. Segev. Function-private identity-based encryption: Hiding the function in functional encryption. In R. Canetti and J. A. Garay, editors, *Advances in Cryptology — CRYPTO 2013*, volume 8043 of *LNCS*, pages 461–478. Springer, 2013.

[8] C. Bosch, Q. Tang, P. Hartel, and W. Jonker. Selective document retrieval from encrypted databse. In D. Gollmann and F. C. Freiling, editors, *Information Security Conference - 15th Information Security Conference (ISC 2012)*, volume 7483 of *LNCS*, pages 224–241. Springer, 2012.

[9] X. Boyen. The uber-assumption family. In S. D. Galbraith and K. G. Paterson, editors, *Pairing-Based Cryptography — Pairing 2008*, volume 5209 of *LNCS*, pages 39–56. Springer, 2008.

[10] E. Bresson, O. Chevassut, and D. Pointcheval. Dynamic group diffie-hellman key exchange under standard assumptions. In L. R. Knudsen, editor, *Advances in Cryptology – EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 321–336. Springer, 2002.

[11] Y. Chang and M. Mitzenmacher. Privacy preserving keyword searches on remote encrypted data. In J. Ioannidis, A. D. Keromytis, and M. Yung, editors, *Proceedings of the Third international conference on Applied Cryptography and Network Security*, volume 3531 of *LNCS*, pages 442–455. Springer, 2005.

[12] M. Chase and S. Kamara. Structured encryption and controlled disclosure. In M. Abe, editor, *Advances in Cryptology — ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 577–594. Springer, 2010.

[13] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky. Searchable symmetric encryption: improved definitions and efficient constructions. In *Proceedings of the 13th ACM conference on Computer and Communications Security*, pages 79–88. ACM, 2006.

[14] C. Dong, G. Russello, and N. Dulay. Shared and searchable encrypted data for untrusted servers. In V. Atluri, editor, *Data and Applications Security XXII, 22nd Annual IFIP WG 11.3 Working Conference on Data and Applications Security*, volume 5094 of *LNCS*, pages 127–143. Springer, 2008.

[15] C. Dong, G. Russello, and N. Dulay. Shared and searchable encrypted data for untrusted servers. *Journal of Computer Security*, 19(3):367–397, 2011.

[16] E. J. Goh. Secure indexes. Technical Report 216, IACR, 2003.

[17] P. Golle, J. Staddon, and B. Waters. Secure conjunctive keyword search over encrypted data. In M. Jakobsson, M. Yung, and J. Zhou, editors, *Applied Cryptography and Network Security, Second International Conference*, volume 3089 of *LNCS*, pages 31–45. Springer, 2004.

[18] B. Hore, S. Mehrotra, M. Canim, and M. Kantarcioglu. Secure multidimensional range queries over outsourced data. *The VLDB Journal*, 21(3):333–358, 2012.

[19] M. Islam, M. Kuzu, and M. Kantarcioglu. Access pattern disclosure on searchable encryption: Ramification, attack and mitigation. In *Proceedings of the Network and Distributed System Security Symposium, NDSS 2012*, page To appear. Internet Society, 2012.

[20] F. Kerschbaum and A. Sorniotti. Searchable encryption for outsourced data analytics. In J. Camenisch and C. Lambrinoudakis, editors, *Public Key Infrastructures, Services and Applications - 7th European Workshop*, volume 6711 of *LNCS*, pages 61–76. Springer, 2011.

[21] M. Kuzu, M. Islam, S. Mohammad, and M. Kantarcioglu. Efficient similarity search over encrypted data. In *Proceedings of the 2012 IEEE 28th International Conference on Data Engineering*, pages 1156–1167. IEEE Computer Society, 2012.

[22] R. A. Popa and N. Zeldovich. Multi-key searchable encryption. http://eprint.iacr.org/2013/508, 2013.

[23] M. Raykova, A. Cui, B. Vo, B. Liu, T. Malkin, S. M. Bellovin, and S. J. Stolfo. Usable, secure, private search. *IEEE Security & Privacy*, 10:53–60, 2012.

[24] E. Shen, E. Shi, and B. Waters. Predicate privacy in encryption systems. In O. Reingold, editor, *Proceedings of the 6th Theory of Cryptography Conference on Theory of Cryptography*, volume 5444 of *LNCS*, pages 457–473. Springer, 2009.

[25] D. X. Song, D. Wagner, and A. Perrig. Practical Techniques for Searches on Encrypted Data. In *IEEE Symposium on Security and Privacy*, pages 44–55. IEEE Computer Society, 2000.

[26] Q. Tang. Privacy preserving mapping schemes supporting comparison. In *Proceedings of the 2010 ACM workshop on Cloud computing security workshop*, pages 53–58, 2010.

[27] Q. Tang. *Theory and Practice of Cryptography Solutions for Secure Information Systems*, chapter Search in Encrypted Data: Theoretical Models and Practical Applications, pages 84–108. IGI, 2013.