Achieving Efficient Conjunctive Keyword Searches over Encrypted Data

Lucas Ballard, Seny Kamara, and Fabian Monrose

Department of Computer Science,
Johns Hopkins University,
Baltimore, MD, USA
{lucas, seny, fabian}@cs.jhu.edu

Abstract. We present two provably secure and efficient schemes for performing conjunctive keyword searches over symmetrically encrypted data. Our first scheme is based on Shamir Secret Sharing and provides the most efficient search technique in this context to date. Although the size of its trapdoors is linear in the number of documents being searched, we empirically show that this overhead remains reasonable in practice. Nonetheless, to address this limitation we provide an alternative based on bilinear pairings that yields constant size trapdoors. This latter construction is not only asymptotically more efficient than previous secure conjunctive keyword search schemes in the symmetric setting, but incurs significantly less storage overhead. Additionally, unlike most previous work, our constructions are proven secure in the standard model.

1 Introduction

Remote and untrusted storage systems [21,14,13] allow clients with limited resources to store and distribute large amounts of data at low cost. However, in order to preserve confidentiality, the remotely-stored data must be encrypted prior to transmission. Unfortunately, encryption restricts a client's ability to selectively access segments of her data, especially when she wishes to only retrieve specific content (e.g., related to a given keyword). To address this dilemma, a number of techniques have been recently proposed for achieving a less stringent storage model, one based on the notion of secure, delegated, searchable encryption (e.g., [28,17,12,10,29]). Intuitively, in order to provide secure searchable encryption schemes, most of these approaches associate an index with each document that, when combined with a trapdoor for a keyword, returns information signifying the association of the keyword with the document. Informally, such keyword searches are considered secure if they leak at most one bit of information about each document, namely, whether or not that document contains the keyword.

While the ability to perform single keyword searches is useful in some settings, clearly, it is more desirable to search on multiple keywords, and in particular, on boolean combinations of these keywords. Unfortunately, most previous

S. Qing et al. (Eds.): ICICS 2005, LNCS 3783, pp. 414–426, 2005.

suggestions for permitting multiple keyword searches either fail to do so efficiently, or leak unnecessary information. To see why, consider that the suggested approaches for achieving this goal have been to either (*i*) send a trapdoor for each keyword and expect the server to return the set intersection (in the case of conjunctions) or the set union (in the case of disjunctions) of the matching documents (e.g., see [28,17]), or (*ii*) store information corresponding to every possible boolean combination of keywords on the server. The former suggestion leaks information that is linear in the number of conjuncts being searched, and the latter, while secure, incurs storage overhead that is exponential in the number of keywords associated with the document.

In this paper, we focus on providing provably-secure conjunctive keyword search over symmetrically encrypted data, while minimizing the computational and storage overhead imposed on both the client and server. To this end, we present two constructions, one based on a non-standard use of Shamir Secret Sharing [27] and another based on bilinear pairings. We show that our first construction is the most *practical* conjunctive keyword search scheme known to date. Although the size of the trapdoors it generates is linear in the number of documents being searched, our experiments show that this overhead remains manageable in practice. For situations where constant sized trapdoors are required, we provide an alternative construction that is the most *asymptotically* efficient scheme we are aware of (in terms of both space and time complexity) in the symmetric setting. Moreover, both of our constructions are provably secure in the standard model.

2 Related Work

Song, Wagner, and Perrig introduced the notion of searchable encryption in [28]. In that work, the authors present a new encryption algorithm that embeds extra information into the ciphertext such that when it is used in conjucntion with a trapdoor, it discloses whether a particular keyword is stored in a document. Unfortunately, search requires computation linear in the size of each document and reveals statistical information about the distribution of the underlying plaintext.

Both of these shortcomings are addressed by the work of Goh [17], which presents a construction that uses per-document indexes derived from Bloom filters [7]. There, each word in the document is processed using a pseudo-random function and then inserted into a Bloom filter. The client then provides a trapdoor consisting of an indicator of which bits in the filter should be tested, thereby resulting in constant per-document search time. Moreover, Goh's work also introduced the notion of semantic security against chosen-keyword attacks (called IND-CKA), which is the first formal notion of security defined for searchable encryption.

As discussed earlier, neither of these schemes allow users to perform boolean keyword searches securely and efficiently. This shortcoming was first addressed by Golle, Staddon and Waters in [18], where they present two solutions that achieve the desired level of security. The first is provably secure under the De-

cision Diffie-Hellman assumption [8] and requires two modular exponentiations per document for searching. Additionally, the size of the trapdoors (referred to as search capabilities in [18]) is linear in the number of documents being searched. Although it is shown that portions of the trapdoors can be transmitted before a search even takes place, this overhead is still undesirable. Furthermore, it requires the client to know the number of searches she wishes to perform a priori. The second construction is based on bilinear pairings and is proven secure under a new hardness assumption. That scheme achieves constant sized trapdoors, but requires a linear number (with respect to keywords) of pairing computations per document in order to perform a search—an overhead that is arguably unrealistic, particularly in the presence of a large collection of documents.

More recently, Park, Kim and Lee proposed the first public-key searchable encryption schemes [10,29,15] that allow for secure conjunctive keyword searches [25]. Their constructions, based on the Bilinear Decision Diffie-Hellman (BDDH) and Bilinear Decision Diffie-Hellman Inversion (BDDHI) assumptions, have constant sized trapdoors and are more efficient than the schemes presented in [18]. Since the constructions presented in this paper are more efficient, in terms of computational and storage overhead, than both of the previous approaches, we argue that our schemes offer the most pragmatic choice in the symmetric setting.

3 Preliminaries

3.1 Model

We assume the standard model for symmetric searchable encryption schemes (as used in [28,17,12,18]), which includes a client and a server that can be trusted to interact in a protocol, but not to abstain from attempting to learn information that is not explicitly released by the client. The client has a set of m documents $\mathcal{D} = (D_1, \ldots, D_m)$ that she wishes to store on the server in encrypted form, while still retaining the ability to search through them. To do so, she first generates an index $\mathcal{I}_i$ for each document D_i and stores both the index and the encrypted document $\mathcal{E}(D_i)$ on the server. Here, we assume that $\mathcal{E}$ is an arbitrary symmetric encryption scheme, such as AES [16], and is independent of our index constructions. To search the document collection for a given keyword w, the client generates and sends a trapdoor $\mathcal{T}$ to the server who proceeds to search each index for w. The server then returns the appropriate set of documents to the client.

In this work, we also assume the standard model for conjunctive keyword searches over encrypted data as presented in [18]. Namely, we work in the setting where each document is associated with a list of keywords. In particular, we make the following assumptions: (i) the number of keywords associated with a document remains fixed and (ii) no keyword appears at two different locations in a list. The first constraint can be satisfied by simply adding null keywords to the list, while the second can be satisfied by prepending each keyword with a field name or the value of a counter. As in [18,25], to reduce computational burden, trapdoors specify which positions should be searched within an index.

3.2 Notation

Throughout this paper we use the following notation. Let Γ be a dictionary of words, and 2^{Γ} be the set of all possible documents. Further, let $\mathcal{D} \subseteq 2^{\Gamma}$ be a collection of m documents $\mathcal{D} = (D_1, \ldots, D_m)$. We associate a list of keywords $W_i = (w_{i,1}, \ldots w_{i,n})$ with each document D_i. When the associated document is clear from the context, we simplify the notation as $W_i = (w_1, \ldots, w_n)$. $\mathcal{I}_i$ refers to an index for a document D_i, and $\mathcal{T}_c$ to a trapdoor for a conjunction of d words $c = (w_1, \ldots, w_d)$. If a trapdoor is composed of one term for each document in $\mathcal{D}$ (i.e., is linear in the size of $\mathcal{D}$), then we say that $\mathcal{T}_c$ is composed of m tokens $(T_1, \ldots, T_m)$.

Furthermore, we write $x \xleftarrow{R} X$ to represent a random variable x being drawn uniformly from a set X. The output of a deterministic algorithm $\mathcal{A}$ will be denoted by $x \leftarrow \mathcal{A}$ and that of a probabilistic algorithm by $x \xleftarrow{R} \mathcal{A}$. Finally, let $\mathsf{negl}(k)$ denote a negligible function in k, and $||$ denote concatenation.

4 Definitions

In this section we reintroduce the definition of a secure conjunctive keyword search scheme and recall the notion of semantic security against chosen-keyword attacks.

Definition 1 (Secure Conjunctive Keyword Search (SCKS)). *Let $\mathcal{W} = (W_1, \ldots, W_m)$ be a collection of keyword lists. A SCKS scheme consists of four probabilistic polynomial-time algorithms:*

- $\mathsf{Keygen}(1^k)$*: is a probabilistic key generation algorithm that is executed by the client in order to instantiate the scheme. It takes as input a security parameter k, and returns a secret key K.*
- $\mathsf{BuildIndex}(K, W_i)$*: is executed by the client to construct an index. It takes as input a secret key K and a keyword list W_i. It returns W_i's index $\mathcal{I}_i$.*
- $\mathsf{Trapdoor}(K, \ell_1, \ldots, \ell_d, w_1, \ldots, w_d)$*: is executed by the client to generate a trapdoor for a given conjunction of keywords. It takes as input a secret key K, the locations in the index to search $(\ell_1, \ldots \ell_d)$, and a list of d conjuncts $(w_1, \ldots, w_d)$. It returns a trapdoor $\mathcal{T}_c$ for the conjunction $c = (w_1 \wedge \cdots \wedge w_d)$.*
- $\mathsf{SearchIndex}(\mathcal{I}_i, \mathcal{T}_c)$*: is executed by the server on behalf of the client to search for the occurrence of a conjunction in an index. It takes as input an index $\mathcal{I}_i = \mathsf{BuildIndex}(K, W_i)$, where $W_i = (w_1, \ldots, w_n)$, and a trapdoor $\mathcal{T}_c$ for a conjunction $c = (w'_1 \wedge \cdots \wedge w'_d)$. It returns* true *if $w'_j = w_{\ell_j}$ for $1 \leq j \leq d$; and* false *otherwise.*

Intuitively, the notion of security we seek to capture can be summarized as follows: given access to a set of indexes, a server should not be able to learn any partial information about the associated keyword lists that he cannot learn from a trapdoor that was *explicitly* given to him by the client. Note that in the context of conjunctive keyword searches, this implies that the trapdoor for a

given conjunction $c = (w_1 \wedge \cdots \wedge w_d)$ should not help the server in generating a trapdoor for any other conjunction $c' = (w'_1 \wedge \cdots \wedge w'_\delta)$, even if $\{w'_1, \ldots, w'_\delta\} \subset \{w_1, \ldots, w_d\}$. In addition, this notion of security should hold even against a server that can mount chosen keyword attacks, or in other words, against a server that can trick the client into generating trapdoors for keywords of its choice. More formally, this notion of security is known as semantic security against chosen keyword-attacks as introduced in [17]. Golle, Staddon and Waters [18] present three games that formally capture semantic security against chosen-keyword attacks for SCKS schemes and show that they are all asymptotically equivalent. In this work, we only make use of two of these games, namely *indistinguishability of ciphertext from ciphertext* and *indistinguishability of ciphertext from random*, which we briefly review below.

Definition 2 (Indistinguishability of ciphertext from ciphertext (ICC) [18]). *For all probabilistic polynomial-time adversaries $\mathcal{A}$:*

$$\Pr\left[\begin{array}{c} b' = b : K \leftarrow \mathsf{Keygen}(1^k); \\ (W_0, W_1) \leftarrow \mathcal{A}^{\mathsf{BuildIndex}(K,\cdot),\mathsf{Trapdoor}(K,\cdot)}; \\ b \stackrel{R}{\leftarrow} \{0,1\}; \mathcal{I}_b \leftarrow \mathsf{BuildIndex}(K, W_b); \\ b' \leftarrow \mathcal{A}^{\mathsf{BuildIndex}(K,\cdot),\mathsf{Trapdoor}(K,\cdot)}(\mathcal{I}_b) \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(k)$$

with the restriction that $\mathcal{A}$ must choose (W_0, W_1) such that $|W_0| = |W_1|$ and such that $\mathsf{SearchIndex}(\mathcal{I}_0, \mathcal{T}) = \mathsf{SearchIndex}(\mathcal{I}_1, \mathcal{T})$ *for all $\mathcal{T}$ generated using its* Trapdoor *oracle.*

Definition 3 (Indistinguishability of ciphertext from random (ICR) [18]). *For all probabilistic polynomial-time adversaries $\mathcal{A}$:*

$$\Pr\left[\begin{array}{c} b' = b : K \leftarrow \mathsf{Keygen}(1^k); \\ W_0 \leftarrow \mathcal{A}^{\mathsf{BuildIndex}(K,\cdot),\mathsf{Trapdoor}(K,\cdot)}; W_1 \stackrel{R}{\leftarrow} 2^\Gamma; \\ b \stackrel{R}{\leftarrow} \{0,1\}; \mathcal{I}_b \leftarrow \mathsf{BuildIndex}(K, W_b); \\ b' \leftarrow \mathcal{A}^{\mathsf{BuildIndex}(K,\cdot),\mathsf{Trapdoor}(K,\cdot)}(\mathcal{I}_b) \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(k)$$

with the restriction that W_1 must be chosen such that $|W_0| = |W_1|$ and such that $\mathsf{SearchIndex}(\mathcal{I}_0, \mathcal{T}) = \mathsf{SearchIndex}(\mathcal{I}_1, \mathcal{T})$ *for all $\mathcal{T}$ generated using its* Trapdoor *oracle.*

Theorem 1 ([18]). *If there exists an adversary $\mathcal{A}$ that wins game ICC with non-negligible probability, then there exists another adversary $\mathcal{B}$ that wins game ICR with the same probability.*

5 A Construction Based on Secret Sharing

In this section we describe our first secure conjunctive keyword search scheme, denoted as SCKS-SS. It is based on Shamir's threshold secret sharing scheme

(SSS) and is provably secure in the standard model. We note that this construction does not take advantage of SSS's threshold properties, which suggests that similar constructions could be achieved using random oracles or other secret sharing schemes. We chose SSS due to its universal familiarity and efficiency.

In [27], Shamir proposed a (k, n)-threshold secret sharing scheme based on polynomial interpolation in the field $\mathbb{Z}_p$. Informally, the scheme is composed of two algorithms $\mathsf{SSS} = (\mathsf{share}, \mathsf{recover})$ defined as follows. If $S \in \mathbb{Z}_p$ is a secret value we wish to share between n players, and if we require that at least k shares are needed to recover S, share operates as follows: a dealer generates a random $k-1$ degree polynomial $\mathcal{P}$ such that $\mathcal{P}(0) = S$; it then chooses n random points on $\mathcal{P}$ and distributes them as shares. In order to recover the secret $\mathcal{P}(0) = S$, the recover algorithm simply requires that at least k players pool their shares, perform standard polynomial interpolation, and evaluate the resulting polynomial at 0. The unconditional security of SSS follows from the fact that one cannot interpolate a $k-1$ degree polynomial with fewer than k points.

5.1 Our Construction

Our construction makes use of a pseudo-random function $f : \{0,1\}^k \times \{0,1\}^l \rightarrow \mathbb{Z}_p \times \mathbb{Z}_p$, where l is the length of the longest word in Γ. Each keyword list has an associated identifier, i, which is *never* reused. In particular, even if two lists have the same set of keywords at the same locations, their identifier will be different. Given a list W_i, we write $f_K^i(w)$, where $w \in W_i$, to refer to the following operation: $f(K, i||w)$. Let $\mathcal{W} = (W_1, \ldots, W_m)$ be a collection of m keyword lists, each composed of n words. $\mathsf{SCKS\text{-}SS} = (\mathsf{Keygen}, \mathsf{BuildIndex}, \mathsf{Trapdoor}, \mathsf{SearchIndex})$ is then given as follows:

- $\mathsf{Keygen}(1^k)$: Generate a secret key $K \xleftarrow{R} \{0,1\}^k$ and a random prime $p > n$.
- $\mathsf{BuildIndex}(K, W_i)$: For each word $w_j \in W_i$, let $\sigma_j = f_K^i(w_j) = (x_j, y_j)$. Output $\mathcal{I}_i = (\sigma_1, \ldots, \sigma_n)$
- $\mathsf{Trapdoor}(K, \ell_1, \ldots, \ell_d, w'_1, \ldots, w'_d)$: For each $W_i \in \mathcal{W}$, let $S_i = \mathsf{recover}(f_K^i(w'_1), \ldots, f_K^i(w'_d))$. Output $\mathcal{T} = (S_1, \ldots, S_m, \ell_1, \ldots, \ell_d)$.
- $\mathsf{SearchIndex}(\mathcal{I}_i, \mathcal{T})$: If $S_i = \mathsf{recover}(\sigma_{\ell_1}, \ldots, \sigma_{\ell_d})$, then output true, otherwise output false.

5.2 Security

Since correctness follows from the description of the scheme, we provide only a proof of security. To show that SCKS-SS is semantically secure against chosen-keyword attacks, we first state two useful lemmas. Due to space considerations, we omit the proofs of these lemmas, but refer the reader to the full version of this paper [3]. The first states that given two arbitrary keyword lists, their indexes are independent to any probabilistic polynomial-time adversary. We note that this holds even if the lists are composed of the same keywords. The second lemma states that given two different conjunctions, their corresponding trapdoors will

be indepedent (also to any probabilistic polynomial-time adversary), even if they are generated in order to search the same keyword list.

The usefulness of these lemmas will become apparent in our main theorem (Theorem 2), where we claim that since the adversary cannot learn anything about its challenge index from access to its BuildIndex and Trapdoor oracles, we need only consider semantic security against chosen plaintext attacks.

Lemma 1. *Let W_a and W_b be two keyword lists such that $a \neq b$. If f_K is a pseudo-random function, then $\mathcal{I}_a$ is independent of $\mathcal{I}_b$ for all probabilistic polynomial-time adversaries.*

Lemma 2. *Let $\mathcal{W} = (W_1, \ldots, W_m)$ be a collection of m keyword lists and let $c = (w_1 \wedge \cdots \wedge w_d)$ and $c' = (w'_1 \wedge \cdots \wedge w'_\delta)$ be two conjunctions of length d and δ, respectively. If f_K is a pseudo-random function and if $c \neq c'$, then for any probabilistic polynomial-time adversary $\mathcal{A}$, $\mathcal{T}_c$ is independent of $\mathcal{T}_{c'}$.*

Theorem 2. *If f_K is a pseudo-random function, then* SCKS-SS *is semantically secure against chosen-keyword attacks.*

Proof. Since the identifier of a keyword list is never repeated, we know from Lemma 1 that if f_K is a pseudo-random function then no two indexes will be correlated even if they contain the same keywords. It follows that $\mathcal{A}$ cannot learn anything about its ICC challenge index $\mathcal{I}_b$ from any of the indexes returned by its BuildIndex oracle.

Furthermore, consider the restriction imposed on $\mathcal{A}$'s choice of keyword lists in the ICC game, namely that it must choose two lists W_0 and W_1, such that $\mathsf{SearchIndex}(\mathcal{I}_0, \mathcal{T}) = \mathsf{SearchIndex}(\mathcal{I}_1, \mathcal{T})$ for all trapdoors $\mathcal{T}$ returned by its Trapdoor oracle. This implies that any such trapdoor will be useless to $\mathcal{A}$ in distinguishing whether $\mathcal{I}_b$ is the index for W_0 or W_1. In addition, by Lemma 2, we know that if f_K is a pseudo-random function, then trapdoors generated for different conjunctions are independent. Taken together, the two preceding statements imply that $\mathcal{A}$ cannot use the results of its Trapdoor queries to either search over its challenge index $\mathcal{I}_b$, or to generate any new trapdoors with which it can try to search over $\mathcal{I}_b$.

From the previous discussion, it is then safe to only consider $\mathcal{A}$'s challenge index. If f is a random function, then $\mathcal{I}_0$ and $\mathcal{I}_1$ are independent of W_0 and W_1, respectively. If we replace f by a pseudo-random function f_K, then to any probabilistic polynomial-time adversary, $\mathcal{I}_b$ will be independent of its associated keyword list W_b. Thus $\mathcal{A}$ will not be able to distinguish between W_0 and W_1 given $\mathcal{I}_b$, otherwise we could build an adversary $\mathcal{B}$ that could distinguish between f_K and a random function. □

6 A Construction Based on Bilinear Maps

Our second construction, SCKS-XDH, achieves constant transmission overhead at the cost of placing a larger computational burden on the server. The security

of the scheme is based on a new variant of the External Diffie-Hellman (XDH) assumption. The XDH assumption was first introduced in [26] and formalized in [9,2]. It has been used more recently as a hardness assumption in [**?**,11]

Assumption 1 (Decision Diffie-Hellman (DDH)). *Let $\langle g \rangle = \mathbb{G}$ be a cyclic group of order p and $c \xleftarrow{R} \mathbb{Z}_p$. The Decision Diffie-Hellman assumption holds in $\mathbb{G}$ if no probabilistic polynomial-time adversary $\mathcal{A}$ can distinguish between tuples (g, g^a, g^b, g^{ab}) and (g, g^a, g^b, g^c), with probability non-negligibly greater than $\frac{1}{2}$:*

$$\left|\Pr\left[\mathcal{A}(g^a, g^b, g^{ab}) = 1\right] - \Pr\left[\mathcal{A}(g^a, g^b, g^c) = 1\right]\right| \leq \frac{1}{2} + \mathsf{negl}(|p|)$$

Assumption 2 (External Diffie-Hellman (XDH) [2]). *Let $\langle P \rangle = \mathbb{G}_1$ and $\langle Q \rangle = \mathbb{G}_2$ be two disjoint cyclic subgroups of order p of an elliptic curve, and let $\hat{e}$ be a non-degenerate bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$. The XDH assumption holds across $\mathbb{G}_1$ and $\mathbb{G}_2$ if the DDH assumption holds within $\mathbb{G}_1$.*

A concrete algebraic setting where this assumption holds is discussed further in [2]. To prove the security of our construction, we make use of a slightly stronger variant of the XDH assumption which we call the *Mixed XDH assumption.*

Assumption 3 (Mixed External Diffie-Hellman (MXDH)). *Let $\langle P \rangle = \mathbb{G}_1$ and $\langle Q \rangle = \mathbb{G}_2$ be two groups of order p, and let $\hat{e}$ be a non-degenerate bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$. The MXDH assumption holds across $\mathbb{G}_1$ and $\mathbb{G}_2$ if given (P, aP, bP, cP, aQ, bQ), no probabilistic polynomial-time adversary $\mathcal{A}$ can distinguish between tuples (P, aP, bP, abP) and (P, aP, bP, cP), where $c \xleftarrow{R} \mathbb{Z}_p$ (i.e., the DDH assumption holds within $\mathbb{G}_1$).*

6.1 Our Construction

Let $\langle P \rangle = \mathbb{G}_1$ and $\langle Q \rangle = \mathbb{G}_2$ be two groups of prime order p and $\hat{e}$ be a non-degenerate bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ such that the XDH assumption holds accross $\mathbb{G}_1$ and $\mathbb{G}_2$. Additionally, let $f : \{0,1\}^k \times \{0,1\}^l \rightarrow \mathbb{Z}_p$ be a pseudo-random function, where l is the length of the longest word in Γ. Recall that in our model we assume that all keywords are distinct and that this can be achieved by simply concatenating the value of a counter to each keyword. We define $\mathsf{SCKS\text{-}XDH} = (\mathsf{Keygen}, \mathsf{BuildIndex}, \mathsf{Trapdoor}, \mathsf{SearchIndex})$ as follows:

- $\mathsf{Keygen}(1^k)$: Generate a secret key $K \xleftarrow{R} \{0,1\}^k$, and choose two points P and Q such that $\langle P \rangle = \mathbb{G}_1$ and $\langle Q \rangle = \mathbb{G}_2$. Q is kept private.
- $\mathsf{BuildIndex}(K, W_i)$: Choose $r_i \xleftarrow{R} \mathbb{Z}_p$. For each word $w_j \in W_i$, let $s_j = f_K(w_j)$. Output $\mathcal{I}_i = (r_i P, r_i s_1 P, \ldots, r_i s_n P)$.
- $\mathsf{Trapdoor}(K, \ell_1, \ldots, \ell_d, w'_1, \ldots, w'_d)$: Choose $\rho \xleftarrow{R} \mathbb{Z}_p$. Let $t = \left(\rho \sum_{j=1}^{d} f_K(w'_j)\right) Q$. Output $\mathcal{T} = (t, \rho Q, \ell_1, \ldots, \ell_d)$.
- $\mathsf{SearchIndex}(\mathcal{I}_i, \mathcal{T})$: If $\hat{e}(t, r_i P) = \hat{e}(\rho Q, \sum_{j=1}^{d} r_i s_{\ell_j} P)$, then output true, otherwise output false.

To demonstrate correctness, consider a user that wishes to search for a conjunction $c = (w'_1 \wedge \cdots \wedge w'_d)$ over the index locations specified by the sequence of integers $(\ell_1, \ldots, \ell_d)$. Let $\mathcal{T} = \mathsf{Trapdoor}(K, \ell_1, \ldots, \ell_d, w'_1, \ldots, w'_d)$ be the trapdoor enabling the server to search for c. Furthermore, let $W_i = (w_1, \ldots, w_n)$ be a keyword list, and $\mathcal{I}_i = \mathsf{BuildIndex}(K, W_i)$ be its index. If W_i is such that $w_{\ell_j} = w'_j$ for $1 \leq j \leq d$ (i.e. W_i includes all the words in the conjunction c at the appropriate locations) then: $\hat{e}(t, r_i P) = \hat{e}((\rho \sum_{j=1}^{d} f_K(w'_j))Q, r_i P) = \hat{e}(Q, P)^{r_i \rho \sum_{j=1}^{d} f_K(w'_j)} = \hat{e}(Q, P)^{\rho \sum_{j=1}^{d} r_i s_{\ell_j}} = \hat{e}(\rho Q, \sum_{j=1}^{d} r_i s_{\ell_j} P)$ and $\mathsf{SearchIndex}(\mathcal{I}_i, \mathcal{T})$ will return true.

6.2 Security

Theorem 3. *If the Mixed XDH assumption holds, then* SCKS-XDH *is semantically secure against chosen-keyword attacks.*

Proof. Given an adversary $\mathcal{A}$ that wins the ICR game with non-negligible probability over $\frac{1}{2}$, we describe an adversary $\mathcal{B}$ that uses $\mathcal{A}$ to break the MXDH assumption with the same probability. By theorem 1 this implies that there exists another adversary that can win game ICC also with non-negligible probability over $\frac{1}{2}$. Let $\langle P \rangle = \mathbb{G}_1$, $\langle Q \rangle = \mathbb{G}_2$, and $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be an appropriate XDH setting, and let (P, aP, bP, cP, aQ, bQ) be $\mathcal{B}$'s MXDH challenge. $\mathcal{B}$'s goal is to solve the DDH problem in $\mathbb{G}_1$, or in other words to decide whether $c = ab$.

$\mathcal{B}$ begins by simulating $\mathcal{A}$. To start, $\mathcal{A}$ will make a polynomial number of $\mathsf{BuildIndex}$ queries which $\mathcal{B}$ will answer as follows. Let $\mathcal{A}$'s query keyword list be $W_i = (w_{i,1} \wedge \cdots \wedge w_{i,n})$, where $0 \leq i \leq \mathrm{poly}(k)$. $\mathcal{B}$ chooses a value $r_i \xleftarrow{R} \mathbb{Z}_p$ and for each word $w_{i,j} \in W_i$, where $1 \leq j \leq n$, it picks a random value $z_{i,j} \xleftarrow{R} \mathbb{Z}_p$ and computes $r_i z_{i,j} bP$. To be consistent across different queries, $\mathcal{B}$ keeps track of the correspondence between keywords $w_{i,j} \in W_i$ and the random $z_{i,j}$ values it chooses. Notice that if $\mathcal{B}$ is given bP as part of its MXDH challenge, it can compute $r_i z_{i,j} bP$. Finally, it returns the index $\mathcal{I}_i = (r_i P, r_i z_{i,1} bP, \ldots, r_i z_{i,n} bP)$ to $\mathcal{A}$.

When $\mathcal{A}$ makes a $\mathsf{Trapdoor}$ query with index locations $(\ell_1, \ldots, \ell_d)$ and conjunction $c = (w'_1 \wedge \cdots \wedge w'_d)$, $\mathcal{B}$ begins by choosing a random value $\rho \xleftarrow{R} \mathbb{Z}_p$. It then computes $t = \rho(\sum_{\lambda=1}^{d} \gamma_\lambda bQ)$ where $\gamma_\lambda = z_{i,j}$ if w'_λ previously appeared at some position in one of $\mathcal{A}$'s queries, and $\gamma_\lambda \xleftarrow{R} \mathbb{Z}_p$ otherwise. Observe that while $\mathcal{B}$ does not know b, it can compute $z_{i,j} bQ$ (and thus t), since bQ is part of its MXDH challenge.

Finally, $\mathcal{B}$ returns the trapdoor $\mathcal{T} = (t, \rho Q, \ell_1, \ldots, \ell_d)$ to $\mathcal{A}$. Note that $\mathcal{T}$ is a valid trapdoor for $c = (w'_1 \wedge \ldots, \wedge w'_n)$, and in particular, that since $\mathcal{B}$ consistently uses the same value $z_{i,j}$ for word w_j, $\mathsf{SearchIndex}(\mathcal{I}_i, \mathcal{T})$ will return true if and only if W_i includes all the words in the conjunction c at the locations specified by $(\ell_1, \ldots, \ell_d)$.

After polynomially many $\mathsf{BuildIndex}$ and $\mathsf{Trapdoor}$ queries, $\mathcal{A}$ submits a keyword list $W^* = (w^*_1, \ldots, w^*_n)$. $\mathcal{B}$ returns to $\mathcal{A}$ the challenge index $\mathcal{I}^* = (aP, \gamma^*_1 cP, \ldots, \gamma^*_n cP)$, where $\gamma^*_\lambda = z_{i,j}$ if w^*_λ previously appeared in one of $\mathcal{A}$'s queries to

either its BuildIndex or Trapdoor oracles, and $\gamma_\lambda^* \stackrel{R}{\leftarrow} \mathbb{Z}_p$ otherwise. Observe that if $c = ab$, then $\mathcal{I}^*$ is a correct index for W^*, while if $c \neq ab$ then $\mathcal{I}^*$ is a correct index for some other arbitrary keyword list. In particular, if c is random then I^* is an index for a random set of keywords. After the challenge, $\mathcal{A}$ is allowed to make more Trapdoor and BuildIndex queries (with the same restrictions as before), which $\mathcal{B}$ answers as it did in the previous steps.

Finally, $\mathcal{A}$ outputs a bit β that represents its decision as to whether I^* is an index for W^* or some random keyword list. $\mathcal{B}$ then returns β as its own answer to its MXDH challenge. It follows that $\mathcal{B}$'s probability in breaking its MXDH challenge is equal to $\mathcal{A}$'s probability in breaking the ICR game, which we assumed holds with non-negligible probability over $\frac{1}{2}$. □

7 Efficiency

Comparison with Previous Constructions. We now consider the computational and space complexity of our constructions. In particular, we compare their efficiency to the work of Park, Kim and Lee [25]. We note that while the schemes in [25] are in the public-key setting, they are still more efficient than previous symmetric constructions presented in [18]. Since any public-key searchable encryption scheme can be converted to a symmetric one, we focus our comparison with the schemes in [25].

We refer to the first and second constructions in [25] as PKL-1 and PKL-2, respectively. Recall that m denotes the number of documents stored on the server, n the number of keywords associated with each document, and d the number of keywords comprising a search. Unless otherwise specified, measurements are in terms of the requirements to process *all* documents on the server.

In what follows we discuss the running time and storage overhead for all relevant operations, including building indexes, generating trapdoors, searching and storing indexes, and sending trapdoors. First, we note that SCKS-SS is the most computationally efficient construction for index generation and searching, requiring only m polynomial interpolations on d points. Furthermore, though both the size of its trapdoors and the running time of the trapdoor generation algorithm are linear in the number of documents, we show that this overhead still remains practical. Storage requirements for indexes are less than previous approaches [25,18], requiring only $2mn$ points in $\mathbb{Z}_p$ (where p is only 128 bits). We also note that SCKS-XDH incurs significantly less storage and transmission overhead than PKL-1 and PKL-2 as it need not store or send elements in integer groups. Additionally, SCKS-XDH is more efficient than both PKL-1 and PKL-2 for BuildIndex as it requires only $m(n+1)$ multiplications in $\mathbb{G}_1$, is faster than PKL-1 for trapdoor generation, and is comparable to PKL-2 (but slower than PKL-1) for SearchIndex.

Empirical Evaluation of SCKS-SS and SCKS-XDH. To evaluate the feasibility of our constructions, we implemented and benchmarked the relevant operations. All tests were performed on a 3.0 GHz Pentium 4 machine running Fedora

Core 3 Linux. Our implementations are in C++ and made use of the MIRACL [24] library for multi-precision and curve-based operations, and OpenSSL [23] for all other cryptographic primitives.

Our implementation of SCKS-SS performs all operations in the field $\mathbb{Z}_p$, where p is a 128-bit prime. We chose to instantiate f using HMAC-SHA1 [6]. In particular, we let $f_K^i(w) = (h_K(w||i||0), h_K(w||i||1))$ where $h_K(\cdot)$ denotes the last 128 bits of HMAC-SHA1's output under the key K. And though polynomial interpolation can be done as efficiently as $O(d \log^2 d)$ [20], we use standard Lagrangian interpolation [20], which is $O(d^2)$ to generate trapdoors and perform searches. Our implementation of SCKS-XDH uses a 160 bit curve, which gives security comparable to that of Diffie-Hellman in 1024 bit integer groups [22]. Due to the algebraic setting of the MXDH assumption, we can represent points in $\mathbb{G}_1$ using 161 bits, but require 481 bits for points in $\mathbb{G}_2$ [5]. Accordingly, operations in $\mathbb{G}_2$ are slower than in $\mathbb{G}_1$[1].

To evaluate the efficacy of our constructions, we present the time required to create 10,000 indexes of 10 keywords each, to generate trapdoors, and to search these indexes. In both constructions index generation grows linearly in the number of keywords. SCKS-SS requires slightly less than 2 seconds to generate $10,000$ indexes with 10 keywords each, while SCKS-XDH requires 445 seconds.

The SCKS-SS operations that require interpolation, namely Trapdoor and SearchIndex, incur time that is quadratic in the number of keywords being searched. We note, however, that according to [19], user queries on the Web typically contain at most 3 keywords. If we assume such a setting, SCKS-SS is able to search $10,000$ files in about 0.86 seconds, while trapdoor generation requires less than 1.5 seconds. The time required for SCKS-XDH to generate trapdoors and search a given index is essentially constant. Trapdoor generation requires 111 ms while searching 10,000 indexes requires approximately 720 seconds.

Although SCKS-SS requires trapdoors linear in the number of documents, since each token is only 16 bytes long, a trapdoor for $10,000$ documents is merely 156 KB in size. The space required to store the indexes associated with a collection of $10,000$ documents of 10 keywords each is 3.1 MB—which is much more space efficient than any previously known construction.

Although SCKS-XDH is less efficient in terms of index generation and searching than SCKS-SS, it requires less storage and only incurs constant transmission overhead. In fact, to store an index, the server need only need keep a point in $\mathbb{G}_1$ for each keyword of each document. As such, the indexes associated to a collection of $10,000$ documents with 10 keywords each can be stored in approximately 2.1 MB. Also, since trapdoors are pairs of points in $\mathbb{G}_2$ they can be represented in 0.12 KB.

Acknowledgements. The authors thank Avi Rubin for fruitful discussions and Eu-Jin Goh for comments on an earlier draft of this paper. This work was supported in part by a Bell Labs Graduate Research Fellowship and NSF award 0430338.

[1] Although SCKS-XDH could benefit from reusing line function coefficients in Miller's algorithm [4] (as suggested in [25]) we did not implement this optimization.

References

1. G. Ateniese, J. Camenisch, and B. de Medeiros. Untraceable rfid tags via insubvertible encryption. In *12th ACM Conference on Computer and Communications Security (CCS 2005)*, 2005.
2. L. Ballard, M. Green, B. de Medeiros, and F. Monrose. Correlation resistant storage. Technical Report TR-SP-BGMM-050507, Johns Hopkins University Department of Computer Science, 2005.
3. L. Ballard, S. Kamara, and F. Monrose. Achieving efficient conjunctive keyword searches over encrypted data. Technical Report TR-SP-BKM-050920, Johns Hopkins University Department of Computer Science, 2005.
4. P. Barreto, H. Kim, B. Lynn, and M. Scott. Efficient algorithms for pairing-based cryptosystems. In *Advances in Cryptology - CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 354–369. Springer-Verlag, 2002.
5. P. Barreto, B. Lynn, and M. Scott. On the selection of pairing-friendly groups. In *Selected Areas in Cryptography - SAC '03*, volume 3006 of *Lecture Notes in Computer Science*, pages 17–25. Springer-Verlag, 2004.
6. M. Bellare, R. Canetti, and H. Krawczyk. Keying hash functions for message authentication. In *Advances in Cryptology - CRYPTO 1996*, volume 1109 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 1996.
7. B. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970.
8. D. Boneh. The Decision-Diffie Hellman Problem. In *Third International Symposium on Algorithmic Number Theory (ANTS-III)*, volume 1423 of *Lecture Notes in Computer Science*, pages 48–63. Springer-Verlag, 1998.
9. D. Boneh, X. Boyen, and H. Sacham. Short group signatures. In *Advances in Cryptology - CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 41–55. Springer-Verlag, 2004.
10. D. Boneh, G. di Crescenzo, R. Ostrovsky, and G. Persiano. Public key encryption with keyword search. In *Advances in Cryptology - Eurocrypt 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 506–522. Springer-Verlag, 2004.
11. Jan Camenisch, Susan Hohenberger, and Anna Lysyanskaya. Compact e-cash. In *Advances in Cryptology – EUROCRYPT 2005*, Lecture Notes in Computer Science, pages 302–321. Springer, 2005.
12. Y. Chang and M. Mitzenmacher. Privacy preserving keyword searches on remote encrypted data. In *Third International Conference on Applied Cryptography and Network Security (ACNS 2005)*, volume 3531 of *Lecture Notes in Computer Science*, pages 442–455. Springer-Verlag, 2005.
13. I. Clarke, O. Sandberg, B. Wiley, and T. Hong. Freenet: a distributed anonymous information storage and retrieval system. In *Workshop on Design Issues in Anonymity and Unobservability*, volume 2009 of *Lecture Notes in Computer Science*, pages 46–66, 2001.
14. Microsoft Corp. Federated, available and reliable storage for an incompletely trusted environment (Farsite). See `http://research.` microsoft.com/sn/Farsite/.
15. D. Davis, F. Monrose, and M. Reiter. Time-scoped searching of encrypted audit logs. In *6th International Conference on Information and Communications Security (ICICS 2004)*, volume 3269 of *Lecture Notes in Computer Science*, pages 532–545. Springer-Verlag, 2004.
16. Federal Information Processing Standards. Advanced Encryption Standard (AES) – FIPS 197, November 2001.

17. E-J. Goh. Secure indexes. Technical Report 2003/216, IACR ePrint Cryptography Archive, 2003. See http://eprint.iacr.org/2003/216.
18. P. Golle, J. Staddon, and B. Waters. Secure conjunctive keyword search over encrypted data. In *Applied Cryptography and Network Security Conference (ACNS)*, volume 3089 of *Lecture Notes in Computer Science*, pages 31–45. Springer, 2004.
19. B. Jansen, A. Spink, J. Bateman, and T. Saracevic. Real life information retrieval: a study of user queries on the web. *SIGIR Forum*, 32(1):5–17, 1998.
20. D. Knuth. *The Art of Computer Programming*, volume 2. Addison-Wesley, Reading, Mass., 2nd edition, 1981.
21. J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, C. Wells, and B. Zhao. Oceanstore: an architecture for global-scale persistent storage. In *Ninth international conference on Architectural support for programming languages and operating systems*, pages 190–201. ACM Press, 2000.
22. A. Lenstra and E. Verheul. Selecting cryptographic key sizes. In *3rd International Workshop on Practice and Theory in Public Key Cryptography (PKC 2000)*, volume 1751 of *Lecture Notes in Computer Science*, pages 446–465. Springer-Verlag, 2000.
23. The OpenSSL Library. See http://www.openssl.org.
24. Shamus Software Ltd. Multiprecision integer and rational arithmetic C/C++ library (MIRACL). See http://indigo.ie/∼mscott.
25. D. Park, K. Kim, and P. Lee. Public key encryption with conjunctive field keyword search. In *5th International Workshop on Information Security Applications (WISA 2004)*, volume 3325 of *Lecture Notes in Computer Science*, pages 73–86. Springer-Verlag, 2004.
26. M. Scott. Authenticated ID-based key exchange and remote log-in with simple to ken and PIN number. Technical Report 2002/164, International Association for Cryptological Research, 2002. Available at http://eprint.iacr.org/2002/164.
27. A. Shamir. How to share a secret. *Communications of the ACM*, 22:612–613, November 1979.
28. D. Song, D. Wagner, and A. Perrig. Practical techniques for searching on encrypted data. In *IEEE Symposium on Research in Security and Privacy*, pages 44–55. IEEE Computer Society, May 2000.
29. B. Waters, D. Balfanz, G. Durfee, and D. Smetters. Building an encrypted and searchable audit log. In *Network and Distributed System Security Symposium (NDSS 2004)*. The Internet Society, 2004.