Multi-Key Searchable Encryption

Raluca Ada Popa and Nickolai Zeldovich
MIT CSAIL

Abstract

We construct a searchable encryption scheme that enables keyword search over data encrypted with *different* keys. The scheme is practical and was designed to be included in a new system for protecting data confidentiality in client-server applications against attacks on the server.

1 Introduction

A promising approach to preventing confidential data disclosures due to adversaries that compromise servers is to store only encrypted data on servers, and to encrypt and decrypt documents only on client machines. In the case of a multi-user application, each user may have access to a different set of documents stored on the server; this can be achieved by ensuring that each document is encrypted with a separate per-document key, and arranging for each user's client machine to have access to the keys of the documents that the corresponding user has access to.

One challenge with this approach lies in supporting applications that allow users to search for documents that contain a given word. Many applications, such as document sharing, chat, forums, and calendars, support search over documents shared by different users. Prior work on searchable encryption schemes would require the client to provide the server with a search token under each key that a matching document might be encrypted with, and thus the number of tokens scales with the number of documents to search. This can be slow when there is a large number of documents.

We present a cryptographic scheme that allows a client to provide a single search token to the server, but still allows the server to search for that token's word in documents *encrypted with different keys*. We call such a scheme *multi-key search*. Intuitively, the scheme hides the content of the document and the words one searches for, and the only information the server learns is whether some word being searched for matches a word in a document. We formalize the security guarantees with cryptographic security definitions and prove the security of our scheme under variants of the Bilinear Decisional Diffie-Hellman and External Diffie-Hellman assumptions, as well as in the random oracle model. The scheme is practical and was designed to be included in a new system for protecting data confidentiality against attacks on the server.

The most challenging aspect when coming up with such a scheme is that there is no single trusted user; for example, in many web applications, anyone, including an adversary, can create an account and become a user. As a result, users cannot agree on a secret, and each document must be encrypted under different keys that are generated independently, rather than generated from a common secret key. Another challenge is that the scheme must be practical because our goal is to use it in a real system.

In the rest of the paper, we describe the related work in Sec. 2, we explain the problem setting in Sec. 3, we provide syntax and security definitions in Sec. 5, we present our construction together with a performance measurement in Sec. 6 and 7, respectively, and finally, we prove the security of our scheme in Sec. 9 under the assumptions in Sec. 8.

2 Related work

Most of the research on searchable encryption [14, 10, 4, 7, 3, 8, 5, 2, 16, 15, 13, 11, 6] focused on the case when the data is encrypted with the same key, and considered various aspects of the resulting cryptosystem, such as public- versus secret-key, more expressive computation such as conjunctions and disjunctions, indexable schemes, and others.

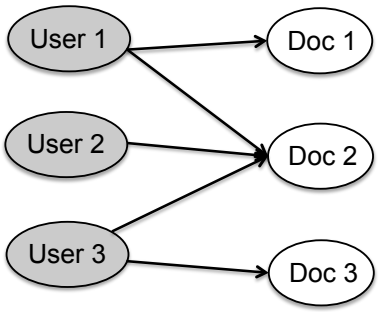

Figure 1: Access graph example.

To the best of our knowledge, Lopez-Alt et al. [12] is the only work considering computation over data encrypted with different keys. They design a fully homomorphic encryption (FHE) scheme in which anyone can evaluate a function over data encrypted with different keys. However, the decryption requires all the parties to come together and run an MPC protocol. Translated to our setting, this requires a client to retrieve all the keys under which the data is encrypted so the client still needs to do work proportional in the number of keys, which is what we are trying to avoid. Moreover, due to the semantic security of FHE, the server does not learn whether a document matches a keyword: it only learns the encryption of whether the document matches; therefore, the server would have to return the entire data, which is not practical.

A related scheme is the one of Bao et al. [2], who consider a setting where users have different keys but all the data is encrypted with one key and the search happens over data encrypted with one key. One cannot directly apply their scheme to the multi-key setting by creating an instance of the scheme for every key because this results in many search tokens; the reason is that the search tokens are tied to a secret different for every different key. Moreover, one requires different security definitions and security proofs when considering data encrypted under different keys with users only accessing a subset of them. Other works [8, 15, 16, 13] fall in the same category of multi-user one-key schemes, and have similar properties.

3 Problem setting

In our model, there is a set of users, a server, and a set of documents. The server stores encrypted documents. Each user has access to a subset of the documents. A user can create a document and then give access to other users to the document by giving them the decryption key of the document. We call the graph of user accesses to documents, an *access graph*, defined below. Fig. 1 shows an example of an access graph.

Definition 3.1 (Access graph)**.** *An access graph $G = (U, D, E)$ consists of a set of users U, a set of documents D, as well as a set of edges E, where an edge e is a pair (i, j) for $i \in U$ and $j \in D$ denoting user i has access to document j. We write $e \in G$ to mean that $e \in E$.*

At a high level, the following security guarantees are desirable. If some user was not given access to a document, the user should not be able to read the contents of that document or search over that document, *even if the user colludes with the server*. The setting is entirely distributed. Each user generates his key and there is no trusted party for choosing keys, and no globally trusted user. Moreover, there is no trusted party to create document keys or to help with providing access to documents.

The functionality goal is to allow a user to search a word over all the documents he can access, say n documents, even if those documents are encrypted under different keys. Note that the user has access to all the keys for these n documents, but the user should only give one search token to the server, instead of n tokens.

Let's now consider a more concrete model for such a multi-key search. We denote the key of user i with uk_i, and the key of document j with k_j. Consider that a user, say Alice, (with key uk_A) has n encrypted documents at the server, and each is encrypted under a key k_j for $j = 1 \ldots n$. Alice wants to search for a word w over all the documents she has access to, so she uses uk_A to compute a *token* for a word w. In order to allow the server to match the token against words encrypted with $k_1, \ldots, k_n$, Alice gives the server some public information called *delta*. Alice provides one delta per key k_j, denoted $\Delta_{\mathsf{uk}_A, k_j}$. The server can use $\Delta_{\mathsf{uk}_A, k_j}$ to convert a search token under key uk_A to a search token under k_j, a process we call *adjust*. In this way, the server can obtain tokens for word w under $k_1, \ldots, k_n$ while only receiving one token from Alice, and then performing a traditional single-key search with the new tokens.

Multi-key search provides efficiency guarantees over single-key search. If T is the total number of words Alice searches, she provides $O(n + T)$ pieces of information to the server: n deltas and T tokens, the size of all of which only depends on the security parameter. In contrast, if Alice uses a single-key searchable encryption as in previous work, she provides $O(nT)$ pieces of information to the sever, because she provides n tokens, one for each key k_j, for each of T words.

4 Preliminaries

We denote by κ the security parameter throughout this paper. For a distribution $\mathcal{D}$, we write $x \leftarrow \mathcal{D}$ when x is sampled from the distribution $\mathcal{D}$. If S is a finite set, by $x \leftarrow S$ we mean x is sampled from the uniform distribution over the set S.

We use $p(\cdot)$ to denote that p is a function that takes one input. Similarly, $p(\cdot, \cdot)$ denotes a function p that takes two inputs.

We say that a function f is negligible in an input parameter κ, if for all $d > 0$, there exists K such that for all $\kappa > K$, $f(\kappa) < k^{-d}$. For brevity, we write: for all sufficiently large κ, $f(\kappa) = \mathrm{negl}(\kappa)$. We say that a function f is polynomial in an input parameter κ, if there exists a polynomial p such that for all κ, $f(\kappa) \leq p(\kappa)$. We write $f(\kappa) = \mathrm{poly}(\kappa)$. A similar definition holds for $\mathrm{polylog}(\kappa)$.

Let $[n]$ denote the set $\{1, \ldots, n\}$ for $n \in \mathbb{N}^*$.

When saying that a Turing machine A is PPT we mean that A is a probabilistic polynomial-time machine.

Two ensembles, $X = \{X_\kappa\}_{\kappa \in \mathbb{N}}$ and $Y = \{Y_\kappa\}_{\kappa \in \mathbb{N}}$, are said to be *computationally indistinguishable* (denoted $\{X_\kappa\}_{\kappa \in \mathbb{N}} \overset{c}{\approx} \{Y_\kappa\}_{\kappa \in \mathbb{N}}$) if for every probabilistic polynomial-time algorithm D,

$$|\Pr[D(X_\kappa, 1^\kappa) = 1] - \Pr[D(Y_\kappa, 1^\kappa) = 1]| = \mathrm{negl}(\kappa).$$

We use asymmetric bilinear map groups of Type 2 for our construction [9]. Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two disjoint cyclic subgroups on an elliptic curve of Type 2, and let e be a non-degenerate bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. Let $\mathsf{params} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2, g_T) \leftarrow \mathsf{CSetup}(1^\kappa)$ be the procedure that generates curve parameters, where g_1, g_2, and g_T are generators of $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$.

5 Syntax and security definitions

We now formalize the syntax and security definitions.

Definition 5.1 (Multi-key search)**.** *A multi-key search scheme* MK *is a tuple of algorithms (*MK.Setup*,* MK.KeyGen*,* MK.Delta*,* MK.Token*,* MK.Enc*,* MK.Adjust*,* MK.Match*) as follows:*

- $\mathsf{params} \leftarrow \mathsf{MK.Setup}(1^\kappa)$*: Takes as input the security parameter and outputs system wide parameters.*
- $k \leftarrow \mathsf{MK.KeyGen}(\mathsf{params})$*: Takes as input the system parameters and outputs a secret key, which could be a key for a user or for a document.*

- $\Delta \leftarrow \mathsf{MK.Delta}(k_1, k_2)$*: Takes as input two keys and outputs a delta.*
- $\mathsf{tk} \leftarrow \mathsf{MK.Token}(k, w)$*: Takes as input a key* k *and a word* w *and outputs a search token* tk*.*
- $c \leftarrow \mathsf{MK.Enc}(k, w)$*: Takes as input a key* k *and a word* w *and outputs an encryption of the word* c*.*
- $\mathsf{stk} \leftarrow \mathsf{MK.Adjust}(\mathsf{tk}, \Delta)$*: Takes as input a token* tk *and a delta* Δ *and outputs a search token* tk'*.*
- $b \leftarrow \mathsf{MK.Match}(\mathsf{stk}, c)$*: Takes as input a search token* stk *and a ciphertext* c *and outputs a bit* b*.*

Correctness. *For any polynomial* $n(\cdot)$*, for every sufficiently large security parameters* κ*, for all* $w \neq w' \in \{0,1\}^{n(\kappa)}$*,*

$$\Pr\left[\begin{array}{l} \mathsf{params} \leftarrow \mathsf{MK.Setup}(1^\kappa); \\ \mathsf{uk} \leftarrow \mathsf{MK.KeyGen}(\mathsf{params}); k \leftarrow \mathsf{MK.KeyGen}(\mathsf{params}); \\ \Delta \leftarrow \mathsf{MK.Delta}(\mathsf{uk}, k); \\ \mathsf{stk} \leftarrow \mathsf{MK.Adjust}(\mathsf{MK.Token}(\mathsf{uk}, w), \Delta) : \\ \mathsf{MK.Match}(\mathsf{stk}, \mathsf{MK.Enc}(k, w)) = \textit{True and } \mathsf{MK.Match}(\mathsf{stk}, \mathsf{MK.Enc}(k, w')) = \textit{False} \end{array}\right] = 1 - \mathrm{negl}(\kappa).$$

Correctness says that when searching for a word w, encryptions of the word w in some document will match (after adjusting the token for w to the key of the document), but encryptions of a different word w' will not match the search.

For simplicity, we do not include a decryption algorithm in the syntax of the scheme, but a multi-key search scheme can be easily augmented with a decryption algorithm by appending to each ciphertext produced in MK.Enc a symmetric-key semantically secure encryption with the same key as the argument to MK.Enc.

Remark 5.2. *In an alternate syntax, each user has a public key* pk*, and the algorithm* MK.Delta *takes as input the public key of a user instead of his private key. A public-key* MK.Delta *algorithm has the advantage that when a user, say Alice, wants to give access to another user, say Bob, to a document, Alice can just compute the delta to the document for Bob and provide it to the server. (In fact, our construction can be adapted to public-key by setting the public key of a user to* $\mathsf{pk} = g_2^{1/\mathsf{uk}}$*, where* uk *is the secret key of the user.)*

However, inherently, such a multi-key scheme cannot hide the word searched for because the functionality of the scheme allows a dictionary attack. Assume that an adversary wants to learn what Alice searches for, and let pk_A *be Alice's public key. An adversary can create a document with some key* k*, and encrypt in this document every word of a dictionary using key* k*. Then, the adversary can produce a delta for Alice to this document by computing* $\Delta_A := \mathsf{MK.Delta}(\mathsf{pk}_A, k)$*. Now, for every search token* tk *of Alice, the adversary computes* $\mathsf{stk} := \mathsf{MK.Adjust}(\mathsf{tk}, \Delta_A)$ *and uses* stk *to find a match in the encrypted dictionary. Once a match is found, the adversary knows what word the user searched for.*

Intuitively, we want two security properties from the MK scheme: the ciphertext and the token should not reveal the value of the underlying plaintext, and the only information revealed to the server is whether a search token matches a ciphertext only when the server has a delta for some document or whether one is searching for the same word as before. Moreover, if the key of a document leaks, the key of the user should not leak and the contents of the other documents the user has access to should not leak.

We formalize these properties with two games, *data hiding* and *token hiding*, that express these goals. One holistic security definition would be a stronger guarantee, but that greatly complicates the proofs. Nevertheless, the separate definitions also capture the desired security goals.

5.1 Data hiding

Data hiding requires that the adversary not be able to distinguish between ciphertexts of two values not matched by some token. The case when the token matches a ciphertext is handled by the token hiding game. In the following definition, documents are numbered from 0 onwards and users from 1 onwards. The reason there is document 0 is that this is a special document used in the challenge.

Definition 5.3 (Data hiding game)**.** *The data hiding game is between a challenger* Ch *and an adversary* Adv *on security parameter* κ *and public parameters* params*.*

- Ch *computes* $\mathsf{params} \leftarrow \mathsf{CSetup}(1^\kappa)$ *and provides them to* Adv*.*
- Adv *provides an access graph* G *with users numbered from* 1 *and documents numbered from* 0 *to* Ch *along with keys* k_j *for every document with* $j > 0$*.*
- Ch *generates* $k_0 \leftarrow \mathsf{MK.KeyGen}(1^\kappa, \mathsf{params})$ *for document* 0*. Then, for every user* i*, it generates* $\mathsf{uk}_i \leftarrow \mathsf{MK.KeyGen}(1^\kappa)$ *and for every edge* $(i, j) \in G$*, it provides* $\mathsf{MK.Delta}(\mathsf{uk}_i, k_j)$ *to* Adv*.*
- *Challenge step:* Adv *chooses* $w_0^*, w_1^* \leftarrow \{0,1\}^{n(\kappa)}$ *and provides* w_0^*, w_1^* *to* Ch*.* Ch *chooses a random bit* b *and provides* $\mathsf{MK.Enc}(k_0, w_\mathsf{b}^*)$ *to* Adv*.*
- *Adaptive step:* Adv *makes the following queries to* Ch *adaptively. The* ℓ*-th query can be:*
  1. *"Encrypt* w_ℓ *to document* 0*":* Ch *returns* $\mathsf{MK.Enc}(k_0, w_\ell)$*.*
  2. *"Token for word* w_ℓ *for user* i*":* Ch *returns* $\mathsf{MK.Token}(\mathsf{uk}_i, w_\ell)$*.*
- Adv *outputs* b'*, its guess for* b*.*

Restriction on Adv*: for all token queries* w_ℓ *for user* i*, if* $(i, 0) \in G$*, it must be that* $w_\ell \notin \{w_0^*, w_1^*\}$*.*

Adv *wins the game if* $b' = b$*. Let* $\mathsf{win}_{\mathsf{Adv}}(\kappa)$ *be the random variable indicating whether* Adv *wins the game for security parameter* κ*.*

Definition 5.4. *A multi-key search scheme is data hiding if, for all* PPT *adversaries* Adv*, for all sufficiently large* κ*,* $\Pr[\mathsf{win}_{\mathsf{Adv}}(\kappa)] < 1/2 + \mathrm{negl}(\kappa)$*.*

Here is how the definition models our intentions:

- The fact that Adv can provide keys for all documents except for the challenge one models the fact that an adversary could steal keys of document or create documents, but such actions should not allow Adv to learn information about a document he does not own.
- The restriction on the token queries of Adv is required because otherwise Adv could distinguish the ciphertexts based on the functionality of the scheme.
- Note that Adv can ask tokens for words that are part of the challenge (e.g., w_0 or w_1) for users that do not have a delta to document 0. This ensures that any user i that does not have a delta to a document cannot search that document.
- We do not need to allow Adv to ask for encrypt queries to documents i for $i > 0$ because Adv has the corresponding secret keys and can encrypt by itself.

A stronger definition would allow an adaptive step before the challenge step as well. Our scheme can also be proven secure in that setting, but results in a more complicated proof, which we do not provide here.

5.2 Token hiding

Token hiding requires that an adversary cannot learn the word one searches for.

Definition 5.5. *A u-free document in a particular graph is a document with no edge from user u in that graph. A u-free user in a particular graph is a user that has edges only to u-free documents in that graph.*

User 0 will be the challenge user, for which Adv will have to distinguish tokens. Thus, we will refer to 0-free users and 0-free documents as simply "free users" and "free documents".

Definition 5.6 (Token hiding game)**.** *The token hiding game is between a challenger* Ch *and an adversary* Adv *on security parameter κ and public parameters* params*.*

- Ch *computes* $\mathsf{params} \leftarrow \mathsf{CSetup}(1^\kappa)$ *and provides them to* Adv*.*
- Adv *provides an access graph G with users numbered from 0 and documents numbered from 1, along with keys uk_i for every free user i and k_j for every free document j.*
- Ch *generates* $\mathsf{uk}_i \leftarrow \mathsf{MK.KeyGen}(1^\kappa)$ *for every non-free user i,* $k_j \leftarrow \mathsf{MK.KeyGen}(1^\kappa)$ *for every non-free document j. For every edge $(i, j) \in G$,* Ch *sends* $\mathsf{MK.Delta}(\mathsf{uk}_i, k_j)$ *to* Adv*.*
- *Adaptive step.* Adv *makes the following queries to* Ch *adaptively. At query ℓ:*
  1. *"Encrypt w_ℓ for document j":* Ch *returns* $\mathsf{MK.Enc}(k_j, w_\ell)$*.*
  2. *"Token w_ℓ for user i" with $i > 0$: receives* $\mathsf{MK.Token}(\mathsf{uk}_i, w_\ell)$*.*
- *Challenge step:* Adv *sends w_0^* and w_1^* to* Ch *and receives* $\mathsf{MK.Token}(\mathsf{uk}_0, w_\mathsf{b}^*)$ *for a random bit* b*.*
- Adv *repeats the adaptive step.*
- Adv *outputs* b'*, its guess for* b*.*

Restriction on Adv*: For every "Token w_ℓ for user i" query: $w_\ell \notin \{w_0^*, w_1^*\}$ or user i is free. For every "Encrypt w_ℓ for document j" query: $w_\ell \notin \{w_0^*, w_1^*\}$ or document j is free.*

Adv *wins the game if $b' = b$. Let* $\mathsf{win}_{\mathsf{Adv}}^{\mathsf{token}}(\kappa)$ *be the random variable indicating whether* Adv *wins the game for security parameter κ.*

Definition 5.7. *A multi-key search scheme is token-hiding if, for all* PPT *adversaries* Adv*, for all sufficiently large κ,* $\Pr[\mathsf{win}_{\mathsf{Adv}}^{\mathsf{token}}(\kappa)] < 1/2 + \mathrm{negl}(\kappa)$.

As before, the reason Adv can pick keys is to signify that Adv can corrupt certain users or documents, or can even create nodes in the access graph.

The constraints on the game are so that the adversary cannot distinguish the challenge words trivially, because the functionality of the scheme distinguishes them (either because there is a search match or the token is deterministic). Note that the definition (and in fact the scheme as well) allows an adversary to tell if two tokens are equal: in practice, if the same set of documents match a token, it is likely that the token is the same so we did not consider important to hide this equality relation among tokens. A solution for hiding the token is to use composite groups and multiply a random element from the second group to the token, but we do not explore this further here.

6 Construction

Let $H : \{0,1\}^* \to \mathbb{G}_1$ and $H_2 : \mathbb{G}_T \times \mathbb{G}_T \to \{0,1\}^*$ be hash functions, modeled as random oracles. Our multi-key search scheme is as follows:

- $\mathsf{params} \leftarrow \mathsf{MK.Setup}(1^\kappa)$: return $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2, g_T) \leftarrow \mathsf{CSetup}(1^\kappa)$.
- $k \leftarrow \mathsf{MK.KeyGen}(\mathsf{params})$: return $k \leftarrow \mathbb{Z}_p$.
- $\Delta \leftarrow \mathsf{MK.Delta}(k_1, k_2)$: return $\Delta = g_2^{k_2/k_1} \in \mathbb{G}_2$.
- $\mathsf{tk} \leftarrow \mathsf{MK.Token}(k, w)$: return $\mathsf{tk} = H(w)^k \in \mathbb{G}_1$.
- $c \leftarrow \mathsf{MK.Enc}(k, w)$: Draw $r \leftarrow \mathbb{G}_T$. Output $c = \big(r, H_2(r, e(H(w), g_2)^k)\big)$.
- $\mathsf{tk}' \leftarrow \mathsf{MK.Adjust}(\mathsf{tk}, \Delta)$: return $\mathsf{tk}' = e(\mathsf{tk}, \Delta) \in \mathbb{G}_T$.
- $b \leftarrow \mathsf{MK.Match}(\mathsf{tk}, c)$: Let $c = (r, h)$. Return $H_2(r, \mathsf{tk}) \stackrel{?}{=} h$.

Remark 6.1 (Alternate constructions)**.** *Using asymmetric pairings here is crucial for security. With symmetric pairings ($G_1 = G_2$), there is an attack that can determine the search word: given $H(w)$, $H(w)^k$, and $H(w_2)$, one can distinguish $H(w_2)^k$ from R by computing crossed pairings and thus can do a dictionary attack. Asymmetric groups prohibit applying the pairing between elements of $\mathbb{G}_1$.*

Another way to hide the search token would be to employ composite-order groups and multiply a random element $R \in G_h$ by the token for a word. One can also simulate composite order groups with standard groups using the orthogonal vector space techniques of Freeman and Lewko, which enables faster implementations.

Remark 6.2 (Indexed search)**.** *If the encryption scheme were deterministic, it would be easier to search for matches because an index could be constructed over the data. To make the scheme indexable in this way, one can modify* MK.Enc *to just output $e(H(w), g_2)^k$. If a user makes sure that there are no repetitions of words w in a document encrypted with the same key, making the encryption deterministic results in roughly the same security guarantees (although the data-hiding definitions need a few changes).*

Theorem 6.3. *The scheme above is a data- and token-hiding multi-key search scheme, based on the BDHV and XDHV assumptions in the random oracle model for H and H_2.*

Proof. We prove correctness of the scheme here, and in Sec. 9, we prove that it achieves the security properties.

Consider the setup from the correctness definition: $\mathsf{params} \leftarrow \mathsf{MK.Setup}(1^\kappa)$, $k_1 \leftarrow \mathsf{MK.KeyGen}(\mathsf{params})$, $k_2 \leftarrow \mathsf{MK.KeyGen}(\mathsf{params})$, $\Delta \leftarrow \mathsf{MK.Delta}(k_1, k_2)$, $\mathsf{tk} \leftarrow \mathsf{MK.Adjust}(\mathsf{MK.Token}(k_1, w), \Delta)$.

This means that $\mathsf{tk} = e(H(w)^{k_1}, g_2^{k_2/k_1}) = e(H(w), g_2)^{k_2}$.

Then $H_2(r, \mathsf{tk}) = H_2(r, e(H(w), g_2)^{k_2})$, so $\mathsf{MK.Match}(\mathsf{tk}, \mathsf{MK.Enc}(k_2, w))$ outputs True as desired.

The chance that $H_2(r, e(H(w), g_2)^{k_2}) = H_2(r, e(H(w'), g_2)^{k_2})$ is statistically negligible (in fact, it can be zero if the hash functions' output size is not smaller than the input size). Therefore, $\mathsf{MK.Match}(\mathsf{tk}, \mathsf{MK.Enc}(k_2, w'))$ outputs False. We thus showed correctness of our scheme.

□

7 Implementation

We implemented the scheme in C++ and used the PBC library [1] for implementation of a Type 2 curve [9], called Type D in the library. Below are evaluation results on an AMD Opteron(tm) Processor 2.4GHz, running on one core, when scheme is encrypting average-sized words, randomly generated. The scheme has a modest overhead.

Algorithm	MK.KeyGen	MK.Delta	MK.Token	MK.Enc	MK.Adjust	MK.Match
Time (ms)	0.35	6.3	0.89	6.3	5.5	0.0021

8 Assumptions

Our construction can be proven secure under variants of the Decisional Diffie-Hellman and External Diffie-Hellman assumptions, both of which are standard assumptions and were used in previous constructions, and in the random oracle model. Our assumptions are simple variants of these, and one can verify they hold in the generic group model.

Definition 8.1 (Bilinear Diffie-Hellman Variant (BDHV) assumption)**.** *For all* PPT *algorithms* Adv*, for every sufficiently large security parameter* κ*,*

$$|\Pr[\mathsf{params} \leftarrow \mathsf{CSetup}(1^\kappa); \quad a,b,c \leftarrow \mathbb{Z}_p : \mathsf{Adv}(\mathsf{params}, g_1^a, g_2^b, g_2^{1/a}, g_1^c, e(g_1,g_2)^{abc}) = 1] -$$
$$\Pr[\mathsf{params} \leftarrow \mathsf{CSetup}(1^\kappa); \quad a,b,c \leftarrow \mathbb{Z}_p, R \leftarrow \mathbb{G}_T : \mathsf{Adv}(\mathsf{params}, g_1^a, g_2^b, g_2^{1/a}, g_1^c, R) = 1]| = \mathrm{negl}(\kappa).$$

Definition 8.2 (External Diffie-Hellman Variant (XDHV) assumption)**.** *For all* PPT *algorithms* Adv*, for every sufficiently large security parameter* κ*,*

$$|\Pr[\mathsf{params} \leftarrow \mathsf{CSetup}(1^\kappa); \quad a,b,c,m \leftarrow \mathbb{Z}_p : \mathsf{Adv}(\mathsf{params}, g_1^a, g_1^b, g_1^{ab}, g_2^{ca}, g_2^{cd}, g_1^d, g_2^{1/d}) = 1] -$$
$$\Pr[\mathsf{params} \leftarrow \mathsf{CSetup}(1^\kappa); \quad a,b,c,m \leftarrow \mathbb{Z}_p, R \leftarrow \mathbb{G}_1 : \mathsf{Adv}(\mathsf{params}, g_1^a, g_1^b, R, g_2^{ca}, g_2^{cd}, g_1^d, g_2^{1/d}) = 1]|$$
$$= \mathrm{negl}(\kappa).$$

This assumption consists of the XDH assumption in the first three terms, but with extra information about a in the form of g_2^{ca}, but masked by c, which itself is masked by d.

As mentioned, we also model the hash functions H and H_2 as random oracles.

9 Security proof

The proofs are in the random oracle model for H and H_2, and H is a programmable random oracle.

To show that our scheme is secure with either of the security games, we consider a sequence of hybrid games starting from the game in the security definition in consideration, moving through gradually simpler games, and reaching the final game; in the final game, no adversary can guess the challenge bit b correctly with more than negligible chance information-theoretically.

During the sequence of hybrid games, we will sometimes show that Game "target" $\Leftarrow$ Game "new", meaning that if a scheme is secure in Game "new", it will be secure in Game "target", so it suffices to prove that the scheme is secure in Game "new".

Other times, we will show that Game "target" $\Leftrightarrow$ Game "new" meaning that the games are computationally indistinguishable. We will not review here the notion of game indistinguishability. Loosely speaking, any PPT adversary $\mathcal{D}$ playing the role of the adversary in Game "target" and Game "new" cannot tell in which of the

two games it is. If two games are computationally indistinguishable and no PPT adversary can win in one game with more than negligible probability, then no PPT adversary can win in the other game either.

For brevity, we do not include in the hybrid games the initial step in which Ch computes $\mathsf{params} \leftarrow \mathsf{CSetup}(1^\kappa)$ and provides them to Adv, the fact that Adv always has access to H_1 and H_2, as well as the final step when Adv outputs his guess for b, the challenger's bit. For clarity, we highlight certain parts of a game in blue, to indicate that these are differences from the previous game.

9.1 Data hiding proof

Proof. The sequence of hybrid games in the proof are related as follows:

Data hiding game $\Leftarrow$ Game 1 $\Leftarrow$ Game 2 $\Leftarrow$ Game 3 $\overset{\text{BDHV}}{\Leftrightarrow}$ Game 4 $\Leftarrow$ Game 5.

Games 1– 3 provide gradual simplications of the original game. Game 4 is computationally indistinguishable from Game 3 based on the BDHV assumption. In Game 5, any adversary has chance of guessing statistically close to $1/2$.

Game 1 no longer has the keys k_j for $j > 0$: see the difference in blue. We will also replace the algorithms of the multi-key scheme with the exact quantities returned.

Game 1

- Adv_1 provides an access graph G with one document, labeled 0, and any number of users.
- Ch_1 generates $k_0 \leftarrow \mathsf{MK.KeyGen}(1^\kappa, \mathsf{params})$ for document 0. Then, Ch_1 provides $g_1^{k_0/\mathsf{uk}_i}$ for every edge $(i, 0) \in G$, and g_1^{1/uk_i} for every user i.
- Adv_1 provides w_0^*, w_1^*.
- Ch_1 chooses a random bit b and provides $r^*, H_2(r^*, e(H(w_\mathsf{b}^*), g_2)^{k_0})$.
- Adaptive step: Adv_1 makes the following queries to Ch_1 adaptively. The ℓ-th query can be:
  1. "Encrypt w_ℓ": Ch_1 returns $r_\ell, H_2(r_\ell, e(H(w_\ell), g_2)^{k_0})$.
  2. "Token for word w_ℓ for user i": Ch_1 returns $H(w_\ell)^{\mathsf{uk}_i}$.

Restriction on Adv_1: for all token queries w_ℓ for user i, if $(i, 0) \in G$, it must be that $w_\ell \notin \{w_0^*, w_1^*\}$.

If the scheme is secure in this game, then it is secure in the data hiding game. The reason is that if there is there is a PPT adversary Adv that wins in the data hiding game, there is a PPT adversary Adv_1 that wins the Game 1. Adv_1 can use Adv to win Game 1. Adv_1 can simulate the inputs to Adv by simply storing the k_j values from Adv and computing g^{k_j/uk_i} when given g^{1/uk_i}, as in the third step of the data-hiding game that Adv is expecting.

Next, we would like to remove from the game users that do not have access to document 0. The intuition is that whatever information the adversary gets about those users is unrelated to document 0 and hence to the challenge. We create a new game in which the adversary creates only users with access to document 0.

Game 2

- Adv_2 provides an access graph G with one document, labeled 0, and any number of users all with access to document 0.
- Ch_2 generates $k_0 \leftarrow \mathsf{MK.KeyGen}(1^\kappa, \mathsf{params})$ and provides $g_2^{k_0/\mathsf{uk}_i}$ and g_2^{1/uk_i} for every user i.
- Adv_2 provides w_0^*, w_1^*.
- Ch_2 chooses a random bit b and provides $r^*, H_2(r^*, e(H(w_\mathsf{b}^*), g_2)^{k_0})$.
- Adaptive step: Adv_2 makes the following queries to Ch_2 adaptively. The ℓ-th query can be:
  1. "Encrypt w_ℓ": Ch_2 returns $r_\ell, H_2(r_\ell, e(H(w_\ell), g_2)^{k_0})$.
  2. "Token for word w_ℓ for user i": Ch_2 returns $H(w_\ell)^{\mathsf{uk}_i}$.

Restriction on Adv_2: for all w_ℓ in token queries, $w_\ell \notin \{w_0^*, w_1^*\}$.

Claim 9.1. *If a scheme is secure in Game 2, the scheme is secure in Game 1.*

Proof. For contradiction, let Adv_1 be an adversary that breaks Game 1, and let us construct an adversary Adv_2 that breaks Game 2. Adv_2's strategy is as follows: for users i with access to doc 0, Adv_2 uses its challenger Ch_2 to answer token queries of Adv_1; for other users, Adv_2 generates a random key for each such user i, uk_i, and answers Adv_1's queries using that key.

Let Ch_2 be the challenger of Adv_2 in Game 2. Adv_2 works as follows:

1. Receive a graph G from Adv_1. Construct a graph G' which is G from which we remove the users with no access to doc 0 as well as their edges. Provide G' to Ch_2. Receive $g_2^{k_0/\mathsf{uk}_i}$ and g_2^{1/uk_i} for every user $i \in G'$ from Ch_2. Choose $\mathsf{uk}_i \leftarrow \mathsf{MK.KeyGen}(1^\kappa)$ for all users $i \in G' - G$. For every edge $(i, 0)$, compute g_2^{1/uk_i}. Provide all this information to Adv_1.
2. Adv_2 gets w_0^* and w_1^* from Adv_1, forwards them to Ch_2 and returns Ch_2's answer.
3. Adaptive step: answer Adv_1's queries as follows:
   - Forward any encrypt query to Ch_2 and provide Ch_2's result to Adv_1.
   - Forward any token request for user $i \in G'$ to Ch_2 and return answer to Adv_1. Compute $H(w_\ell)^{\mathsf{uk}_i}$ for every user $i \in G - G'$ using the generated uk_i.
4. Adv_2 outputs Adv_1's decision.

We can see that since Adv_1 makes no token queries containing w_0^*, w_1^* for users with access to doc 0, Adv_2 will also satisfy the restriction in Game 2.

We can see that Adv_2 simulates Adv_1's inputs perfectly and when Adv_1 distinguishes, so does Adv_2; since Adv_1 wins in Game 1 with nonnegligible probability, Adv_2 also wins in Game 2 with the same probability, concluding the proof.

□

We would like to simplify the game by only allowing encryption queries to w_0^* and w_1^*. Note that Adv_2 can compute by himself the result of any encrypt query for a word $w_\ell \notin \{w_0^*, w_1^*\}$ by simply requesting a token for w_ℓ for any user and using the delta information $g_2^{k_0/\mathsf{uk}_i}$. So it suffices to receive encryptions for the w_0^* and w_1^* only, as in the following game.

Game 3

- Adv_3 provides an access graph G with one document, labeled 0, and any number of users all with access to document 0.
- Ch_3 generates $k_0 \leftarrow \mathsf{MK.KeyGen}(1^\kappa, \mathsf{params})$ and provides $g_2^{k_0/\mathsf{uk}_i}$ and g_2^{1/uk_i} for every user i.
- Adv_3 provides w_0^*, w_1^*.
- Ch_3 chooses a random bit b and provides $r^*, H_2(r^*, e(H(w_\mathsf{b}^*), g_2)^{k_0})$.
- Adaptive step: Adv_3 makes the following queries to Ch_3 adaptively. The ℓ-th query can be
  - "Encrypt w_ℓ", for $w_\ell \in \{w_0^*, w_1^*\}$: Ch_3 returns $r_\ell, H_2(r_\ell, e(H(w_\ell), g_2)^{k_0})$
  - "Token for word $w_\ell \notin \{w_0^*, w_1^*\}$ for user i": Ch_3 returns $H(w_\ell)^{\mathsf{uk}_i}$.

Claim 9.2. *If a scheme is secure in Game 3, the scheme is secure in Game 2.*

Proof. For contradiction, assume there is a PPT adversary Adv_2 that can break Game 2, and let us show how to construct an PPT adversary Adv_3 that can break Game 3.

Let Ch_3 be the challenger of Adv_3 in Game 3. The idea is that Adv_3 will answer encrypt queries for word $w_\ell \notin \{w_0^*, w_1^*\}$ by asking for a token for w_ℓ and then computing the ciphertext, and for words w_0^* or w_1^*, by asking Ch_3 for encryptions. Adv_3 proceeds as follows.

1. Adv_3 receives the graph G from Adv_2. Adv_3 creates an additional user I with edge to document 0 and adds it to G. Adv_3 sends the new graph to Ch_3, records the answers from Ch_3 and returns all answers to Adv_2 except for $g_2^{k_0/\mathsf{uk}_I}$ and g_2^{1/uk_I}.
2. Challenge step: Adv_3 receives w_0^*, w_1^* from Adv_2 and provides them to Ch_3. Adv_3 forwards these to Ch_3 and receives $r^*, H_2(r^*, e(H(w_\mathsf{b}^*), g_2)^{k_0})$. Adv_3 sends all these values to Adv_2.
3. Adv_3 answers the queries of Adv_2 from the adaptive step as follows:
   - "Encrypt w_ℓ" : If $w_\ell \in \{w_0^*, w_1^*\}$, Adv_3 sends this query to Ch_3 and returns Ch_3's result. Else Adv_3 asks Ch_3 for "token w_ℓ user I", receives $H(w_\ell)^{\mathsf{uk}_I}$ and computes $r, H_2(r, e(H(w_\ell), g_2)^{k_0})$ for some $r \leftarrow \mathbb{Z}_p$ by using k_0/uk_I.
   - "Token w_ℓ for user i": forward this query to Ch_3 and send the response to Adv_2.
4. Adv_3 outputs Adv_2 decision.

We can see that Adv_3 plays the game with Ch_3 correctly because it never asks Ch_3 for encryption to words not in $\{w_0^*, w_1^*\}$. Moreover, Adv_3 simulates the inputs to Adv_2 exactly so Adv_3 also has a nonnegligible chance of deciding correctly equal to the one of Adv_2, which concludes the proof.

□

We now use the BDHV assumption to replace $e(H(w_b^*), g_2)$ with a random value R, which is desirable so that the adversary loses the information about b that $e(H(w_b^*), g_2)$ provides.

Game 4

- Adv_4 provides an access graph G with one document, labeled 0, and any number of users all with access to document 0.
- Ch_4 generates $k_0 \leftarrow \mathsf{MK.KeyGen}(1^\kappa, \mathsf{params})$ and provides $g_2^{k_0/\mathsf{uk}_i}$ and g_2^{1/uk_i} for every user i.
- Adv_4 provides w_0^*, w_1^*.
- Ch_4 chooses a random bit b and provides $r^*, H_2(r^*, R)$ for $R \leftarrow \mathbb{G}_T$.
- Adaptive step: Adv_4 makes the following queries to Ch_4 adaptively. The ℓ-th query can be:
  1. "Encrypt w_ℓ" for $w_\ell \in \{w_0^*, w_1^*\}$: If $w_\ell = w_\mathsf{b}^*$, Ch_4 returns r_ℓ and $H_2(r_\ell, R)$, else Ch_4 returns $r_\ell, H_2(r_\ell, R^\alpha)$, where α is such that $g_1^\alpha = H(w_{1-b}^*)/H(w_\mathsf{b}^*)$.
  2. "Token for word w_ℓ for user i" for $w_\ell \notin \{w_0, w_1\}$: Ch_4 returns $H(w_\ell)^{\mathsf{uk}_i}$.

Claim 9.3. *Assuming BDHV and that H is a programmable random oracle, Game 3 and Game 4 are computationally indistinguishable.*

Proof. For contradiction, we assume that there is a PPT adversary $\mathcal{D}$ that distinguishes the two games, and show how to construct a PPT reduction B that breaks BDHV.

B receives as input $\mathsf{params}, g_1^a, g_2^b, g_2^{1/a}, g_1^c$ and T, where T is either $e(g_1, g_2)^{abc}$ or random. To distinguish what is T, B proceeds as follows.

B wants to embed some of the values from its challenge into the random oracle results when $\mathcal{D}$ queries for w_0^* or w_1^*. However, $\mathcal{D}$ could make queries to these values before declaring to B the values in the challenge step.

As a solution, B will guess which of the queries to the random oracle H are for challenge values. Without loss of generality, assume that $\mathcal{D}$ makes unique queries to H. We have three cases:

- B makes no query to the random oracle H including w_0^* or w_1^* before the challenge step.
- B queries exactly one of w_0^* and w_1^* to H before the challenge step.
- B queries both w_0^* and w_1^* to H before the challenge step.

Let i_0 be the guessed index of the query to H in which B requests w_0^*; i_0 could be $\perp$ if B does not request this value before the challenge step. Let p be a polynomial upper-bounding the runtime of $\mathcal{D}$ and hence the number of queries to H that $\mathcal{D}$ makes. B assigns a probability of $1/3$ to each case above and draws i_0, i_1 from $1, \ldots, p(\kappa)$.

When $\mathcal{D}$ provides w_0^* and w_1^* to B in the challenge step, B can check whether it guessed i_0 and i_1 correctly. If it did not, B outputs a random guess in its game, and halts.

- *Initialization:* B generates params and sends them to $\mathcal{D}$. B chooses $\alpha \leftarrow \mathbb{Z}_p$.
- *H simulation:* Initialize oracle. For each query w of $\mathcal{D}$ to H, B does:
  - If this is the i_0-th query, return g_1^c.
  - If this is the i_1-th query, return $g_1^{c\alpha}$.

– Otherwise, choose $q \leftarrow \mathbb{Z}_p$, store $\mathsf{oracle}[w] := q$ and return g_1^q.

- B receives a graph G from $\mathcal{D}$. For each user $i > 1$, let $\Delta_i \leftarrow \mathbb{Z}_p$ and let $\Delta_1 := 1$. Instead of $g_2^{k_0/\mathsf{uk}_i}$, provide g_2^{b/Δ_i}, and instead of g_2^{1/uk_i}, provide $g_2^{1/a}$ to $\mathcal{D}$.
- Challenge step: Receive w_0^* and w_1^* from $\mathcal{D}$. Validate whether i_0 and i_1 were correct guesses. If not, output a bit at random and halt. Else, provide $r^*, H_2(r^*, T)$ to $\mathcal{D}$.
- For each query of $\mathcal{D}$ during adaptive step:
  - For "encrypt w_ℓ": if $w_\ell = w_{\mathsf{b}}^*$, return r_ℓ and $H_2(r_\ell, T)$, else return r_ℓ, $H_2(r_\ell, T^\alpha)$.
  - For "Token w_ℓ user i": return $g_1^{a\Delta_i \mathsf{oracle}[w_\ell]}$.
- Output $\mathcal{D}$'s answer.

Let us argue that B simulates the inputs to $\mathcal{D}$ correctly. All the inputs to the random oracle H are correctly distributed, and the chance that c equals some value q drawn by B is statistically small.

B will have a chance of $1/\mathrm{poly}$ of guessing correctly i_0 and i_1. Therefore, all we have to show is that when B guesses these values correctly, B has a nonnegligible chance of outputting b.

For this purpose, let us show that the inputs B provides to $\mathcal{D}$ are statistically close to the inputs from Game 3. Consider the following change of variables and note it preserves distributions:

$$a \leftrightarrow \mathsf{uk}_1, \quad b \leftrightarrow k_0/\mathsf{uk}_1, \quad g_1^c \leftrightarrow H(w_{\mathsf{b}}^*), \quad g_1^{c\alpha} \leftrightarrow H(w_{1-\mathsf{b}}^*), \quad \Delta_i \leftrightarrow \mathsf{uk}_i/\mathsf{uk}_1$$

B sends $\mathcal{D}$: $g_2^{1/a\Delta_i} = g_2^{1/\mathsf{uk}_i}$, $g_2^{b/\Delta_i} = g_2^{k_0/\mathsf{uk}_i}$.

For "encrypt" and the challenge step, note that if $T = e(g_1, g_2)^{abc}$ then $T = e(H(w_{\mathsf{b}}^*), g_2)^{k_0}$ as in Game 3, else T has the same distribution as R in Game 4.

For "token", $g_1^{a\Delta_i \times \mathsf{oracle}[w_\ell]} = H(w_\ell)^{\mathsf{uk}_i}$, as desired.

Finally, when $\mathcal{D}$ distinguishes Game 3 from Game 4, B also breaks the BDHV assumption, which completes the proof.

□

Note that in Game 4, all the information using uk_i, k_0 is useless to an adversary because the challenge ciphertexts do not depend on these values. Therefore, we can simplify further the game:

Game 5

- Adv_5 provides w_0^*, w_1^*.
- Ch_5 chooses a random bit b and provides $r^*, H_2(r^*, R)$ for $R \leftarrow \mathbb{G}_T$.
- Adv_5 can repeat the following query; query ℓ is "Encrypt w_ℓ" for $w_\ell \in \{w_0^*, w_1^*\}$: Ch_5 draws $r_\ell \leftarrow \mathbb{Z}_p$; if $w_\ell = w_{\mathsf{b}}^*$, Ch_5 returns $r_\ell, H_2(r_\ell, R)$, else Ch_5 returns $r_\ell, H_2(r_\ell, R^\alpha)$, where α is such that $g_1^\alpha = H(w_{1-b}^*)/H(w_{\mathsf{b}}^*)$.

By the security of the random oracle H_2, no Adv can distinguish in Game 5 with non-negligible probability, concluding our proof.

□

9.2 Token hiding proof

Proof. We will create a set of hybrid games that progressively simplify the game until it becomes easy to show that Adv cannot learn b.

The first game, Game 1 is the same as the token hiding game except that it removes the encrypt queries. The intuition is that the output of the encrypt algorithm in our construction can be deduced from the outputs of the token and delta algorithms.

Game 1

1. Adv_1 provides G along with keys uk_i and k_j for free users and documents.
2. Ch_1 generates a new key for every non-free user i and document j using $\mathsf{MK.KeyGen}(1^\kappa)$. For every edge $(i, j) \in G$, Ch_1 sends $\mathsf{MK.Delta}(\mathsf{uk}_i, k_j)$ to Adv_1.
3. Adaptive step: Adv_1's ℓ-th query is "Token w_ℓ for user i" and Adv_1 receives $\mathsf{MK.Token}(\mathsf{uk}_i, w)$.
4. Adv_1 provides w_0^* and w_1^* to Ch_1. Ch_1 chooses $b \leftarrow \{0, 1\}$ and sends $\mathsf{MK.Token}(\mathsf{uk}_0, w_b^*)$ to Adv_1.
5. Adv_1 runs the adaptive step again.

Restriction on Adv_1: For every "Token w_ℓ for user i" query: $w_\ell \notin \{w_0^*, w_1^*\}$ or user i is free.

Claim 9.4. *If a scheme is secure with Game 1, the scheme must be token hiding.*

Proof. For contradiction, assume there is a PPT adversary Adv that wins the token hiding game, and let us construct a PPT adversary Adv_1 that wins Game 1. Let Ch_1 be the challenger in Game 1. Adv_1 uses Adv as follows.

- On input a graph G and keys from Adv, Adv_1 simply forwards these to Ch_1. Adv_1 forwards the responses from Ch_1 to Adv and records these as well.
- Adaptive step: For "Token" queries, Adv_1 sends the same queries to Ch_1 and forwards the responses to Adv.

  For a query "Encrypt w_ℓ for document j", Adv_1 proceeds as follows. If document j is free, Adv_1 knows k_j from Adv so it simply computes $\mathsf{MK.Enc}(k_j, w_\ell)$. If document j is non-free, Adv_1 must have a delta between user 0 and document j, say $\Delta_{0,j}$. Adv_1 requests "Token w_ℓ for document 0" to Ch_1, which is a valid request because $w_\ell \notin \{w_{0,\alpha}, w_{1,\alpha}\}_\alpha$ because of the constraints on Adv. Upon receiving token Token back, Adv_1 sends $r, H_2(r, e(t, \Delta_{0,j}))$ to Adv.
- Adv_1 forwards the challenges from Adv to Ch_1 and sends Ch_1's answer to Adv_1.
- Adv_1 proceeds as above in the second adaptive step.
- Adv_1 ouputs Adv's answer.

We can see that Adv_1 simulates Adv's inputs perfectly. Since Adv wins in the token hiding game with non-negligible probability, so will Adv_1 win in Game 1.

□

To simplify the game further, we would like to remove the free documents and the free users from the game, and only work with non-free users and documents.

Game 2

1. Adv_2 provides a graph G that has only non-free documents and users.
2. Ch_2 generates a new key for every user i and document j using $\mathsf{MK.KeyGen}(1^\kappa)$. For every edge $(i,j) \in G$, Ch_2 provides $g_2^{k_j/\mathsf{uk}_i}$ to Adv_2. For every user $i > 0$, Ch_2 provides g_2^{1/uk_i}.
3. Adaptive step: Adv_2's ℓ-th query can be "Token w_ℓ for user i", in which case it receives $H(w_\ell)^{\mathsf{uk}_i}$ from Ch_2.
4. Adv_2 provides w_0^* and w_1^* to Ch_2. Ch_2 chooses b at random and provides $H(w_\mathsf{b}^*)^{\mathsf{uk}_0}$ to Adv_2.
5. Adv_2 runs the adaptive step again.

Restriction on Adv_2: $w_\ell \notin \{w_0^*, w_1^*\}$, for all ℓ.

Claim 9.5. *If a scheme is secure with Game 2, the scheme is secure with Game 1.*

Proof. For contradiction, assuming there is a PPT adversary Adv_1 for Game 1, let us show how to construct a PPT reduction Adv_2 that wins in Game 2. Let Ch_2 be the challenger of Adv_2 in Game 2. Adv_2 works as follows:

- Receive G from Adv_1 along with uk_i and k_j for all free nodes. Remove from G all free nodes and thus obtain a new graph G'. Send G' to Ch_2. Store uk_i, k_j.
- Ch_2 replies with $g_2^{k_j/\mathsf{uk}_i}$ and g_2^{1/uk_i} corresponding to non-free nodes. Adv_2 needs to compute all deltas for G for Adv_1. For an edge between two free nodes, Adv_2 has both keys so it can directly compute the delta. For an edge between two non-free nodes, Adv_2 got $g_2^{k_j/\mathsf{uk}_i}$ from Adv_1. For an edge between a non-free user i and a free document j, Adv_2 knows g_2^{1/uk_i} and k_j so it can compute delta $g_2^{k_j/\mathsf{uk}_i}$. To provide g_2^{1/uk_i} to Adv_1, Adv_2 either uses its knowledge of uk_i for free users or receives this value from Ch_2.
- Adv_2 now answers Adv_1's queries, which are of the form "Token w_ℓ for user i". We have two cases. If i is a free user, Adv_2 can directly compute the token using uk_i. If i is non-free, i can ask Ch_2 for the token and forward it to Adv_2.

We can see that Adv_2 still satisfies the constraints of its game and simulates the inputs to Adv_1 perfectly. Moreover, whenever Adv_1 wins, Adv_2 wins as well.

□

We now write a final hybrid in which $H(w_{b,\ell})^{\mathsf{uk}_0}$ is a random value preserving the equality relations of $w_{b,\ell}$. Claim 9.6 shows that Game 2 and Game 3 are computationally indistinguishable.

Game 3

1. Adv_3 provides G with only non-free documents and users.

2. Ch_3 generates a new key for every user i and document j using $\mathsf{MK.KeyGen}(1^\kappa)$. For every edge $(i,j) \in G$, Ch_3 provides $g_2^{k_j/\mathsf{uk}_i}$ to Adv_3. For every user $i > 0$, Ch_3 provides g_2^{1/uk_i}.
3. Adaptive step: Adv_3 queries ℓ-th query: "Token w_ℓ for user i" and receives $H(w_\ell)^{\mathsf{uk}_i}$.
4. Adv_3 provides w_0^* and w_1^* to Ch_3. Ch_3 chooses b at random and sends R, for $R \leftarrow \mathbb{G}_T$.
5. Adv_3 runs the adaptive step again.

Restriction on Adv_3: $w_\ell \notin \{w_0^*, w_1^*\}$, for all ℓ.

We can see that in this game Adv_3 receives no information about b information theoretically. The chance Adv_3 has to guess b is exactly $1/2$, which completes our proof.

Claim 9.6. *Assuming XDHV holds, Game 2 is computationally indistinguishable from Game 3, in the random oracle model for H.*

Proof. For contradiction, assume that there is a PPT adversary $\mathcal{D}$ that can distinguish the two games (i.e., distinguish between Ch_2 and Ch_3). Let us construct a PPT adversary B that can break the XDHV assumption.

Let p be a polynomial in which $\mathcal{D}$ runs. As in the proof of data hiding, B wants to embed a ciphertext from its challenge, g_1^b into the oracle result to $\mathcal{D}$, when $\mathcal{D}$ queries for the challenge ciphertext w_b^*. However, $\mathcal{D}$ can query w_b^* to H before the challenge step, so before B knows the value of w_b^*. Therefore, B will guess which of the queries to the random oracle H are w_b^*. If during the challenge step the guess turns out to be incorrect, B outputs a bit at random and halts. Otherwise B proceeds.

B receives as input $g_1^a, g_1^b, T, g_2^{ca}, g_2^{cd}, g_1^d, g_2^{1/d}$ and must decide if $T = g_1^{ab}$ or T is random.

- *H simulation:* B flips a coin, and if the coin is heads, B predicts that $\mathcal{D}$ will never ask for w_b^* to the random oracle; otherwise, B predicts that $\mathcal{D}$ with ask for w_b^* and chooses an index at random $I \in \{0, \ldots, p(\kappa)\}$ to represent the index of the query during which $\mathcal{D}$ will ask for w_b^*. For each query w of $\mathcal{D}$ to H, B does:
  - If this is the I-th query, return g_1^b.
  - Otherwise, choose $q \leftarrow \mathbb{Z}_p$, store $\mathsf{oracle}[w] := q$ and return g_1^q.
- *Initialization.* B starts adversary $\mathcal{D}$ and receives a graph G. B provides the following information for the graph. For each document j, let $\alpha_j \leftarrow \mathbb{Z}_p$ if $j > 1$ and let $\alpha_1 := 1$ for $j = 1$. For each user i, let $\Delta_i \leftarrow \mathbb{Z}_p$ if $i > 1$, and let $\Delta_1 := 1$ if $i = 1$.
  - $g_2^{k_j/\mathsf{uk}_0} := g_2^{dc\alpha_j}$, for $j \geq 1$.
  - $g_2^{k_j/\mathsf{uk}_i} := g_2^{ac\alpha_j/\Delta_i}$.
  - $g_2^{1/\mathsf{uk}_i} := g_2^{1/d\Delta_i}$.
- Adaptive step: If user is 0, B outputs $g_1^{a\mathsf{oracle}[w]}$. Otherwise, user $i > 0$, and B outputs $g_1^{d\Delta_i\mathsf{oracle}[w]}$. Note that it is crucial that $w \neq w_\mathsf{b}^*$ because B would not know $\mathsf{oracle}[w_\mathsf{b}^*]$ (which should be b).
- B receives w_0^* and w_1^*. B checks if w_b^* is indeed the I-th element $\mathcal{D}$ queried to H, B's guess. If not, B outputs a random bit and halts. Otherwise, B sends T to $\mathcal{D}$.

- B proceeds as in the adaptive step.
- B outputs $\mathcal{D}$'s decision.

Let us argue that B simulates the inputs to $\mathcal{D}$ correctly, whenever B does not halt early. All the inputs to the random oracle H are uniformly random distributed, and the chance that a equals some value q drawn by B is statistically small.

Consider the following change of variables and note that it preserves distributions:

$$a \leftrightarrow \mathsf{uk}_0, \;\; g_1^b \leftrightarrow H(w_\mathsf{b}^*), \;\; c \leftrightarrow k_1/\mathsf{uk}_1\mathsf{uk}_0, \;\; d \leftrightarrow \mathsf{uk}_1, \;\; \Delta_i \leftrightarrow \mathsf{uk}_i/\mathsf{uk}_1, \;\; \alpha_j \leftrightarrow k_j/k_1.$$

The quantities $\mathcal{D}$ receives are:

- For $g_2^{k_j/\mathsf{uk}_0}$ with $j \geq 1$: $g_2^{dc\alpha_j} = g_2^{\mathsf{uk}_1 \frac{k_1}{\mathsf{uk}_0\mathsf{uk}_1} \frac{k_j}{k_1}} = g_2^{k_j,\mathsf{uk}_0}$, as desired.
- For $g_2^{k_j/\mathsf{uk}_i}$: $g_2^{ac\alpha_j/\Delta_i} = g_2^{\frac{k_1}{\mathsf{uk}_1}\frac{\alpha_j}{\Delta_i}} = g_2^{k_j/\mathsf{uk}_i}$, as desired,
- For g_2^{1/uk_i}: $g_2^{1/d\Delta_i} = g_2^{\frac{1}{\mathsf{uk}_1\Delta_i}} = g_2^{1/\mathsf{uk}_i}$, as desired.
- Adaptive step: $g_1^{a\mathsf{oracle}[w_\ell]} = H(w_\ell)^{\mathsf{uk}_0}$, and $g_1^{d\Delta_i\mathsf{oracle}[w]} = H(w)^{\mathsf{uk}_i}$ as desired.
- If T is random, the challenge step is as in Game 3. When $T = g_1^{ab} = H(w_\mathsf{b}^*)^{\mathsf{uk}_0}$, the challenge step is as in Game 2.

We can see that B simulates the inputs to $\mathcal{D}$ statistically close, whenever B does not halt early. Since $\mathcal{D}$ has a nonnegligible chance of distinguishing Game 3 from Game 2, when B does not halt, B also has a non-negligible chance of breaking the the XDHV assumption. The chance that B does not halt is at least $1/2p$, so the overall advantage of B remains non-negligible.

□

□

10 Acknowledgements

We thank Allison Lewko for pointers to common assumptions in asymmetric bilinear map groups, and Stefano Tessaro for comments on the syntax.

References

[1] PBC library: The pairing-based cryptography library. http://crypto.stanford.edu/pbc/.

[2] Feng Bao, Robert H. Deng, Xuhua Ding, and Yanjiang Yang. Private query on encrypted data in multi-user settings. In *ISPEC*, pages 71–85, 2008.

[3] Mihir Bellare, Alexandra Boldyreva, and Adam O'Neill. Deterministic and efficiently searchable encryption. In *CRYPTO*, pages 535–552, 2007.

[4] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public key encryption with keyword search. In *EUROCRYPT'04*, pages 506–522, 2004.

[5] Dan Boneh and Brent Waters. Conjunctive, subset, and range queries on encrypted data. In *TCC*, pages 535–554, 2007.

[6] David Cash, Stanislaw Jarecki, Charanjit Jutla, Hugo Krawczyk, Marcel Rosu, and Michael Steiner. Highly-scalable searchable symmetric encryption with support for boolean queries. Cryptology ePrint Archive, Report 2013/169, 2013. http://eprint.iacr.org/.

[7] Yan-Cheng Chang and Michael Mitzenmacher. Privacy preserving keyword searches on remote encrypted data. In *ACNS*, pages 442–455, 2005.

[8] Reza Curtmola, Juan Garay, Seny Kamara, and Rafail Ostrovsky. Searchable symmetric encryption: improved definitions and efficient constructions. In *CCS*, pages 79–88, 2006.

[9] S. D. Galbraith, K. G. Paterson, and N. P. Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, 156, 2008.

[10] Eu-Jin Goh. Secure indexes. Cryptology ePrint Archive, Report 2003/216.

[11] Seny Kamara, Charalampos Papamanthou, and Tom Roeder. Dynamic searchable symmetric encryption. In *CCS*, pages 965–976, 2012.

[12] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In *STOC*, pages 1219–1234, 2012.

[13] Remya Rajan. Efficient and privacy preserving multi user keyword search for cloud storage services. In *IJATER*, pages 48–51, 2012.

[14] Dawn Xiaodong Song, David Wagner, and Adrian Perrig. Practical techniques for searches on encrypted data. In *Proceedings of the 21st IEEE Symposium on Security and Privacy*, Oakland, CA, May 2000.

[15] Yanjiang Yang, Haibing Lu, and Jian Weng. Multi-user private keyword search for cloud computing. In *CloudCom*, pages 264–271, 2011.

[16] Fangming Zhao, Takashi Nishide, and Kouichi Sakurai. Multi-user keyword search scheme for secure data sharing with fine-grained access control. In *ICISC*, pages 406–418, 2011.