This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE ICC 2009 proceedings

Error-Tolerant Searchable Encryption

Julien Bringer*
*Sagem Sécurité
Osny, France

Hervé Chabanne*†, Bruno Kindarji*†
†Institut Télécom, Télécom ParisTech
Paris, France

***Abstract*—In this paper, we describe a new primitive for Error-Tolerant Searchable Encryption and a security model for it. This generic scheme permits to make searches on encrypted data with only an approximation of some keyword. It enables to efficiently query secure databases in order to get the exact data with a close estimation of it. An application to biometric identification arises from this construction.**

This is the first construction both for Error-Tolerant Searchable Encryption and for a biometric identification protocol over encrypted personal data.

Keywords. Searchable encryption, Error tolerance, Public key cryptography.

I. Introduction

Searchable encryption is a primitive that permits to make queries on a set of encrypted data. This paper focuses on Error-tolerant searchable encryption, a new primitive that is motivated by the following application.

Biometrics are some natural information that every human being carries along, and that are sufficiently discriminatory to be used in security applications such as authentication or identification. Practically speaking, a biometric template is some information obtained from someone's physical characteristics or behaviour, thanks to a sensor. The following paragraph describes a realistic model for biometrics identification.

Let $B = \{0,1\}^N$ be the Hamming space. A biometric template $b \in B$ is the result of a measurement from someone's biometrics thanks to a sensor. Two different measures b, b' of the same user $\mathcal{U}$ are with high probability at a Hamming distance $d(b,b') \leq \lambda_{min}$; measures b_1, b_2 of different users $\mathcal{U}_1$,$\mathcal{U}_2$ are at a Hamming distance $d(b_1,b_2) > \lambda_{max}$. For example, the human iris can be represented on a 2048-bit long vector (Daugman's IrisCode, [1]), such that two independent IrisCodes are at a Hamming Distance of about 1024, whereas two IrisCodes coming from the same iris are significantly closer.

A biometric identification system recognizes a person among a collection of templates; such an application aims at recovering the identity of the person from his template. As biometrics are personal data, we think appropriate to protect the information stored on the server's side, as much as protecting the information that travels between the sensor and the service provider. A biometric identification protocol *over encrypted templates* consists in securely recognizing some biometric template b among reference elements b_i from a database. A trivial way to do so is to use authentication methods like [2]–[4] over the full database. We here introduce Error-Tolerant Searchable Encryption to directly tackle the issues raised by identification.

Work partially supported by french ANR RNRT project BACH.

A. Related Works

Searching over encrypted data was investigated by Song *et al.* [5] with a symmetrical cryptography setting. The Public Key Encryption with keyword Search primitive was introduced by Boneh *et al.* [6], and indexes encrypted documents using keywords. Some constructions [7]–[9] use different trapdoors for each keyword used in the indexing process, whereas [10] applies more computation-efficient methods. However, each of these constructions only focuses on exact keyword search. [11], [12] consider the use of wildcards to enlarge the range of keyword searched, but this technique only covers part of the possible close keywords, leaving the case for approximate keyword search unsolved.

Error-tolerant search over plain data is also a field that has already been investigated. [13] is a review on the subject, that presents the different options for the matter. We propose to focus on hashing-based methods as explored by [13]–[15]. [16] uses hashing for iris-based identification. We extend the scope of their paper to match privacy requirements through encrypting the templates.

To the extent of our knowledge, this paper presents the first description and implementation of an encryption scheme that is error-tolerant and searchable.

B. Contribution and organization of this paper

This paper defines a new cryptographic primitive, Public Key Error-Tolerant Searchable Encryption, that we link to the Public-key Encryption with Keyword Search defined in [6]. We define in Section II the security requirements that are needed for such a system to be trustworthy against powerful opponents that have access to the communications, the database, or that are malicious users.

We detail the different functions needed for our proposition in Section III. Moreover, we detail in Section IV a step-by-step construction of such a scheme. We prove that, within reasonable assumptions, this scheme fulfils the security requirements. Section V concludes.

II. Model Presentation

In this section, we describe a formal model for an Error-Tolerant Searchable Encryption scheme. Our thought process follows that of [10].

978-1-4244-3435-0/09/$25.00 ©2009 IEEE

In the sequel, we note $[\![m, n]\!]$ the set of all integers between m and n (inclusive).

A. Entities for the Protocol

Our primitive models the interactions between users that store and retrieve information, and a remote server. We distinguish the user who stores the data from the one who wants to get it. This leads to three entities:

- The server $\mathcal{S}$: a remote storage system. As the server is untrusted – it can be an outsourcing server, a free service offered by some mail provider – we consider the content to be public. Communications to and from this server are also subject to eavesdropping,
- The sender $\mathcal{X}$ incrementally creates the database, by sending data to $\mathcal{S}$,
- The receiver $\mathcal{Y}$ makes queries to the server $\mathcal{S}$.

We emphasize that $\mathcal{X}$ and $\mathcal{Y}$ are not necessarily the same user, as $\mathcal{X}$ has full knowledge of the database he created whereas $\mathcal{Y}$ knows only what he receives from $\mathcal{S}$.

B. Definition of the Primitives

In the sequel, messages are binary strings of a fixed length N, and $d(x_1, x_2)$ the Hamming Distance between $x_1, x_2 \in \{0,1\}^N$ is the canonical distance, *i.e.* the number of positions in $[\![1, N]\!]$ in which x_1 and x_2 differ.

Here comes a formal definition of the primitives that enable to perform an error-tolerant searchable encryption; this definition cannot be parted from the definition of Completeness(λ_{min}) and ϵ-Soundness(λ_{max}), which follows.

Definition 1: A $(\epsilon, \lambda_{min}, \lambda_{max})$-*Public Key Error-Tolerant Searchable Encryption* is obtained with the following probabilistic polynomial-time methods:

- $\mathsf{KeyGen}(1^k)$ initializes the system, and outputs public and private keys (pk, sk); k is the security parameter. The public key pk is used to store data on a server, and the secret key sk is used to retrieve information from that server.
- $\mathsf{Send}_{\mathcal{X},\mathcal{S}}(x, pk)$ is a protocol in which $\mathcal{X}$ sends to $\mathcal{S}$ the data $x \in \{0,1\}^N$ to be stored on the storage system. At the end of the protocol, $\mathcal{S}$ associated an identifier to x, noted $\varphi(x)$.
- $\mathsf{Retrieve}_{\mathcal{Y},\mathcal{S}}(x', sk)$ is a protocol in which, given a fresh data $x' \in \{0,1\}^N$, $\mathcal{Y}$ asks for the identifiers of all data that are stored on $\mathcal{S}$ and are close to x', with Completeness(λ_{min}) and ϵ-Soundness(λ_{max}). This outputs a set of identifiers, noted $\Phi(x')$.

These definitions are comforted by the condition 1 that defines Completeness and ϵ-Soundness for the parameters $\lambda_{min}, \lambda_{max}$, see Sec. II-C. In a few words, Completeness implies that a registered message x is indeed found if the query word x' is at a distance less than λ_{min} from x, while ϵ-Soundness means that with probability greater than $1 - \epsilon$, no message at a distance greater than λ_{max} from x' will be returned.

The Send protocol produces an output $\varphi(x)$ that identifies the data x. This output $\varphi(x)$ is meant to be a unique identifier, which is a binary string of undetermined length – in other words, elements of $\{0,1\}^\star$ – that enables to retrieve x. It can be a timestamp, a name or nickname, *etc.* depending on the application.

The most significant difference from the primitives introduced in [6] is that messages are no longer associated to keywords. Moreover, these primitives enable some imprecision on the message that is looked up. For example, one can imagine a mailing application, where all the mails are encrypted, and where it is possible to make queries on the mail subject. If there is a typo in the query, then looking for the correct word should also give the mail among the results – at least, we would like that to happen.

Note that wildcards are not well-adapted to this kind of application, as a wildcard permits to catch errors providing that we know where it is located, whereas error-tolerance does not have this constraint.

C. Security Requirements

First of all it is important that the scheme actually works, *i.e.* that the retrieval of a message near a registered one gives the correct result. This can be formalized into the following condition:

*Condition 1 (**Completeness(λ_{min}), ϵ-Soundness(λ_{max})**):* Let $x_1, \ldots, x_p \in B = \{0,1\}^N$ be p different binary vectors, and let $x' \in B$ be another binary vector. Suppose that the system was initialized, that all the messages x_i have been sent by user $\mathcal{X}$ to the system $\mathcal{S}$ with identifiers $\varphi(x_i)$, and that user $\mathcal{Y}$ retrieved the set of identifiers $\Phi(x')$ associated to x'.

1) The scheme is said to be **complete** if the identifiers of all the x_i that are near x' are in the resulting set $\Phi(x')$, *i.e.* if $\eta_c = \Pr_{x'}[\exists i,\, d(x', x_i) \leq \lambda_{min} \,\&\, \varphi(x_i) \notin \Phi(x')]$ is negligible.
2) The scheme is said to be **ϵ-sound** if $\eta_s = \Pr_{x'}[\exists i \in [\![1, p]\!],\, d(x', x_i) > \lambda_{max} \,\&\, \varphi(x_i) \in \Phi(x')]$ is bounded by ϵ.

The first condition simply means that registered data is effectively retrieved if the input is close. η_c expresses the probability of failure of this Retrieve operation.

The second condition means that only the close messages are retrieved, thus limiting false alarms. η_s measures the reliability of the Retrieve query, *i.e.* if all the results are identifiers of messages near to x'.

These two properties (*Completeness* and ϵ-*Soundness*) are sufficient to have a working set of primitives which allows to make approximate queries on a remote storage server. The following conditions, namely *Sender Privacy* and *Receiver Privacy*, ensure that the data stored in the server is secure, and that communications can be done on an untrusted network.

*Condition 2 (**Sender Privacy**):* The scheme is said to respect *Sender Privacy* if the advantage of any server is negligible in the $\mathsf{Exp}_{\mathcal{A}}^{\text{Sender Privacy}}$ experiment, described below. Here, $\mathcal{A}$ is an 'honest-but-curious' opponent taking the place of $\mathcal{S}$,

and $\mathcal{C}$ is a challenger at the user side.

$\mathsf{Exp}_{\mathcal{A}}^{\text{Sender Privacy}}$

1.	(pk, sk)	$\leftarrow$	$\mathsf{KeyGen}(1^k)$	$(\mathcal{C})$
2.	$\{x_2, \ldots, x_\Omega\}$	$\leftarrow$	$\mathcal{A}$	$(\mathcal{A})$
3.	$\varphi(x_i)$	$\leftarrow$	$\mathsf{Send}_{\mathcal{X},\mathcal{S}}(x_i, pk)$	$(\mathcal{C})$
4.	$\{x_0, x_1\}$	$\leftarrow$	$\mathcal{A}$	$(\mathcal{A})$
5.	$\varphi(x_e)$	$\leftarrow$	$\mathsf{Send}_{\mathcal{X},\mathcal{S}}(x_e, pk)$ $e \in_R \{0,1\}$	$(\mathcal{C})$
6.	Repeat steps $(2, 3)$			
7.	$e' \in \{0,1\}$	$\leftarrow$	$\mathcal{A}$	$(\mathcal{A})$

The advantage of the adversary is $|\Pr[e' = e] - \frac{1}{2}|$.

Informally, in a first step, the adversary receives Send requests that he chose himself; $\mathcal{A}$ then looks for a couple (x_0, x_1) of messages on which he should have an advantage. $\mathcal{C}$ chooses one of the two messages, and the adversary must guess, by receiving the Send requests, which one of x_0 or x_1 it was.

This condition permits to have privacy on the content stored on the server. The content that the sender transmits is protected, justifying the title 'Sender Privacy'.

Another important privacy aspect is the secrecy of the data that is retrieved. We do not want the server to have information on the fresh data x' that is queried; this is expressed by the following condition.

*Condition 3 (**Receiver Privacy**):* The scheme is said to respect *Receiver Privacy* if the advantage of any server is negligible in the $\mathsf{Exp}_{\mathcal{A}}^{\text{Receiver Privacy}}$ experiment described below. As in the previous condition, $\mathcal{A}$ denotes the 'honest-but-curious' opponent taking the place of $\mathcal{S}$, and $\mathcal{C}$ the challenger at the user side.

$\mathsf{Exp}_{\mathcal{A}}^{\text{Receiver Privacy}}$

1.	(pk, sk)	$\leftarrow$	$\mathsf{KeyGen}(1^k)$	$(\mathcal{C})$
2.	$\{x_1, \ldots, x_\Omega\}$ $d(x_i, x_j) > \lambda_{max}, \forall i, j \in [\![1, \Omega]\!]$	$\leftarrow$	$\mathcal{A}$	$(\mathcal{A})$
3.	$\varphi(x_i), (i \in [\![1, \Omega]\!])$	$\leftarrow$	$\mathsf{Send}_{\mathcal{X},\mathcal{S}}(x_i, pk)$	$(\mathcal{C})$
4.	$\{x'_2, \ldots, x'_p\}$	$\leftarrow$	$\mathcal{A}$	$(\mathcal{A})$
5.	$\Phi(x'_j), (j \in [\![2, p]\!])$	$\leftarrow$	$\mathsf{Retrieve}_{\mathcal{Y},\mathcal{S}}(x'_j, sk)$	$(\mathcal{C})$
6.	(x'_0, x'_1)	$\leftarrow$	$\mathcal{A}$	$(\mathcal{A})$
7.	$\Phi(x'_e)$	$\leftarrow$	$\mathsf{Retrieve}_{\mathcal{Y},\mathcal{S}}(x'_e, sk)$ $e \in_R \{0,1\}$	$(\mathcal{C})$
8.	Repeat steps $(4, 5)$			
9.	$e' \in \{0,1\}$	$\leftarrow$	$\mathcal{A}$	$(\mathcal{A})$

The advantage of the adversary is $|\Pr[e' = e] - \frac{1}{2}|$.
This condition is the mirror image of the previous one. It transposes the idea that the receiver $\mathcal{Y}$ can make his queries to $\mathcal{S}$ without leaking information on their content. The processing of the experiment is the same as the Sender Privacy experiment, except that $\mathcal{A}$ has to distinguish between Retrieve queries instead of Send queries.

Remark 1: Conditions 2 and 3 are the transposition of their homonym statement in [10]. They aim for the same goal, *i.e.* privacy – against the server – of the data that is registered first, then looked for.

Section IV is dedicated to give a construction that fits these security conditions.

III. Useful tools

We now describe a few utilities that we use in our construction, namely Locality-Sensitive Hashing, Bloom Filters along with their extensions, and Private Information Retrieval Protocols.

A. Locality-Sensitive Hashing

Our fuzzy search problem can be related to the *approximate nearest neighbour* (**ANN**) problem (see *e.g.* [13]). To solve this problem, a possible lead is to look for hash functions (not cryptographic ones) that give the same result for near points, as defined in [17]:

Definition 2 (LSH, [17]): Let B be a metric space, U a set with a smaller dimensionality, $\lambda_{min}, \lambda_{max} \in \mathbb{R}$ with $\lambda_{min} < \lambda_{max}$, $\epsilon_1, \epsilon_2 \in [0, 1]$ with $\epsilon_1 > \epsilon_2$. A family $H = \{h_1, \ldots, h_\mu\}, h_i : B \to U$, is $(\lambda_{min}, \lambda_{max}, \epsilon_1, \epsilon_2)$-*LSH (Locality-Sensitive Hashing)*, if for all $x, x' \in B$, $\Pr_H[h(x) = h(x')] > \epsilon_1$ (resp. $\Pr_H[h(x) = h(x')] < \epsilon_2$) if $d(x, x') < \lambda_{min}$ (resp. $d(x, x') > \lambda_{max}$).

Such functions reduce the differences occurring between similar data with high probability, whereas distant data should remain significantly remote.

A noticeable example of a LSH family was proposed by Kushilevitz *et al.* in [14]; see also [15], [17], [18].

B. Bloom Filters

As introduced by Bloom in [19], a set of Bloom filters is a data structure used for answering set membership queries. We work here with the *Bloom filters with storage* (**BFS**) defined in [10] as an extension of Bloom filters. Their aim is to give not only the result of the set membership test, but also an index associated to the element. The iterative definition below introduces these objects and the notion of *tags* and *buckets* which are used in the construction.

Definition 3 (Bloom Filter with Storage, [10]): Let D be a finite subset of a set Y. For a collection of ν hash functions $H' = \{h'_1, \ldots, h'_\nu\}$, with each $h'_j : Y \to [\![1, m]\!]$, a set V of *tags* associated to D with a *tagging* function $\psi : D \to \mathcal{P}(V)$. A (ν, m)-*Bloom Filter with Storage* is H', together with an array of subsets $(T_1, \ldots, T_m)$ of V, called *buckets*, iteratively defined as:

1) $\forall i \in [\![1, m]\!], T_i \leftarrow \emptyset$,
2) $\forall y \in D, \forall j \in [\![1, \nu]\!]$, update the bucket T_α with $T_\alpha \leftarrow T_\alpha \cup \psi(y)$ where $\alpha = h'_j(y)$.

In other words, the bucket structure is empty at first, and for each element $y \in D$ to be indexed, we add to the bucket T_α all the tags associated to y. Construction of such a structure is illustrated in Fig. 1.

Example 1: In Fig. 1, assume that $D = \{y_1, y_2, y_3\}$ and $\nu = 3$, the tags associated to y_1 (resp. y_2) have already been incorporated into the buckets T_2, T_3 and T_α (resp. T_1, T_2 and T_3) so that $T_1 = \{\psi(y_2)\}$, $T_2 = T_3 = \{\psi(y_1), \psi(y_2)\}$,

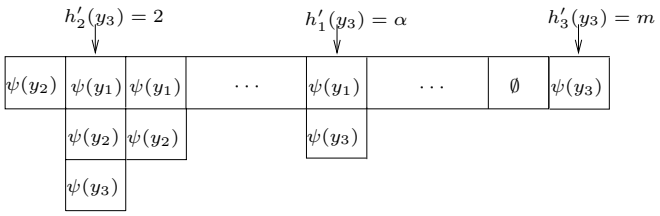

Figure 1. Construction of Bloom Filters with Storage

$T_\alpha = \{\psi(y_1)\}$ and $T_i = \emptyset$ otherwise. We are now treating the case of y_3:

- $h'_1(y_3) = \alpha$ so $T_\alpha \leftarrow T_\alpha \cup \{\psi(y_3)\}$, i.e. $T_\alpha = \{\psi(y_1), \psi(y_3)\}$;
- $h'_2(y_3) = 2$ so $T_2 \leftarrow T_2 \cup \{\psi(y_3)\}$, i.e. $T_2 = \{\psi(y_1), \psi(y_2), \psi(y_3)\}$;
- $h'_3(y_3) = m$ so $T_m \leftarrow T_m \cup \{\psi(y_3)\}$, i.e. $T_m = \{\psi(y_3)\}$.

This construction enables to retrieve a set of tags associated to an element $y \in D$: it is designed to obtain $\psi(y)$, the set of tags associated to y, by computing $\bigcap_{j=1}^{\nu} T_{h'_j(y)}$. For instance, in the previous example, $\bigcap_{j=1}^{\nu} T_{h'_j(y_3)} = T_2 \cap T_\alpha \cap T_m = \{\psi(y_3)\}$. This intersection may capture inappropriate tags, but the choice of relevant hash functions and increasing their number allow to reduce the probability of that event. These properties are summed up in the following lemma.

Lemma 1 ([19]): Let $(H', T_1, \ldots, T_m)$ be a (ν, m)-Bloom filter with storage indexing a set D with tags from a tag set V. Then, for $y \in D$, the following properties hold:

- $\psi(y) \subset T(y) = \bigcap_{j=1}^{\nu} T_{h'_j(y)}$, *i.e.* each of y's tag is retrieved,
- the probability for a false positive $t \in V$ is $\Pr[t \in T(y) \text{ and } t \notin \psi(y)] = \left(1 - \left(1 - \frac{\nu}{m}\right)^{|D|}\right)^{\nu}$.

C. Combining BFS and LSH

We want to apply Bloom filters to data that are very likely to vary. To this aim, we first apply LSH-families as input to Bloom filters.

We choose μ hash functions from an adequate LSH family $h_1, \ldots, h_\mu : B \to \{0,1\}^t$, and ν hash functions dedicated to a Bloom filter with Storage $h'_1, \ldots, h'_\nu : \{0,1\}^t \times [\![1, \mu]\!] \to [\![1, m]\!]$. The LSH family is denoted H, and H' is the BFS one. To obtain a BFS with locality-sensitive functionality, we use composite $\mu \times \nu$ hash functions induced by both families.

More precisely, we first compute from an element x in the space B the locally-reduced $c_1 = h_1(x), \ldots, c_\mu = h_\mu(x)$. Thanks to the LSH property, we know that, with high probability, the locally-reduced c'_i obtained from a close x' perfectly match. It is now possible to apply the BFS functions to c_i, more precisely, to $c_i || i$ where $||$ is the concatenation operator.

We define $h^c_{(j,i)} : B \to [\![1, m]\!]$ the corresponding composite functions (c stands for composite) with $h^c_{(j,i)}(y) = h'_j(h_i(y) \parallel i)$. Let $H^c = \{h^c_{(j,i)}, (j,i) \in [\![1, \nu]\!] \times [\![1, \mu]\!]\}$ the set of all these functions.

To sum up, we modify the update of the buckets in Def. 3 by $\alpha = h'_j(h_i(y) \parallel i)$. Later on, to recover tags related to an approximate query $x' \in B$, all we have to consider is $\bigcap_{j=1}^{\nu} \bigcap_{i=1}^{\mu} T_{h'_j(h_i(x') \| i)}$. Indeed, if x and x' are close enough, then the LSH functions give the same results on x and x', effectively providing a Bloom filter with storage that has the LSH property. This property is numerically estimated in the following lemma:

Lemma 2: Let H, H', H^c be families constructed in this setting. Let $x, x' \in B$ be two binary vectors. Assume that H is $(\lambda_{min}, \lambda_{max}, \epsilon_1, \epsilon_2)$-LSH from B to $\{0,1\}^t$; assume that H' is a family of ν pseudo-random hash functions. If the tagging function ψ associates only one tag per element, then the following properties stand:

1) If x and x' are far enough, then except with a small probability, $\psi(x')$ does not intersect all the buckets that index x, *i.e.* $\Pr_{x'}\left[\psi(x') \subset \bigcap_{h^c \in H^c} T_{h^c(x)} \text{ and } d(x,x') \geq \lambda_{max}\right] \leq \left(\epsilon_2 + (1-\epsilon_2)\frac{1}{m}\right)^{|H^c|}$,
2) If x and x' are close enough, then except with a small probability, $\psi(x')$ is in all the buckets that index x', *i.e.* $\Pr_{x'}\left[\psi(x') \not\subset \bigcap_{h^c \in H^c} T_{h^c(x)} \text{ and } d(x,x') \leq \lambda_{min}\right] \leq 1 - (1-\epsilon_1)^{|H^c|}$.

Note that this lemma used the simplified hypothesis that $\forall x, |\psi(x)| = 1$, which means that there is only one tag per vector. This has a direct application in Section IV-B. In practice, $\psi(x)$ can be a unique handle for x.

Sketch of proof. The first part of the lemma expresses the fact that if $d(x,x') \geq \lambda_{max}$, due to the composition of a LSH function with a pseudorandom function, the collision probability is $\frac{1}{m}$. Indeed, if $h'_1(y_1) = h'_2(y_2)$, either $y_1 = y_2$ and $h'_1 = h'_2$, or there is a collision of two independent pseudo-random hash function. In our case, if $y_1 = y_2$, that means that $y_1 = h_{i_1}(x)||i_1$ and $y_2 = h_{i_2}(x')||i_2$. To these vectors to be the same, $i_1 = i_2$ and $h_{i_1}(x) = h_{i_2}(x')$, which happens with probability ϵ_2.

The second part of the lemma says that for each $h^c \in H^c$, $h^c(x)$ and $h^c(x')$ are the same with probability $1 - \epsilon_1$. Combining the incremental construction of the T_i with this property gives the lemma. □

D. Private Information Retrieval Protocols

To enable privacy-ensuring queries to databases, we use the primitive called Private Information Retrieval protocol (PIR, [20]). Its goal is to retrieve a specific information from a remote server in such a way that he does not know which data was sent. This is done through a method $\mathsf{Query}^{PIR}_{\mathcal{Y},\mathcal{S}}(a)$, that allows $\mathcal{Y}$ to recover the element stored at index a in $\mathcal{S}$ by running the PIR protocol.

Suppose a database $\mathcal{DB}$ is constituted with M bits $X = x_1, ..., x_M$. To be secure, the protocol should satisfy the following properties [21]:

- **Soundness:** When the user and the database follow the protocol, the result of the request is exactly the requested bit.
- **User Privacy:** For all $X \in \{0,1\}^M$, for $1 \leq i, j \leq M$, for any polynomial time algorithm used by the database,

it cannot distinguish with a non-negligible probability the difference between the requests of index i and j.

Among the known constructions of computational secure PIR, block-based PIR – i.e. working on block of bits – allows to efficiently reduce the cost. The best performances are from Gentry and Ramzan [22] and Lipmaa [23] with a communication complexity polynomial in the logarithm of M. Surveys of the subject are available in [24], [25].

Remark 2: PIR protocols enable to retrieve information of a database. A Private Information Storage protocol [25] is a protocol that enables to write information in a database with properties that are similar to that of PIR. The goal is to prevent the database from knowing the content of the information that is being stored; for detailed description of such protocols, see [10], [26].

Such a protocol provides a method $\mathsf{update}(val, index)$, which takes as input an element and a database index, and puts the value val into the database entry $index$. To be secure, the protocol must also satisfy the Soundness and User Privacy properties, meaning that 1. $\mathsf{update}_{\mathsf{BF}}$ does update the database with the appropriate value, and 2. any algorithm run by the database cannot distinguish between the writing requests of (val_i, ind_i) and (val_j, ind_j).

IV. Our construction

A. Technical Description

Our searching scheme uses all the tools we described in the previous section. As we will see in section IV-B, this enables to meet the privacy requirements of section II-C. More precisely:

- We pick a family H' of functions: $h' : \{0,1\}^t \times [\![1, |H|]\!] \to [\![1, m]\!]$, adapted to a Bloom filter structure,
- We choose a family H of functions: $h : \{0,1\}^N \to \{0,1\}^t$ that have the LSH property,
- From these two families, we deduce a family H^c of functions $h^c : \{0,1\}^N \to [\![1, m]\!]$ as specified in Sec. III-C,
- We use a semantically secure public key cryptosystem $(\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$ [27],
- We use a PIR protocol with query function $\mathsf{Query}^{PIR}_{\mathcal{Y},\mathcal{S}}$.
- We use a PIS function $\mathsf{update}_{\mathsf{BF}}(val, i)$ that adds val to the i-th bucket of the Bloom filter, see Remark 2.

Here come the details of the implementation. In a few words, storage and indexing of the data are separated, so that it becomes feasible to search over the encrypted documents. Indexing is made thanks to Bloom Filters, with an extra precaution of encrypting the content of all the buckets. Finally, using our locality-sensitive hashing functions permits error-tolerance.

1) System setup: The method $\mathsf{KeyGen}(1^k)$ initializes m different buckets to $\emptyset$. The public and secret keys of the cryptosystem (pk, sk) are generated by $\mathsf{Setup}(1^k)$, and sk is given to $\mathcal{Y}$.

2) Sending a message: The protocol $\mathsf{Send}_{\mathcal{X},\mathcal{S}}(x, pk)$ goes through the following steps:

1) **Identifier establishment** $\mathcal{S}$ attributes to x a unique identifier $\varphi(x)$, and sends it to $\mathcal{X}$.
2) **Data storage** $\mathcal{X}$ sends $\mathsf{Enc}(x)$ to $\mathcal{S}$, who stores it in a memory cell that depends on $\varphi(x)$.
3) **Data indexing** $\mathcal{X}$ computes $h^c(x)$ for all $h^c \in H^c$, and executes $\mathsf{update}_{\mathsf{BF}}(\mathsf{Enc}(\varphi(x)), h^c(x))$ to send $\mathsf{Enc}(\varphi(x))$ to be added to the filter's bucket of index $h^c(x)$ on the server side.

Note that for privacy concerns, we complete the buckets with random data in order to get the same bucket size l for the whole data structure.

The first phase (identifier establishment) is done to create an identifier that can be used to register and then retrieve x from the database. For example, $\varphi(x)$ can be the time at which $\mathcal{S}$ received x, or the first memory address that is free for the storage of $\mathsf{Enc}(x)$.

The third phase applies the combination of BFS and LSH functions (see Sec. III-C) to x so that it is possible to retrieve x with some approximate data. This is done with the procedure described hereafter.

3) Retrieving data: The protocol $\mathsf{Retrieve}_{\mathcal{Y},\mathcal{S}}(x', sk)$ goes through the following steps:

1) $\mathcal{Y}$ computes each $\alpha_i = h^c_i(x')$ for each $h^c_i \in H^c$, then executes $\mathsf{Query}^{PIR}_{\mathcal{Y},\mathcal{S}}(\alpha_i)$ to receive the filter bucket T_{α_i},
2) $\mathcal{Y}$ decrypts the content of each bucket T_{α_i} and computes the intersection of all the $\mathsf{Dec}(T_{\alpha_i})$,
3) This intersection is a set of identifiers $\{\varphi(x_{i_1}), \ldots, \varphi(x_{i_\gamma})\}$, which is the result of the execution of $\mathsf{Retrieve}$.

As we can see, the retrieving process follows that of Sec. III-C, with the noticeable differences that 1. the identifiers are always encrypted in the database, and 2. the query is made with a PIR. This permits to benefit from both the Bloom filter structure, the locality-sensitive hashing, and the privacy-preserving protocols.

The secure protocols involved do not leak information on the requests that the client makes, and the next section discusses more precisely the security properties achieved.

B. Security Properties

We now demonstrate that this construction faithfully achieves the security requirements we defined in Sec. II-C.

*Proposition 1 (**Completeness**):* Provided that H is a $(\lambda_{min}, \lambda_{max}, \epsilon_1, \epsilon_2)$-LSH family, for a negligible ϵ_1, this scheme is complete.

*Proposition 2 (ϵ-**Soundness**):* Provided that H is a $(\lambda_{min}, \lambda_{max}, \epsilon_1, \epsilon_2)$-LSH family from $\{0,1\}^N$ to $\{0,1\}^t$, and provided that the Bloom filter functions H' behave like pseudo-random functions from $\{0,1\}^t \times [\![1, |H|]\!]$ to $[\![1, m]\!]$, then the scheme is ϵ-sound, with:

$$\epsilon = \left(\epsilon_2 + (1-\epsilon_2)\frac{1}{m}\right)^{|H^c|}$$

Propositions 1 and 2 are direct consequence of Lemma 2.

Remark 3: Proposition 2 assumes that the Bloom filter hash functions are pseudo-random; this hypothesis is pretty standard for Bloom filter analysis. It can be achieved by using, as

Bloom filter hash functions, cryptographic functions with a random oracle-like behaviour.

*Proposition 3 (**Sender Privacy**):* Assume that the underlying cryptosystem is semantically secure and that the PIS function update_{BF} achieves User Privacy, then the scheme ensures Sender Privacy.

Proof. If the scheme does not ensure Sender Privacy, that means that there exists an attacker who can distinguish between the output of $\mathsf{Send}(x_0, pk)$ and $\mathsf{Send}(x_1, pk)$, after the execution of $\mathsf{Send}(x_i, pk)$, $i \in [\![2, \Omega]\!]$.

Note that the content of the Bloom filter buckets does not reveal information that can permit to distinguish between x_0 and x_1. Indeed, the only information $\mathcal{A}$ has with the filter structure is a set of $\mathsf{Enc}(\varphi(x_i))$ placed at different indexes $h^c(x_i)$, $i = e, 2, \ldots, \Omega$. Thanks to the semantic security of Enc, this does not permit to distinguish between $\varphi(x_0)$ and $\varphi(x_1)$. This implies that, with inputs $\mathsf{Enc}(x_i)$, $\mathsf{update}_{\mathsf{BF}}(\mathsf{Enc}(\varphi(x_i)), h^c(x_i))$ (for $i \geq 2$), the attacker can distinguish between $\mathsf{Enc}(x_0)$, $\mathsf{update}_{\mathsf{BF}}(\mathsf{Enc}(\varphi(x_0)), h^c(x_0))$ and $\mathsf{Enc}(x_1)$, $\mathsf{update}_{\mathsf{BF}}(\mathsf{Enc}(\varphi(x_1)), h^c(x_1))$.

As $\mathsf{update}_{\mathsf{BF}}$ does not leak information on its inputs, that means that the attacker can distinguish between $\mathsf{Enc}(x_0)$ and $\mathsf{Enc}(x_1)$ by choosing some other inputs to Enc. That contradicts the semantic security assumption. □

*Proposition 4 (**Receiver Privacy**):* Assume that the PIR ensures User Privacy, then the scheme ensures Receiver Privacy.

Proof.

This property is a direct deduction of the PIR's User Privacy, as the only information $\mathcal{S}$ gets from the execution of a Retrieve is a set of Query^{PIR}. □

These properties show that this protocol for Error-Tolerant Searchable Encryption has the security properties that we looked for. LSH functions are used in such a way that they do not degrade the security properties of the system.

V. Conclusion

In this paper, we propose primitives for Error-Tolerant Searchable Encryption. We also describe an adversarial model that is adapted to error-tolerance, and that is more demanding than previous models for Searchable Encryption.

We finally present a construction for such a scheme. Starting from a Bloom filter, we tweak the hashing functions to also be locality sensitive, and we enhance the sending and querying protocol to gain the adequate security properties. Our construction does match the requirements we imposed.

An immediate application of this paper is biometric identification over encrypted templates. The evaluation of the performances of our scheme on existing databases is the subject of our future works.

References

[1] J. Daugman, "High confidence visual recognition of persons by a test of statistical independence." *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 15, no. 11, pp. 1148–1161, 1993.

[2] J. Bringer, H. Chabanne, M. Izabachène, D. Pointcheval, Q. Tang, and S. Zimmer, "An application of the Goldwasser-Micali cryptosystem to biometric authentication." in *ACISP*, ser. LCNS, J. Pieprzyk, H. Ghodosi, and E. Dawson, Eds., vol. 4586. Springer, 2007, pp. 96–106.

[3] J. Bringer, H. Chabanne, D. Pointcheval, and Q. Tang, "Extended private information retrieval and its application in biometrics authentications," in *CANS*, ser. LCNS, vol. 4856. Springer, 2007, pp. 175–193.

[4] B. Schoenmakers and P. Tuyls, "Efficient binary conversion for Paillier encrypted values." in *EUROCRYPT*, vol. 4004. Springer, 2006, pp. 522–537.

[5] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data." in *IEEE Symposium on Security and Privacy*, 2000, pp. 44–55.

[6] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search." in *EUROCRYPT*, ser. LCNS, C. Cachin and J. Camenisch, Eds., vol. 3027. Springer, 2004, pp. 506–522.

[7] E.-J. Goh, "Secure indexes," Cryptology ePrint Archive, Report 2003/216, 2003, http://eprint.iacr.org/2003/216/.

[8] M. Abdalla, M. Bellare, D. Catalano, E. Kiltz, T. Kohno, T. Lange, J. Malone-Lee, G. Neven, P. Paillier, and H. Shi, "Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions." in *CRYPTO*, vol. 3621. Springer, 2005, pp. 205–222.

[9] D. Khader, "Public key encryption with keyword search based on K-resilient IBE." in *ICCSA (3)*, vol. 3982. Springer, 2006, pp. 298–308.

[10] D. Boneh, E. Kushilevitz, R. Ostrovsky, and W. E. Skeith III, "Public key encryption that allows PIR queries." in *CRYPTO*, vol. 4622. Springer, 2007, pp. 50–67.

[11] M. Abdalla, E. Kiltz, and G. Neven, "Generalized key delegation for hierarchical identity-based encryption." in *ESORICS*, vol. 4734. Springer, 2007, pp. 139–154.

[12] J. Birkett, A. W. Dent, G. Neven, and J. C. N. Schuldt, "Efficient chosen-ciphertext secure identity-based encryption with wildcards." in *ACISP*, ser. LCNS, J. Pieprzyk, H. Ghodosi, and E. Dawson, Eds., vol. 4586. Springer, 2007, pp. 274–292.

[13] P. Indyk, "Nearest neighbors in high-dimensional spaces," in *Handbook of Discrete and Computational Geometry, chapter 39*. CRC Press, 2004, 2rd edition.

[14] E. Kushilevitz, R. Ostrovsky, and Y. Rabani, "Efficient search for approximate nearest neighbor in high dimensional spaces." in *Symposium on the Theory Of Computing*, 1998, pp. 614–623.

[15] A. Andoni and P. Indyk, "Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions." *Commun. ACM*, vol. 51, no. 1, pp. 117–122, 2008.

[16] F. Hao, J. Daugman, and P. Zielinski, "A fast search algorithm for a large fuzzy database," *IEEE Trans. on Inf. Forensics and Security*, 2008.

[17] P. Indyk and R. Motwani, "Approximate nearest neighbors: Towards removing the curse of dimensionality." in *Symposium on the Theory Of Computing*, 1998, pp. 604–613.

[18] A. Kirsch and M. Mitzenmacher, "Distance-sensitive Bloom filters," in *Algorithm Engineering & Experiments*, Jan 2006.

[19] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors." *Commun. ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[20] B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan, "Private information retrieval." *J. ACM*, vol. 45, no. 6, pp. 965–981, 1998.

[21] Y. Gertner, Y. Ishai, E. Kushilevitz, and T. Malkin, "Protecting data privacy in private information retrieval schemes." in *STOC*, 1998, pp. 151–160.

[22] C. Gentry and Z. Ramzan, "Single-database private information retrieval with constant communication rate," in *ICALP*, ser. LCNS, vol. 3580. Springer, 2005, pp. 803–815.

[23] H. Lipmaa, "An oblivious transfer protocol with log-squared communication." in *ISC*, vol. 3650. Springer, 2005, pp. 314–328.

[24] W. I. Gasarch, "A survey on private information retrieval," http://www.cs.umd.edu/ gasarch/pir/pir.html.

[25] R. Ostrovsky and V. Shoup, "Private information storage (extended abstract)." in *STOC*, 1997, pp. 294–303.

[26] R. Ostrovsky and W. E. Skeith III, "Algebraic lower bounds for computing on encrypted data," Cryptology ePrint Archive, Report 2007/064, 2007, http://eprint.iacr.org/.

[27] S. Goldwasser and S. Micali, "Probabilistic encryption." *J. Comput. Syst. Sci.*, vol. 28, no. 2, pp. 270–299, 1984.

[28] J. Pieprzyk, H. Ghodosi, and E. Dawson, Eds., *Information Security and Privacy, 12th Australasian Conference, ACISP 2007, Proceedings*, ser. LCNS, vol. 4586. Springer, 2007.