Practical Techniques for Searches on Encrypted Data*

Dawn Xiaodong Song David Wagner Adrian Perrig

{dawnsong, daw, perrig}@cs.berkeley.edu
University of California, Berkeley

Abstract

It is desirable to store data on data storage servers such as mail servers and file servers in encrypted form to reduce security and privacy risks. But this usually implies that one has to sacrifice functionality for security. For example, if a client wishes to retrieve only documents containing certain words, it was not previously known how to let the data storage server perform the search and answer the query without loss of data confidentiality.

In this paper, we describe our cryptographic schemes for the problem of searching on encrypted data and provide proofs of security for the resulting crypto systems. Our techniques have a number of crucial advantages. They are provably secure*: they provide* provable secrecy *for encryption, in the sense that the untrusted server cannot learn anything about the plaintext when only given the ciphertext; they provide* query isolation *for searches, meaning that the untrusted server cannot learn anything more about the plaintext than the search result; they provide* controlled searching*, so that the untrusted server cannot search for an arbitrary word without the user's authorization; they also support* hidden queries*, so that the user may ask the untrusted server to search for a secret word without revealing the word to the server. The algorithms we present are simple, fast (for a document of length n, the encryption and search algorithms only need $O(n)$ stream cipher and block cipher operations), and introduce almost no space and communication overhead, and hence are practical to use today.*

*We gratefully acknowledge support for this research from several US government agencies. This research was suported in part by the Defense Advanced Research Projects Agency under DARPA contract N6601-99-28913 (under supervision of the Space and Naval Warfare Systems Center San Diego), by the National Science foundation under grant FD99-79852, and by the United States Postal Service under grant USPS 1025 90-98-C-3513. Views and conclusions contained in this document are those of the authors and do not necessarily represent the official opinion or policies, either expressed or implied of the US government or any of its agencies, DARPA, NSF, USPS.

1 Introduction

Today's mail servers such as IMAP servers [11], file servers and other data storage servers typically must be fully trusted—they have access to the data, and hence must be trusted not to reveal it without authorization—which introduces undesirable security and privacy risks in applications. Previous work shows how to build encrypted file systems and secure mail servers, but typically one must sacrifice functionality to ensure security. The fundamental problem is that moving the computation to the data storage seems very difficult when the data is encrypted, and many computation problems over encrypted data previously had no practical solutions.

In this paper, we show how to support searching functionality without any loss of data confidentiality. An example is where a mobile user with limited bandwidth wants to retrieve all email containing the word "Urgent" from an untrusted mail-storage server in the infrastructure. This is trivial to do when the server knows the content of the data, but how can we support search queries if we do not wish to reveal all our email to the server?

Our answer is to present cryptographic schemes that enable searching on encrypted data without leaking any information to the untrusted server.

- Our techniques are *provably secure*. The techniques provide *provable secrecy* for encryption, in the sense that the untrusted server cannot learn anything about the plaintext given only the ciphertext. The techniques provide *controlled searching*, so that the untrusted server cannot search for a word without the user's authorization. The techniques support *hidden queries*, so that the user may ask the untrusted server to search for a secret word without revealing the word to the server. The techniques also support *query isolation*, meaning that the untrusted server learns nothing more than the search result about the plaintext.

- Our schemes are efficient and practical. The algorithms we present are simple and fast. More specifically, for a document of length n, the encryption and search algorithms only need $O(n)$ number of stream cipher and block cipher operations. Our schemes introduce essentially no space and communication overhead. They are also flexible and can be easily extended to support more advanced searches.

Our schemes all take the form of *probabilistic* searching: a search for the word W returns all the positions where W occurs in the plaintext, as well as possibly some other erroneous positions. We may control the number of errors by adjusting a parameter m in the encryption algorithm; each wrong position will be returned with probability about $1/2^m$, so for a ℓ-word document, we expect to see about $\ell/2^m$ false matches. The user will be able to eliminate all the false matches (by decrypting), so in remote searching applications, false matches should not be a problem so long as they are not so common that they overwhelm the communication channel between the user and the server.

This paper is structured as follows. We first introduce the problem of searching on encrypted data in Section 2 and briefly review some important background in Section 3. We then describe our solution for the case of searching with sequential scan in Section 4. We discuss further issues such as advanced search and search with index in Section 5. We discuss related work in Section 6 and finally we conclude in Section 7. Appendix A presents the proofs for all of proofs of security for these schemes.

2 Searching on Encrypted Data

We first define the problem of searching on encrypted data.

Assume Alice has a set of documents and stores them on an untrusted server Bob. For example, Alice could be a mobile user who stores her email messages on an untrusted mail server. Because Bob is untrusted, Alice wishes to encrypt her documents and only store the ciphertext on Bob. Each document can be divided up into 'words'. Each 'word' may be any token; it may be a 64-bit block, an English word, a sentence, or some other atomic quantity, according to the application domain of interest. For simplicity, we typically assume these 'words' have the same length (otherwise we can either pad the shorter 'words' or split longer 'words' to make all the 'words' to have equal length, or use some simple extensions for variable length 'words'; see also Section 5.3). Because Alice may have only a low-bandwidth network connection to the server Bob, she wishes to only retrieve the documents which contain the word W. In order to achieve this goal, we need to design a scheme so that after performing certain computations over the ciphertext, Bob can determine with some probability whether each document contains the word W without learning anything else.

There seem to be two types of approaches. One possibility is to build up an index that, for each word W of interest, lists the documents that contain W. An alternative is to perform a sequential scan without an index. The advantage of using an index is that it may be faster than the sequential scan when the documents are large. The disadvantage of using an index is that storing and updating the index can be of substantial overhead. So the approach of using an index is more suitable for mostly-read-only data.

We first describe our scheme for searching on encrypted data without an index. Since the index-based schemes seem to require less sophisticated constructions, we will defer discussion of searching with an index until the end of the paper (see Section 5.4).

3 Background and Definitions

Our scheme requires several fundamental primitives from classical symmetric-key cryptography. Because we will prove our scheme secure, we use only primitives with a well-defined notion of security. We will list here the required primitives, as well as reviewing the standard definitions of security for them. The definitions may be skipped on first reading for those uninterested in our theoretical proofs of security.

We adopt the standard definitions of security from the provable security literature [2], and we measure the strength of the cryptographic primitives in terms of the resources needed to break them. We will say that an attack R-breaks a cryptographic primitive if the attack algorithm succeeds in breaking the primitive with resources specified by R, and we say that a crypto primitive is R-secure if there is no algorithm that can R-break it. Let $A : \{0,1\}^n \to \{0,1\}$ be an arbitrary algorithm and let X and Y be random variables distributed on $\{0,1\}^n$. The *distinguishing probability* of A—sometimes called the *advantage* of A—for X and Y is

$$\mathrm{Adv}\, A = |\Pr[A(X) = 1] - \Pr[A(Y) = 1]|.$$

With this background, our list of required primitives is as follows:

1. A *pseudorandom generator* G, i.e., a stream cipher. We say that $G : \mathcal{K}_G \to S$ is a (t, e)-secure pseudorandom generator if every algorithm A with running time at most t has advantage $\mathrm{Adv}\, A < e$. The advantage of an adversary A is defined as $\mathrm{Adv}\, A = |\Pr[A(G(U_{\mathcal{K}_G})) = 1] - \Pr[A(U_S) = 1]|$, where $U_{\mathcal{K}_G}, U_S$ are random variables distributed uniformly on $\mathcal{K}_G, S$.

2. A *pseudorandom function* F. We say that $F : \mathcal{K}_F \times \mathcal{X} \to \mathcal{Y}$ is a (t, q, e)-secure pseudorandom function if every oracle algorithm A making at most q oracle

queries and with running time at most t has advantage $\mathrm{Adv}\, A < e$. The advantage is defined as $\mathrm{Adv}\, A = |\Pr[A^{F_k} = 1] - \Pr[A^R = 1]|$ where R represents a random function selected uniformly from the set of all maps from $\mathcal{X}$ to $\mathcal{Y}$, and where the probabilities are taken over the choice of k and R.

3. A *pseudorandom permutation* E, i.e., a block cipher. We say that $E : \mathcal{K}_E \times \mathcal{Z} \to \mathcal{Z}$ is a (t, q, e)-secure pseudorandom function if every oracle algorithm A making at most q oracle queries and with running time at most t has advantage $\mathrm{Adv}\, A < e$. The advantage is defined as $\mathrm{Adv}\, A = |\Pr[A^{E_k, E_k^{-1}} = 1] - \Pr[A^{\pi, \pi^{-1}} = 1]|$ where π represents a random permutation selected uniformly from the set of all bijections on $\mathcal{Z}$, and where the probabilities are taken over the choice of k and π. Notice that the adversary is given an oracle for encryption as well as for decryption; this corresponds to the adaptive chosen-plaintext/ciphertext attack model.

In general, the intuition is that (t, q, e)-security represents resistance to attacks that use at most t offline work and at most q adaptive chosen-text queries.

There is of course no fundamental need for three separate primitives, since in practice all three may be built out of just one off-the-shelf primitive. For instance, given any block cipher, we may build a pseudorandom generator using the counter mode [3] or a pseudorandom function using the CBC-MAC [4].

We rely on the following notation. If $f : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ represents a pseudorandom function or permutation, we write $f_k(x)$ for the result of applying f to input x with key $k \in \mathcal{K}$. We write $\langle x, y \rangle$ for the concatenation of x and y, and $x \oplus y$ for the bitwise XOR of x and y. For the remainder of the paper, we let $G : \mathcal{K}_G \to \mathcal{X}^\ell$ be a pseudorandom generator for some ℓ, $F : \mathcal{K}_F \times \mathcal{X} \to Y$ be a pseudorandom function, and $E : \mathcal{K}_E \times \mathcal{Z} \to \mathcal{Z}$ be a pseudorandom permutation. Typically we will have $\mathcal{X} = \{0,1\}^{n-m}$, $\mathcal{Y} = \{0,1\}^m$, and $\mathcal{Z} = \mathcal{X} \times \mathcal{Y} = \{0,1\}^n$.

4 Our Solution with Sequential Scan

In this section, we introduce our solution for searching with sequential scan. We first start with a basic scheme and show that its encryption algorithm provides provable secrecy. We then show how we can extend the first scheme to handle controlled searching and hidden searches. We describe our final scheme which satisfies all the properties we mentioned earlier including query isolation at the end.

4.1 Scheme I: The Basic Scheme

Alice wants to encrypt a document which contains the sequence of words $W_1, \ldots, W_\ell$. Intuitively, the scheme works by computing the bitwise exclusive or (XOR) of the clear-text with a sequence of pseudorandom bits which have a special structure. This structure will allow to search on the data without revealing anything else about the clear text.

More specifically, the basic scheme is as follows. Alice generates a sequence of pseudorandom values $S_1, \ldots, S_\ell$ using some stream cipher (namely, the pseudorandom generator G), where each S_i is $n - m$ bits long. To encrypt a n-bit word W_i that appears in position i, Alice takes the pseudorandom bits S_i, sets $T_i := \langle S_i, F_{k_i}(S_i) \rangle$, and outputs the ciphertext $C_i := W_i \oplus T_i$. Note that only Alice can generate the pseudorandom stream $T_1, \ldots, T_\ell$ so no one else can decrypt. Of course, encryption can be done *on-line*, so that we encrypt each word as it becomes available.

There is some flexibility in how the keys k_i may be chosen. One possibility is to use the same key k at every position in the document. Another alternative is to choose a new key k_i for each position independent of all other keys. More generally, at each position, Alice can either (a) choose k_i to be the same as some previous k_j ($j < i$), or (b) choose k_i independently of all the previous keys. We shall see later how this flexibility allows us to support a variety of interesting features.

The basic scheme provides provable secrecy if the pseudorandom function F and the pseudorandom generator G are secure. By this, we mean that, at each position where k_i is unknown, the values T_i are indistinguishable from truly random bits for any computationally-bounded adversary. We formalize the theorem as below.

Theorem 4.1. *If F is a (t, ℓ, e_F)-secure pseudorandom function and G is a (t, e_G)-secure pseudorandom generator, and if the key material is chosen as described above, then the algorithm described above for generating the sequence $\langle T_1, \ldots, T_\ell \rangle$ is a $(t - \epsilon, e)$-secure pseudorandom generator, where $e = \ell \cdot e_F + e_G + \ell(\ell-1)/(2|\mathcal{X}|)$ and the constant ϵ is negligible compared to t.*

In other words, we expect the basic scheme to be good for encrypting up to about $\ell_{\max} = O(2^{(n-m)/2})$ words, if the pseudorandom function and pseudorandom generator are adequately secure. See Appendix A for a slightly more precise statement of the theorem and for a full proof.

The basic scheme supports searches over the ciphertext in the following way: if Alice wants to search the word W, she can tell Bob W and the k_i corresponding to each location i where a word W may occur. Bob can then search for W in the ciphertext by checking whether $C_i \oplus W_i$ is of the form $\langle s, F_{k_i}(s) \rangle$ for some s. Such a search can be performed in linear time. At the positions where Bob does not know k_i, Bob learns nothing about the plaintext. Thus, the scheme allows a limited form of control: if Alice only wants Bob to be able to search over the first half of the ciphertext, Alice should reveal only the k_i corresponding to

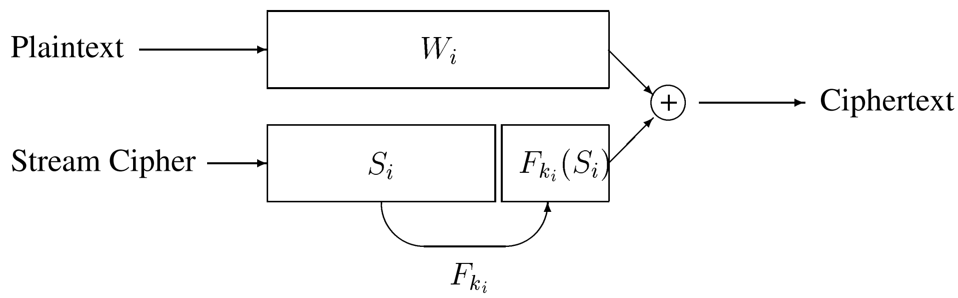

Figure 1. The Basic Scheme

those locations and none of the k_i used in the second half of the ciphertext.

As described so far, the basic scheme is not terribly satisfying: if Alice wants to help Bob search for a word W, either Alice must reveal all the k_i (thus potentially revealing the entire document), or Alice must know in advance which locations W may appear at (which seems to defeat the purpose of remote searching). However, we shall see next how to take care of this difficulty.

4.2 Scheme II: Controlled Searching

Let $f : \mathcal{K}_F \times \{0,1\}^* \to \mathcal{K}_F$ be an additional pseudorandom function, which will be keyed independently of F.

The main idea is to choose our keys as $k_i := f_{k'}(W_i)$. We require that k' be chosen uniformly randomly in $\mathcal{K}$ by Alice and never be revealed. Then, if Alice wish to allow Bob to search for the word W, she reveals $f_{k'}(W)$ and W to him. This allows Bob to identify all the locations where W might occur, but reveals absolutely nothing on the locations i where $W_i \neq W$. This attains our desired goal of *controlled searching*. We show the correctness of this approach in the following theorem.

Theorem 4.2. *Suppose F is a (t, ℓ, e_F)-secure pseudorandom function, f is a (t, ℓ, e_f)-secure pseudorandom function, and G is a (t, e_G)-secure pseudorandom generator. If the key material is chosen as described above, then the algorithm described above for generating the sequence $\langle T_1, \ldots, T_\ell \rangle$ will be a $(t - \epsilon, e_H)$-secure pseudorandom generator, where $e_H = \ell \cdot e_F + e_f + e_G + \ell(\ell-1)/(2|\mathcal{X}|)$.*

This shows that our scheme for controlled searching is about as good as the basic scheme, if the underlying primitives are secure. See Appendix A for a proof as well as a more precise formulation.

Various extensions of this idea are possible. If the document to be encrypted consists of a series of chapters, an alternative approach is to generate the key k_i for the word W in chapter C as $k_i := f_{k'}(\langle C, W \rangle)$. This allows Alice to control which chapters Bob may search in as well as controlling which words Bob may search for.

We can take this idea even further by using a hierarchical key management scheme. Alice sets $k_i := f_l(\langle 0, C \rangle)$ and $l := f_{k'}(\langle 1, W_i \rangle)$. Then she can reveal either (1) $f_{f_{k'}(\langle 1, W \rangle)}(\langle 0, C \rangle)$ for each chapter of interest or (2) $f_{k'}(\langle 1, W \rangle)$ itself if she wishes to succinctly authorize Bob to search for W in all the chapters.

This scheme still does not support hidden search queries: in order to let Bob search for the location where the word W appears, Alice has to reveal W to Bob. We shall see next that this problem can be easily fixed.

4.3 Scheme III: Support for Hidden Searches

Suppose Alice would now like to ask Bob to search for a word W but she is not willing to reveal W to Bob. We propose a simple extension to the above scheme to support this goal.

Alice should merely pre-encrypt each word W of the clear text separately using a deterministic encryption algorithm $E_{k''}$. Note that E is not allowed to use any randomness, and the computation of $E_{k''}(x)$ may depend only on x and must not depend on the position i in the document where x is found. So we may think of this pre-encryption step as ECB encryption of the words of the document using some block cipher. (Of course, if the word is very long, internally the map $E_{k''}$ may be implemented by CBC-encrypting W_i with a constant IV, or some such, but the point is that this process must be the same at every position of the document.) We let $X_i := E_{k''}(W_i)$.

After the pre-encryption phase, Alice has a sequence of E-encrypted words $X_1, \ldots, X_\ell$. Now she post-encrypts that sequence using the stream cipher construction described above to obtain $C_i := X_i \oplus T_i$, where $X_i = E_{k''}(W_i)$ and $T_i = \langle S_i, F_{k_i}(X_i) \rangle$.

To search for a word W, Alice computes $X := E_{k''}(W)$ and $k := f_{k'}(X)$, and sends $\langle X, k \rangle$ to Bob. Note that this

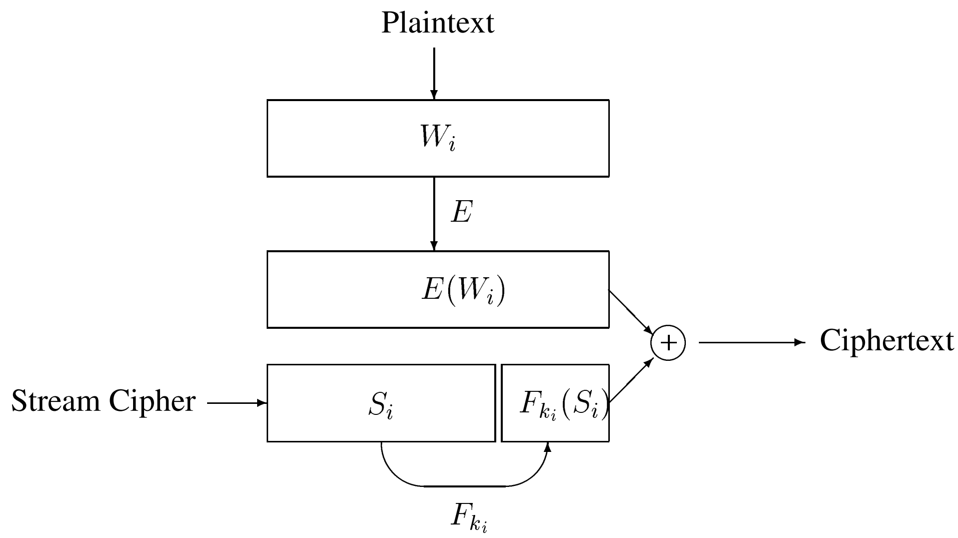

Figure 2. The Scheme for Hidden Search

allows Bob to search for W without revealing W itself. It is easy to see that this scheme satisfies the hidden search property as long as the pre-encryption E is secure.

4.4 Scheme IV: The Final Scheme

Careful readers may have noticed that Scheme III actually suffers from a small inadequacy: if Alice generates keys k_i as $k_i := f_{k'}(E_{k''}(W_i))$ then Alice can no longer recover the plaintext from just the ciphertext because she would need to know $E_{k''}(W_i)$ (more precisely, the last m bits of $E_{k''}(W_i)$) before she can decrypt. This defeats the purpose of an encryption scheme, because even legitimate principals with access to the decryption keys will be unable to decrypt. (Scheme II also has a similar inadequacy, but as we will show below, the best way to fix it is to introduce pre-encryption as in Scheme III.)

We now show a simple fix for this problem. In the fixed scheme, we split the pre-encrypted word $X_i = E_{k''}(W_i)$ into two parts, $X_i = \langle L_i, R_i \rangle$, where L_i (respectively R_i) denotes the first $n-m$ bits (resp. last m bits) of X_i. Instead of generating $k_i := f_{k'}(E(W_i))$, Alice should generate k_i as $k_i := f_{k'}(L_i)$. To decrypt, Alice can generate S_i using the pseudorandom generator (since Alice knows the seed), and with S_i she can recover L_i by XORing S_i against the first $n-m$ bits of C_i. Finally, knowledge of L_i allows Alice to compute k_i and thus finish the decryption.

This fix is not secure if the W_i's are not encrypted since it might be very likely in some cases that different words have the same first $n - m$ bits. Pre-encryption will eliminate this problem, since with high probability all the L_i's are distinct. (Assuming that the pre-encryption E is a pseudorandom permutation, then due to the birthday paradox [15], the probability that at least one collision happens after encrypting ℓ words is at most $\ell(\ell-1)/2^{(n-m+1)}$.)

With this fix, the resulting scheme is provably secure, and in fact we can also show that it provides query isolation, meaning that even when a single key k_i is revealed, no extra information is leaked beyond the ability to identify the positions where the corresponding word W_i occurs.

Theorem 4.3. *Suppose E is a (t, ℓ, e_E)-secure pseudorandom permutation, F is a (t, ℓ, e_F)-secure pseudorandom function, f is a (t, ℓ, e_f)-secure pseudorandom function, G is a (t, e_G)-secure pseudorandom generator, and we choose the key material as described above. Then the algorithm described above for generating the sequence $\langle T_1, \ldots, T_\ell \rangle$ will be a a $(t - \epsilon, e_H)$-secure pseudorandom generator, where $e_H = \ell \cdot e_F + e_f + e_G + \ell(\ell-1)/(2|\mathcal{X}|)$.*

Moreover, if we disclose one k_i and consider the reduced sequence T' obtained by discarding all the T_j values at positions where $W_j = W_i$, then we obtain a $(t-\epsilon, e'_H)$-secure pseudorandom generator, where $e'_H = e_H + e_E + \ell/|\mathcal{X}|$.

Strictly speaking, the proof of the theorem does not actually require E to be a pseudorandom permutation: if φ denotes the (keyed) map sending W to the first $n - m$ bits of $E_{k''}(W)$, then we can make do with the much weaker assumption that collisions in φ should be rare. As a special case, if the first b bits of φ ($b \leq n - m$) can be shown to be a pseudorandom function, then E will necessarily have the required property, and we will be able to prove a result analogous to Theorem 3. This suggests that for pre-encryption

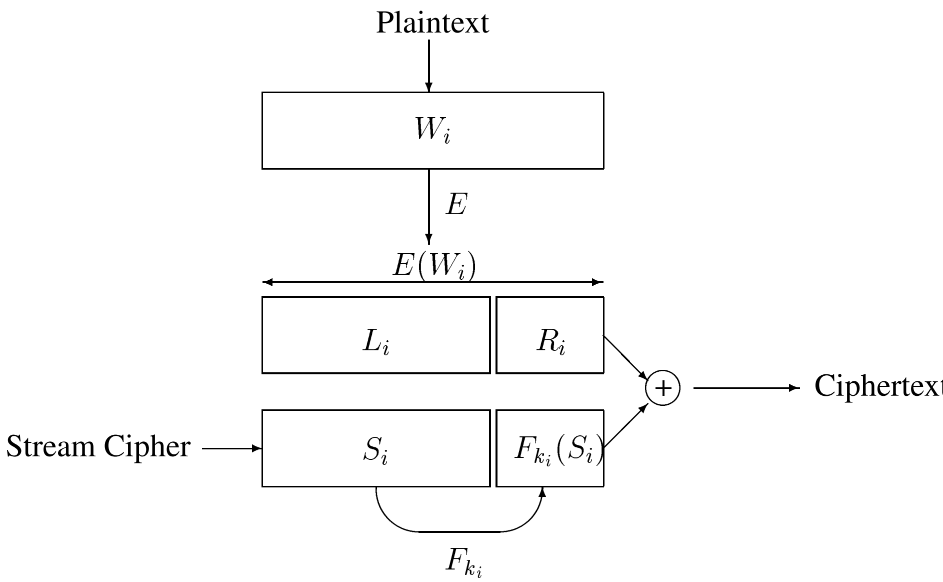

Figure 3. The Final Scheme

of long blocks it may be convenient to take $E_{k''}(W)$ to be the *bit-reversal* of the CBC encryption of W under key k'' (using a constant IV).

After encrypting word by word, Alice can also re-order the ciphertext according to some pseudorandom permutation (known only to Alice). In this case, when Bob performs the search of a word, he will not be able to tell the position where the word occurs in the real plaintext.

5 Discussion

5.1 Other Practical Considerations

We can see that updates in this scheme are easy. For example, if Alice wants to add a new document into Bob's data storage, she can simply encrypt it in the appropriate way and instruct Bob to append it to the already-stored ciphertext. Moreover, since the keys can be generated hierarchically from a master key, the key storage and management is also very convenient: Alice only needs to remember one password, the master key.

The underlying technique of embedding information in pseudorandom bit streams may also be of independent interest: we speculate that this simple trick might prove useful for other applications, too.

5.2 Supporting More Advanced Search Queries

The schemes we presented earlier only address the problem of searching for a single word. We show several examples to illustrate that it is relatively easy to implement more advanced searching functionality using our scheme as a fundamental building block.

It is clear that we can easily support advanced search queries which use boolean operators (e.g., W and W'), proximity queries (e.g., W near W'), and phrase searches (e.g., W immediately precedes W').

We can also support searches if the query is given as a regular expression using, e.g., wildcards in a limited form. For example, if Alice wishes to search for the word $ab[a\text{–}z]$, then she can actually generate 26 search queries of the form $\{aba, abb, \ldots, abz\}$. However, the number of queries required (and the information leaked to the server) clearly grows dramatically as the search word becomes more general.

For many applications the purpose of the search is to find documents which contain a specific word, where the position or the number of occurrences are not relevant. For example, searching email is such an application, where the query takes on the form "find all email from Joe". For this application, the previous search schemes leaked information, since the server would know the positions of the word in the document, or at least the frequency of a word in a document in case the word order is scrambled. Since we only need to know whether a given document contains a word, we can use the following trick. We add a count to each word, which counts how many times that word occurs previously in that document. For example, the first occurrence of the word "urgent" is stored as $\langle 0, \text{urgent}\rangle$, the second occurrence as $\langle 1, \text{urgent}\rangle$, etc. This allows Alice to search for the first occurrence only, if she wishes just to identify the documents where the word appears; and Bob does not gain any information about other positions of the search term in the document. As an additional feature, this encoding allows Alice to search for documents, that contain n or more occurrences of the word W by searching for $\langle n-1, W\rangle$.

5.3 Dealing with Variable-Length Words

In our scheme, the minimal unit we can search for is an individual word. So far we have assumed that the clear text can be easily broken into a sequence of words of a fixed length. But this might not be true in a normal text document. For example, if the minimal unit of search interest is one English word, then we have to deal with the fact that English words differ in length.

One possibility is to pick a fixed-size block that is long enough to contain most words. Words that are too short or too long may be padded to a multiple of the block size with some pre-determined padding format. (Note that the padding cannot be random since Alice needs to know the padding in order to perform the search.) However, such a padding scheme would introduce space inefficiency. Also, for security reasons we cannot decrease the word length below a certain limit.

Another solution is to use variable length words. In this case, to support random-access decryption, the length of each word also needs to be stored with the word. One natural approach is to store the length field before each word in the file, and to glue the length field and word together as one word to perform encryption and search using our standard schemes.

When words lengths may vary, it is important to hide the length information from the server, because revealing the length of each word might allow for statistical attacks. Fortunately, in this case the server does not need to know the lengths to perform a search: he may just scan through the file and check for a match at each possible bit boundary. In this case, the cost of each scan is increased, because the number of operations is determined by the bit-length of the document rather than by the number of blocks in the document. However, such an approach may provide better space efficiency than is available with a block-oriented scheme.

5.4 Searching with an Encrypted Index

Sequential scan may not be efficient enough when the data size is large. For some applications, i.e. large databases, a common technique to speed up the searching is to use a pre-computed index. Here we show how we can answer search queries with the aid of an encrypted index without sacrificing security.

An index contains a list of key words; with each key word is a list of pointers to documents where the key word appears. The key words are words of interest that Alice may want to search for later. Alice can certainly build the index of her clear text documents and then encrypt the clear text and the index and store the ciphertext on Bob. The interesting question is how to encrypt the index.

A naive way would be to just encrypt the key words in the index and leave the lists of positions in clear. This makes it easy for Bob to perform search queries on Alice's behalf, but also leaks a lot of information to Bob and hence may allow him to apply various statistical attacks. Therefore, we reject this naive approach.

A simple way is to also encrypt the document pointers in each list in the index. Consequently, when Bob searches for $E(W)$ and finds a match, he returns Alice the encrypted list of matching positions from the index. Alice may decrypt the encrypted entries and send Bob another request to retrieve the relevant documents. One possible advantage for this scheme is that the request could be embedded in other retrievals so that Bob might have uncertainty about the correlation of the search request and the retrieval request for ciphertext. The disadvantage is that Alice has to spend an extra round-trip time to retrieve the documents.

If Alice does not want to wait for an extra round-trip time, or if Alice would like Bob to merge the results of several search queries for her, other techniques are also available. For example, she may encrypt the list of document pointers in the index using a key k_W related $E(W)$, i.e. $k_W = f_{k'''}(E(W))$. Hence, when Alice wants to search for the word W, she will reveal $\langle E(W), k_W\rangle$ to Bob. In order to prevent Bob from doing statistical analysis on the index, it is better to keep the lists of pointers in a fixed-size list. For words that appear infrequently, Alice can pad the list to the fixed size. For more common words, Alice can split the long list into several lists with the fixed size; then, to search for such a word, Alice will need to ask Bob to perform and merge several search queries in parallel. Note that by keeping the lists of pointers in a fixed-size list, we are mainly preventing Bob from learning statistical information on the key words that he has not searched. For the key words that Bob has searched, he might still be able to learn some statistical information from Alice's access pattern. This is acceptable from our point of view since Alice only wants to retrieve relevant documents in the first place.

Note that a general disadvantage for index search is that whenever Alice changes her documents, she must update the index. There is a trade-off between how much index Alice updates and how much information Bob might be able to learn. For example, if Alice does not change the list of document pointers for one key word entry when she adds a new document into Bob's data storage, Bob would be able to tell that the key word does not appear in the new document.

Hence, Alice needs to update a substantial part of the index to hide the real updates which can be quite expensive. It is an interesting research question to develop schemes which support more efficient updates.

5.5 More Security Issues

In all our schemes, by allowing Bob to search for a word W we effectively disclose to him a list of potential locations where W might occur. If we allow Bob to search for too many words, he may be able to use statistical techniques to start learning important information about the documents. One possible defense is to decrease m (so that false matches are more prevalent and thus Bob's information about the plaintext is 'noisy'), but we have not analyzed the cost-effectiveness of this tradeoff in any detail.

A better defense is for Alice to periodically change the key, re-encrypt all the documents under the new key, and re-order the ciphertext according to some pseudorandom permutation (known to Alice but not to Bob). This will help prevent Bob from learning correlations or other statistical information over time. This technique may also be helpful if Alice wants to hide from Bob the places where the searched word occurs in the documents of interest.

In all the schemes we have discussed so far, we must trust Bob to return all the search results. If Bob holds out on us and returns only some (but not all) of the search results, Alice will have no way to detect this. In our scope of interest, we assume Bob does not misbehave in this way. Even when this type of attack is present, it is possible to combine our scheme with hash tree techniques [17] to ensure the integrity of the data and detect such attacks, although a full description of this countermeasure is out of the scope of the paper.

6 Related Work

Many researchers have investigated the problem of providing secrecy and integrity when using an untrusted file server or external untrusted memory [5, 12, 1, 6]. But as far as we know, no previous work has provided a solution for searching on encrypted data. Secure multi-party computation and oblivious functions are also intensely studied (see, e.g., [14, 8]). We believe there could be a solution to the problem of searching on encrypted data using multi-party computation, but it would require a high overhead, for example multiple servers. Our solution only needs one server to search on encrypted data, and hence is a more practical solution.

Several researchers have studied the Private Information Retrieval (PIR) problem [9], so that clients may access entries in a distributed table without revealing which entries they are interested in. The PIR literature typically aims for very strong information-theoretic security bounds, which makes it harder to find practical schemes: PIR schemes often require multiple non-colluding servers, consume large amounts of bandwidth, do not guarantee the confidentiality of the data, do not support private keyword searching, and do not support controlled searching or query isolation (but see, e.g., [16, 13, 10, 7] for important exceptions which allow to remove some—but not all—of those limitations). In contrast, although our scheme does not solve the PIR problem, it needs only a single server (with no impractical trust assumptions), has low computational complexity, and supports private keyword searching with very strong security properties.

7 Conclusion

We have described new techniques for remote searching on encrypted data using an untrusted server and provided proofs of security for the resulting crypto systems. Our techniques have a number of crucial advantages: they are provably secure; they support controlled and hidden search and query isolation; they are simple and fast (More specifically, for a document of length n, the encryption and search algorithms only need $O(n)$ stream cipher and block cipher operations); and they introduce almost no space and communication overhead. Our scheme is also very flexible, and it can easily be extended to support more advanced search queries. We conclude that this provides a powerful new building block for the construction of secure services in the untrusted infrastructure.

Acknowledgments

We would like to thank Doug Tygar for his valuable suggestions and advice. We would also like to thank John Kubiatowicz for encouraging the work on the problem of searching on encrypted data. We would also like to thank Bob Briscoe for his helpful comments on the paper.

References

[1] Nancy Amato and Michael Loui. Checking linked data structures. In *Proceedings of the 24th annual International Symposium on Fault-Tolerant Computing*, 1994.

[2] M. Bellare. Practice-oriented provable-security. In *Proceedings of First International Workshop on Information Security (ISW 97)*. Springer-Verlag, 1998. Lecture Notes in Computer Science No. 1396.

[3] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A concrete security treatment of symmetric encryption:

Analysis of the des modes of operation. In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science*. IEEE, 1997.

[4] M. Bellare, J. Kilian, and P. Rogaway. The security of the cipher block chaining message authentication code. In *CRYPTO'94*. Springer-Verlag, 1994. Lecture Notes in Computer Science No. 839.

[5] Matt Blaze. A cryptographic file system for unix. In *Proceeding of the 1st ACM Conference on Communications and Computing Security*, 1993.

[6] M. Blum, W. Evans, P. Gemmell, S. Kannan, and M. Noar. Checking the correctness of memories. *Algorithmica*, 12(2/3), 1994.

[7] C. Cachin, S. Micali, and M. Stadler. Computationally private information retrieval with polylogarithmic communication. In *EUROCRYPT'99*. Springer-Verlag, 1999. Lecture Notes in Computer Science No. 1592.

[8] R. Canetti. *Studies in Secure Multi-Party Computation and Applications*. PhD thesis, Weizmann Institue of Science, Israel, 1995.

[9] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private information retrieval. In *Proceedings of the 68th Annual Symposium on Foundations of Computer Science*. IEEE, 1995.

[10] Benny Chor, Niv Gilboa, and Moni Naor. Private information retrieval by keywords. Report 98-03, Theory of Cryptography Library, 1998.

[11] M. Crispin. Internet message access protocol - version 4. RFC1730, December 1994.

[12] Premkumar T. Devanbu and Stuart G. Stubblebine. Stack and queue integrity on hostile platforms. In *Proceeding of IEEE Symposium on Security and Privacy*, 1998.

[13] Y. Gertner, Y. Ishai, and E. Kushilevitz. Protecting data privacy in private information retrieval schemes. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*. ACM, 1998.

[14] Oded Goldreich. Secure multi-party computation. Working Draft, 1998.

[15] S. Goldwasser and M. Bellare. Lecture notes on cryptography. available online from http://www-cse.ucsd.edu/users/mihir/papers/gb.html.

[16] E. Kusilevitz and R. Ostrovsky. Replication is not needed: single database, computationally-private information retrieval. In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science*. IEEE, 1997.

[17] Ralph C. Merkle. A certified digital signature. In G. Brassard, editor, *CRYPTO89*, pages 218–238. Springer-Verlag, 1990. Lecture Notes in Computer Science No. 435.

A Proofs of Security

Define $H' : \mathcal{K}_F \times \mathcal{K}_G \to (\mathcal{X} \times \mathcal{Y})^\ell$ by

$$H'(k, k_G) = \langle s_1, F_k(s_1), \dots, s_\ell, F_k(s_\ell) \rangle,$$

where we write $s_j \in \mathcal{X}$ as shorthand for the j-th block of $G(k_G)$. Please refer to Section 3 for notation and precise definitions of security for the primitives E, F, G.

Lemma A.1. *If F is a (t, ℓ, e_F)-secure pseudorandom function and G is a (t, e_G)-secure pseudorandom generator, then H' (defined as above) is a $(t - \epsilon, e_{H'})$-secure pseudorandom generator, where $e_{H'} = e_F + e_G + \ell(\ell - 1)/(2|\mathcal{X}|)$ and the constant ϵ is negligible compared to t.*

Proof. Define $\Psi(k) = \langle u_1, F_k(u_1), \dots, u_\ell, F_k(u_\ell) \rangle$ where $u_1, \dots, u_\ell$ are ℓ independent random variables each drawn from the uniform distribution on $\mathcal{X}$. Also, let U be a random variable with the uniform distribution on $(\mathcal{X} \times \mathcal{Y})^\ell$. We will write H', Ψ, U for the random variables obtained by choosing k, k_G uniformly at random from $\mathcal{K}_F \times \mathcal{K}_G$. The goal is to show that H' and U are indistinguishable to any computationally-bounded adversary. The proof will proceed by showing first that H' and Ψ are indistinguishable, and second that Ψ and U are indistinguishable.

First, we show that no algorithm with running time $t - \epsilon$ can distinguish between H' and Ψ with advantage better than e_G. Suppose not, i.e., there exists an algorithm A with running time at most $t - \epsilon$ and

$$\mathrm{Adv}\, A = |\Pr[A(H') = 1] - \Pr[A(\Psi) = 1]| \geq e_G.$$

Then we exhibit an algorithm B with running time at most t which distinguishes the output of G from a truly random bit string with advantage at least e_G. The algorithm B works in the following way: on input $s = \langle s_1, \dots, s_\ell \rangle \in \mathcal{X}^\ell$, it runs A on input $I = \langle s_1, F_k(s_1), \dots, s_\ell, F_k(s_\ell) \rangle$ and halts with the output $A(I)$ from A. Note that $\Pr[B(G(k_G)) = 1] = \Pr[A(H') = 1]$ and $\Pr[B(U') = 1] = \Pr[A(\Psi) = 1]$, where U' is a uniformly-distributed random variable on $\mathcal{X}^\ell$. Thus we calculate

$$\begin{aligned} \mathrm{Adv}\, B &= |\Pr[B(G(k_G)) = 1] - \Pr[B(U') = 1]| \\ &= |\Pr[A(H') = 1] - \Pr[A(\Psi) = 1]| \\ &= \mathrm{Adv}\, A \geq e_G, \end{aligned}$$

which contradicts our assumption that $\mathrm{Adv}\, A$ was large.

Second, we show that no algorithm with running time $t-\epsilon$ can distinguish between Ψ and U with advantage better than $e_F + \ell(\ell-1)/(2|\mathcal{X}|)$. Suppose not, i.e., there exists an algorithm A with

$$\mathrm{Adv}\, A = |\Pr[A(\Psi) = 1] - \Pr[A(U) = 1]| \geq e_F + \frac{\ell(\ell-1)}{2|\mathcal{X}|}.$$

Then we construct an oracle algorithm B which distinguishes F from a truly random function, as follows: B chooses $u_1, \ldots, u_\ell$ uniformly and independently at random, queries its oracle f a total of ℓ times with the inputs u_i to receive the outputs $f(u_i)$, runs A on the string $I = \langle u_1, f(u_1), \ldots, u_\ell, f(u_\ell)\rangle$, and halts with $A(I)$ as its output. We want to show that $\mathrm{Adv}\, B$ is large. Of course, $\Pr[B^{F_k} = 1] = \Pr[A(\Psi) = 1]$, so it remains only to characterize $\Pr[B^R = 1]$, where R is a truly random function selected uniformly from the set of all maps $\mathcal{X} \to \mathcal{Y}$.

Let E denote the event that the values $u_1, \ldots, u_\ell$ are all distinct. Also, write $\overline{\mathsf{E}}$ for the complementary event, i.e., the case where there exist i, j with $1 \leq i < j \leq \ell$ such that $u_i = u_j$. Now we may compute

$$\begin{aligned}
\Pr[B^R = 1] &= \Pr[B^R = 1 \mid \mathsf{E}] \cdot \Pr[\mathsf{E}] \\
&\quad + \Pr[B^R = 1 \mid \overline{\mathsf{E}}] \cdot \Pr[\overline{\mathsf{E}}] \\
&\leq \Pr[A(U) = 1 \mid \mathsf{E}] \cdot \Pr[\mathsf{E}] + \Pr[\overline{\mathsf{E}}] \\
&\leq \Pr[A(U) = 1 \mid \mathsf{E}] \cdot \Pr[\mathsf{E}] \\
&\quad + \Pr[A(U) = 1) \mid \overline{\mathsf{E}}] \cdot \Pr[\overline{\mathsf{E}}] + \Pr[\overline{\mathsf{E}}] \\
&\leq \Pr[A(U) = 1] + \frac{\ell(\ell-1)}{2|\mathcal{X}|}.
\end{aligned}$$

Without loss of generality, we may assume $\Pr[B^{F_k} = 1] \geq \Pr[B^R = 1]$. Thus,

$$\begin{aligned}
\mathrm{Adv}\, B &= |\Pr[B^{F_k} = 1] - \Pr[B^R = 1]| \\
&\geq |\Pr[A(\Psi) = 1] - \Pr[A(U) = 1] \\
&\quad -\ell(\ell-1)/(2|\mathcal{X}|)| \\
&\geq \mathrm{Adv}\, A - \ell(\ell-1)/(2|\mathcal{X}|) \geq e_F,
\end{aligned}$$

which contradicts our assumption that $\mathrm{Adv}\, A$ was large.

Next, we note that the above two results suffice to show that no algorithm with running time $t - \epsilon$ can distinguish between H' and U with advantage better than $e_F + e_G + \ell(\ell-1)/(2|\mathcal{X}|)$. Consider any algorithm A that attempts to distinguish H' from U; then

$$\begin{aligned}
\mathrm{Adv}\, A &= |\Pr[A(H') = 1] - \Pr[A(U) = 1]| \\
&= |\Pr[A(H') = 1] - \Pr[A(\Psi) = 1] \\
&\quad + \Pr[A(\Psi) = 1] - \Pr[A(U) = 1]| \\
&\leq |\Pr[A(H') = 1] - \Pr[A(\Psi) = 1]| \\
&\quad + |\Pr[A(\Psi) = 1] - \Pr[A(U) = 1]| \\
&< e_G + e_F + \frac{\ell(\ell-1)}{2|\mathcal{X}|}
\end{aligned}$$

where the final line follows by applying the previous two parts of the proof. $\square$

Next define $H : (\mathcal{K}_F)^\ell \times \mathcal{K}_G \to (\mathcal{X} \times \mathcal{Y})^\ell$ by

$$H(k, k_G) = \langle s_1, F_{k_1}(s_1), \ldots, s_\ell, F_{k_\ell}(s_\ell)\rangle,$$

where s_j is defined as before. This is an independently-keyed version of the construction H' analyzed in Lemma A.1. In other words, the key $k = \langle k_1, \ldots, k_\ell\rangle$ is a vector of ℓ independent random variables that are uniformly distributed on $\mathcal{K}_F$.

Lemma A.2. *If F is a $(t, 1, e_F)$-secure pseudorandom function and G is a (t, e_G)-secure pseudorandom generator, then H (defined as above, with independent keys) is a $(t - \epsilon, e_H)$-secure pseudorandom generator, where $e_H = \ell \cdot e_F + e_G$ and the constant ϵ is negligible compared to t.*

Proof. The outline of the proof is as in Lemma A.1, except the second part (the treatment of the indistinguishability of Ψ and U) must be modified slightly. We define Ψ_i ($0 \leq i \leq n$) by

$$\begin{aligned}
\Psi_i(k) &= \langle u_1, F_{k_1}(u_1), \ldots, u_i, F_{k_i}(u_j), \\
&\qquad u_{i+1}, w_{i+1}, \ldots, u_\ell, w_\ell\rangle
\end{aligned}$$

where the $w_{i+1}, w_{i+2}, \ldots, w_\ell$ are independent, uniformly-distributed random variables on $\mathcal{Y}$ and the u_i are as above (i.e., uniform on $\mathcal{X}$ and independent of everything else). Note, for example, that $\Psi_0 = U$ and $\Psi_\ell = \Psi$. Now we show that each pair of neighbors in the sequence $\Psi_0, \Psi_1, \ldots, \Psi_\ell$ are $(t - \epsilon, e_F)$-indistinguishable.

Suppose not, i.e., there exists some i and some algorithm A distinguishing Ψ_i from Ψ_{i-1} with advantage $\mathrm{Adv}\, A \geq e_F$. Then we construct an oracle algorithm B that distinguishes F from a truly random function, as follows: B chooses $u_1, \ldots, u_\ell \in \mathcal{X}$, $k_1, \ldots, k_{i-1} \in \mathcal{K}_F$, and $w_{i+1}, \ldots, w_\ell \in \mathcal{Y}$ independently and uniformly at random; B uses its own key material to compute $F_{k_1}(u_1), \ldots, F_{k_{i-1}}(u_{i-1})$ uses its oracle f to compute $f(u_i)$; then B runs A on the string

$$\begin{aligned}
I &= \langle u_1, F_{k_1}(u_1), \ldots, u_{i-1}, F_{k-1}(u_{i-1}), \\
&\qquad u_i, f(u_i), u_{i+1}, w_{i+1}, \ldots, u_\ell, w_\ell\rangle
\end{aligned}$$

and finally halts with $A(I)$ as its output. By the definition of B, we have $\Pr[B^{F_k} = 1] = \Pr[A(\Psi_i) = 1]$. Also, we see easily that $\Pr[B^R = 1] = \Pr[A(\Psi_{i-1}) = 1]$, since the output of a random function R that is invoked only once is uniform and independent of everything else in sight. Therefore, we find that

$$\begin{aligned}
\mathrm{Adv}\, B &= |\Pr[B^{F_k} = 1] - \Pr[B^R = 1]| \\
&= |\Pr[A(\Psi_i) = 1] - \Pr[A(\Psi_{i-1}) = 1]| \\
&= \mathrm{Adv}\, A \geq e_F,
\end{aligned}$$

which contradicts our assumption that Adv A was large.

Now a simple application of the triangle inequality suffices to show that no algorithm can $(t-\epsilon, \ell\cdot e_F)$-distinguish $\Psi_\ell = \Psi$ from $\Psi_0 = U$: if A is any such algorithm, then

$$\begin{aligned} \text{Adv}\, A &= |\Pr[A(\Psi_\ell)=1] - \Pr[A(\Psi_0)=1] \\ &\leq \sum_{1\leq i\leq \ell} |\Pr[A(\Psi_i)=1] - \Pr[A(\Psi_{i-1})=1]| \\ &< \ell\cdot e_F. \end{aligned}$$

This suffices to complete the proof, since the rest of the proof of Lemma A.1 now carries through. □

We are, at last, ready to consider the construction H above where now the keys k_i are not necessarily chosen independently, but instead are chosen according to some distribution D on $(\mathcal{X}\times\mathcal{Y})^\ell$. We require D to have the following property:

Definition 1. *We say that a distribution D on the keys $k_1,\ldots,k_\ell$ has the* twining property *if, for all j, either (a) there exists $i<j$ such that $\Pr_D[k_j = k_i] = 1$, or (b) D selects k_j uniformly at random from $\mathcal{K}_F$, independently of $k_1,\ldots,k_{j-1}$.*

Theorem A.3. *If F is a (t,ℓ,e_F)-secure pseudorandom function and G is a (t,e_G)-secure pseudorandom generator, and if the key $k\in(\mathcal{K}_F)^n$ is chosen according to a distribution D with the twining property, then H is a $(t-\epsilon, e_H)$-secure pseudorandom generator, where $e_H = \ell\cdot e_F + e_G + \ell(\ell-1)/(2|\mathcal{X}|)$ and the constant ϵ is negligible compared to t.*

Proof. First observe that we may, without loss of generality, reorder the keys so that, for all j, either (a) $\Pr_D[k_j = k_{j-1}] = 1$, or (b) k_j is selected uniformly and independently of $k_1,\ldots,k_{j-1}$. Thus, we obtain a sequence of keys of the form $k = \langle\langle k_1',\ldots,k_1'\rangle, \langle k_2',\ldots,k_2'\rangle, \ldots\langle k_m',\ldots,k_m'\rangle\rangle$ where the $k_1', k_2',\ldots,k_m'$ are all independent. Let ℓ_i denote the number of times that key k_i' is repeated in k, and let χ be a function which associates to each i the first j such that $\Pr_D[k_j = k_i'] = 1$.

Now we simply combine the techniques used in the proofs of Lemma A.1 and Lemma A.2. We use a hybrid argument as in Lemma A.2, this time defining Ψ_i by

$$\begin{aligned} \Psi_i(k) = &= \langle u_1, F_{k_1}(u_1),\ldots,u_{j'-1}, F_{k_{j'-1}}(u_{j'-1}), \\ &\quad u_{j'}, w_{j'},\ldots,u_\ell, w_\ell\rangle \end{aligned}$$

where we define $j' = \chi(i+1)$. We can see (using the arguments presented in the first part of the proof of Lemma A.1) that $\Psi_m = \Psi$ is $(t-\epsilon, e_G)$-indistinguishable from H. To obtain the desired result, we next show that each pair of neighbors Ψ_{i-1}, Ψ_i in the sequence $\Psi_0, \Psi_1, \ldots, \Psi_m$ is $(t-\epsilon, e_F + \ell_i(\ell_i - 1)/(2|\mathcal{X}|))$-indistinguishable.

Suppose not, i.e., there exists some i and some algorithm A distinguishing Ψ_i from Ψ_{i-1} with advantage Adv $A \geq e_F + \ell_i(\ell_i-1)/(2|\mathcal{X}|)$. Let $j = \chi(i)$ and $j' = \chi(i+1)$. Then we construct an oracle algorithm B that distinguishes F from a truly random function, as follows: B chooses $u_1,\ldots,u_\ell\in\mathcal{X}$, $k_1',\ldots,k_{i-1}'\in\mathcal{K}_F$, and $w_{j'},\ldots,w_\ell\in\mathcal{Y}$ independently and uniformly at random; B uses its own key material to compute $F_{k_1}(s_1),\ldots,F_{k_{j-1}}(s_{j-1})$ and uses its oracle f to compute $f(s_j), f(s_{j+1}),\ldots,f(s_{j'-1})$; then B runs A on the string

$$\begin{aligned} I &= \langle\langle u_1, F_{k_1}(u_1),\ldots,u_{j-1}, F_{j-1}(u_{j-1})\rangle, \\ &\quad \langle u_j, f(u_j),\ldots,u_{j'-1}, f(u_{j'-1})\rangle, \\ &\quad \langle u_{j'}, w_{j'},\ldots,u_\ell, w_\ell\rangle\rangle \end{aligned}$$

and finally halts with $A(I)$ as its output. We have $\Pr[B^{F_k} = 1] = \Pr[A(\Psi_i) = 1]$. Also, using the argument in the second part of the proof of Lemma A.1, we obtain the bound

$$\Pr[B^R = 1] \leq \Pr[A(\Psi_{i-1}) = 1] + \ell_i(\ell_i - 1)/(2|\mathcal{X}|),$$

from which we may conclude that Adv $B \geq$ Adv $A - \ell_i(\ell_i - 1)/(2|\mathcal{X}|) \geq e_F$, and this contradicts our assumption that Adv A was large.

Finally, using the triangle inequality, and noting that

$$\sum_{1\leq i\leq m} \frac{\ell_i(\ell_i-1)}{2|\mathcal{X}|} \quad\leq\quad \frac{\ell(\ell-1)}{2|\mathcal{X}|},$$

we obtain the desired result. □

Next, we consider the security of H when the key material is chosen using a pseudorandom function $f:\mathcal{K}_F\times\{0,1\}^*\to\mathcal{K}_F$ instead of using truly random bits. We will require $k_f\in\mathcal{K}_F$ to be chosen uniformly at random, independent of everything else.

Theorem A.4. *Suppose F is a (t,ℓ,e_F)-secure pseudorandom function, f is a (t,ℓ,e_f)-secure pseudorandom function, and G is a (t,e_G)-secure pseudorandom generator. Suppose moreover that we choose the keys k_i as $k_i = f_{k_f}(W_i)$. Then H will be a $(t-\epsilon, e_H)$-secure pseudorandom generator, where $e_H = \ell\cdot e_F + e_f + e_G + \ell(\ell-1)/(2|\mathcal{X}|)$.*

Proof. We will show that the resulting distribution D on the keys has an 'approximate twining property', in the sense that D is computationally indistinguishable from a distribution with the twining property. In particular, the latter distribution is given by the random variables $k_i^R = R(W_i)$, where R is a truly random function selected uniformly from the set of all maps from $\{0,1\}^*$ to $\mathcal{K}_F$.

A straightforward simulation argument shows that the random variables $H(k, k_G)$ and $H(k^R, k_G)$ are $(t-\epsilon, e_f)$-indistinguishable. Suppose not, so that there exists an algorithm A that distinguishes those two random variables with running time at most $t-\epsilon$ and advantage $\mathrm{Adv}\, A \geq e_f$. Then we show that we can construct an adversary B which (t, ℓ, e_f)-breaks f. The oracle algorithm B works as follows: B picks $k_G \in \mathcal{K}_G$ uniformly at random; B computes $\kappa_i = g(W_i)$ using its oracle g and computes $\langle s_1, \ldots, s_\ell \rangle = G(k_G)$; then B runs A on the string $I = \langle s_1, F_{\kappa_1}(s_1), \ldots, s_\ell, F_{\kappa_\ell}(s_\ell) \rangle$ and finally halts with $A(I)$ as its output. We have $\Pr[B^{f_{k_f}} = 1] = \Pr[A(H(k, k_G)) = 1]$ and $\Pr[B^R = 1] = \Pr[A(H(k^R, k_G)) = 1]$. Thus $\mathrm{Adv}\, B = \mathrm{Adv}\, A \geq e_f$, which contradicts our assumption that $\mathrm{Adv}\, A$ is large.

Finally, we note that k^R has the twining property, so by Theorem A.3, the random variables $H(k^R, k_G)$ and U are $(t-\epsilon, \ell \cdot e_F + e_G + \ell(\ell-1)/(2|\mathcal{X}|))$-indistinguishable. Applying the triangle inequality completes the proof. □

A final goal is to show that the scheme of Section 4.4 is secure even after we reveal one key k_i so that a server may perform a search on our behalf. Let $\rho_{\mathcal{X}} : \mathcal{X} \times \mathcal{Y} \to \mathcal{X}$ be a projection onto the first component, so that $\rho_{\mathcal{X}}(\langle x, y \rangle) = x$.

Theorem A.5. *Suppose E is a (t, ℓ, e_E)-secure pseudorandom permutation, F is a (t, ℓ, e_F)-secure pseudorandom function, f is a (t, ℓ, e_f)-secure pseudorandom function, G is a (t, e_G)-secure pseudorandom generator, and we choose the keys k_i as $k_i = f_{k_f}(L_i)$ where $L_i = \rho_{\mathcal{X}}(E_{k''}(W_i)))$. Then H will be a $(t-\epsilon, e_H)$-secure pseudorandom generator, where $e_H = \ell \cdot e_F + e_f + e_G + \ell(\ell-1)/(2|\mathcal{X}|)$.*

Moreover, if we disclose one k_i and consider the projection of H where we discard all outputs at positions j where $W_j = W_i$, then we obtain a $(t-\epsilon, e'_H)$-secure pseudorandom generator, where $e'_H = e_H + e_E + \ell/|\mathcal{X}|$.

Proof. The techniques used in the proof of Theorem A.4 apply directly (replacing the W_i's with L_i's), and one may readily see that H is a $(t-\epsilon, e_H)$-secure pseudorandom generator.

Now, suppose we disclose k_i. Let E denote the event that there exists some j with $W_j \neq W_i$ but $L_j = L_i$. We may observe that $\Pr[\overline{\mathsf{E}}] \leq e_E + \ell/|\mathcal{X}|$. Let D' be the distribution D modified by projecting away all keys at positions j where $W_j = W_i$. Note that D' represents the distribution of key material for the projected version of H. Also, when conditioned on the event E, the distribution D' has the 'approximate twining property' used in the proof of Theorem A.4, and thus those results apply to the projected version of H. The desired result follows. □