Human-centric Computing and Information Sciences
a SpringerOpen Journal

RESEARCH Open Access

Embedding edit distance to enable private keyword search

Julien Bringer[1] and Hervé Chabanne[2*]

* Correspondence: herve.chabanne@morpho.com
[2]Morpho & Télécom ParisTech, Paris, France
Full list of author information is available at the end of the article

Abstract

Background: Our work is focused on fuzzy keyword search over encrypted data in Cloud Computing.

Methods: We adapt results on private identification schemes by Bringer *et al*. to this new context. We here exploit a classical embedding of the edit distance into the Hamming distance.

Results: Our way of doing enables some flexibility on the tolerated edit distance when looking for close keywords while preserving the confidentiality of the queries.

Conclusion: Our proposal is proved secure in a security model taking into account privacy.

Keywords: Edit distance, Embeddings for edit distance, Private Identification schemes

Introduction

Cloud Computing enables users to have access to shared resources somewhere on the Internet. At least, some storage capacities can easily be envisaged. This brings many sensitive information in the Cloud where they should stay, to preserve their confidentiality, encrypted. To look at their content remotely (and without decrypting them), some specific procedures have been developed. Searchable encryption [1] builds up an index for each keyword of interest. This way, a user can search over his encrypted data for such a keyword and retrieve the files containing it. Note that this search should be made with great care, for privacy reasons, in order for the Cloud to not be able to find out what is the underlying keyword. Symmetric Searchable Encryption (SSE) as introduced by [2] relies on symmetric encryption primitives for efficiency reasons. In [3], Li *et al*. build on SSE for a solution for fuzzy keyword search over encrypted data in Cloud Computing. The fuzziness should here be understood as minor typos introduced by users when entering the request through their keyboard. In this context, the edit distance (Levenshtein distance) is relevant to measure the strings similarity.

Related works

Considers two different techniques: wildcard-based and gram-based techniques [3], for achieving fuzzy keyword search over encrypted data. These two methods build a set consisting of the searched keyword and the nearby words according to the used technique. For instance, for the keyword CASTLE, the fuzzy keyword set for wildcard-based

technique consists of {CASTLE, *CASTLE, *ASTLE, C*ASTLE, C*STLE, ..., CASTL*E, CASTL*, CASTLE*} (respectively {CASTLE, CSTLE, CATLE, CASLE, CASTE, CASTL, ASTLE} for the gram-based technique) for an edit distance of 1. The idea behind these fuzzy keyword sets is to index - before the search phase - the exact keywords but also the ones differing slightly according to a fixed bound on the tolerated edit distance.

Our approach is somewhat different. For iriscode biometric data, the comparison of two iriscodes is made thanks to the computation of an Hamming distance [4]. There is today a trend to generalize this way of performing biometric matching for other modalities [5,6] for easier embedding into cryptographic protocols. In their works on private identification, Bringer *et al.* [7-9] (see also Section **Private identification schemes**) actually show how to carry out fuzzy keyword search for the Hamming distance. Following this trend, our idea is to combine this with a classical embedding of edit distance into the Hamming distance [10,11] (see Section **Edit distance approximation**) to obtain a fuzzy keyword search for the edit distance. This way of doing has at least two advantages. Firstly, contrary to [3] our way of proceeding does not need to a priori define the set of words which are considered as acceptable for the search. Moreover, we inherit of the security properties of [7] in their security model. Note that our proposal thus relies on an asymmetric security model. This can be seen as an asset for Cloud Computing applications. Indeed, using public-key encryption seems relevant in this context. To the best of our knowledge, this is the first scheme enabling fuzzy search with respect to edit distance over data encrypted with a public-key scheme.

Contribution and organization

The main contribution of this work is the proposal for a fuzzy keyword search over encrypted data where fuzzy means that we tolerate some edit distance deviation. A natural application of our results is Cloud Computing. We give proofs for the security properties of our scheme. We also discuss briefly and give some elements about its performances.

In the next Section, we briefly describe classical cryptographic primitives that we use. In Section **Model presentation**, we present our security model. In Section **Useful technical tools**, we recall some already published works on private identification schemes and the embedding of edit distances into the Hamming distance. In Section **Our construction**, we introduce our work and explain its properties.

Cryptographic primitives

Private information retrieval protocol

A Private Information Retrieval protocol (PIR, [12]) is a scheme that enables to retrieve a specific information from a remote server in such a way that the latter does not learn information about the query.

Suppose a database is constituted with M bits $\times = x_1,...,x_M$. To be secure, the protocol should satisfy the following properties [13]:

- **Soundness:** When the user and the database follow the protocol, the result of the request is exactly the requested bit.
- **User Privacy:** For all $\times \in \{0,1\}^M$, for $1 \leq i,j \leq M$, for any 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 [14] and Lipmaa [15] with a communication complexity polynomial in the logarithm of M. Surveys of the subject are available in [16,17].

Some PIR protocols are called Symmetric Private Information Retrieval, when they comply with the **Data Privacy** requirement [13]. This condition states that the querier cannot distinguish between a database that possesses only the information he requested, and a regular one; in other words, that the querier does not get more information than he asked for.

Private information storage protocol

PIR protocols enable to retrieve information of a database. A Private Information Storage (PIS) protocol [17] 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 [1,18].

To be secure, the protocol must also satisfy the Soundness and User Privacy properties, meaning that 1. following the protocol results in the update of the database with the appropriate value, and 2. any algorithm run by the database cannot distinguish between two writing requests.

Model presentation

In this section, we introduce the model of security for an Error-Tolerant Searchable Encryption scheme for edit distance by adapting the model from [7].

Entities for the protocol

The context is Cloud Computing where users can either store or retrieve data from the Cloud. This leads to three different entities:

- The Cloud $\mathcal{CL}$ which represents a single point of access to remote shared resources (i.e. a remote storage system). The Cloud is assumed to be untrusted, so we consider the content as publicly accessible to a third party and that communications in the Cloud and with users can be eavesdropped.
- The sender $\mathcal{X}$ sends data to be stored on the Cloud $\mathcal{CL}$.
- The receiver $\mathcal{Y}$ generates queries to the Cloud $\mathcal{CL}$ to obtain the results of his searches.

Note that the sender and the receiver are not necessarily the same user and it is even possible that several senders and several receivers exist and interact. This corresponds well to the Cloud Computing model.

Definition of the primitives

In the sequel, messages are strings of length N, and $ed(m_1m_2)$ denotes the edit distance between $m_1,m_2 \in \{0,1\}^N$, i.e. the minimum number of character insertions, deletions and substitutions needed to transform one string into the other. Note that edit distance is well defined on larger alphabet and variable length strings. The scheme can be extended to these cases.

To enable error-tolerant searchable encryption, we need three main primitives: the key materials generation, the send request and the receive request.

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

• KeyGen(1^{ℓ}) initializes the system, and generates public and private keys (pk,sk) for a security parameter ℓ. The public key pk is used to store data in the Cloud, and the secret key sk is used to retrieve information.

• $\text{Send}_{\mathcal{X},\mathcal{CL}}(m, pk)$ is a protocol in which $\mathcal{X}$ sends to $\mathcal{CL}$ the data m $\in \{0,1\}^N$ to be stored in the Cloud. At the end of the protocol, $\mathcal{CL}$ has stored the message m at a virtual address noted $\phi(m)$.

• $\text{Retrieve}_{,\mathcal{CL}}(m\prime, sk)$ is a protocol in which, given a fresh message m' $\in \{0,1\}^N$, $\mathcal{Y}$ asks for the virtual addresses of all data that are stored on $\mathcal{CL}$ and are close to m', with respect to the Completeness(λ_{min}) and Soundness(λ_{max}) criteria (cf. Section **Security requirements**). This outputs a set of virtual addresses, noted $\Phi(m')$, where $\mathcal{Y}$ can reach the corresponding messages.

Completeness and Soundness criteria for the parameters λ_{min}, λ_{max} represent the fact that a stored message will be actually retrieved if m' is at an edit distance less than λ_{min} and that no message at a distance greater than λ_{max} from m' will be returned (with a given non negligible probability). We emphasize that the definition above is focused on the searching problem (which is the tough task here): the algorithms' outputs are the virtual addresses where the retriever $\mathcal{Y}$ can retrieve the messages. The messages are possibly stored encrypted via a second encryption scheme.

An important difference compared to [3] is that we do not rely on fuzzy keyword sets, we want to ensure a given tolerance (materialized by λ_{min}, λ_{max}). By avoiding wildcards and grams, we do not make any prior assumption on the location of the errors.

Security requirements

We first recall the completeness and soundness criteria that formalized the condition for the scheme and the Cloud to actually return the correct answer.

Condition 1. ***Completeness***(λ_{min})***, Soundness***(λ_{max}) Let m_1, ..., $m_p \in \{0,1\}^N$ be p different binary strings, and let m'$\in \{0,1\}^N$ be another string. Assume that, after initialization of the system, all the messages m_i have been stored in the Cloud $\mathcal{CL}$ with virtual addresses $\phi(x_i)$, and that a user $\mathcal{Y}$ retrieved the set of virtual addresses $\Phi(m')$ associated to m'.

1. The scheme is said to be **complete**, up to a probability 1 - ϵ_1 if

$$\Pr_{m\prime}\left[\exists i,\ ed(m\prime, m_i) \leq \lambda_{min} \& \varphi(m_i) \notin \Phi(m\prime)\right] \leq \epsilon_1$$

(i.e. that except with a small probability all close messages are retrieved during the search through a Retrieve query).

2. The scheme is said to be **sound**, up to a probability 1 - ϵ_2 if

$$\Pr_{m\prime}\left[\exists i,\ d(m\prime, m_i) > \lambda_{max} \& \varphi(m_i) \in \Phi(m\prime)\right]$$

is bounded by ϵ_2 (i.e. that a false positive happens only with a small probability).

We now give the definition of the security properties that the scheme needs to fulfill to ensure that the data stored in the Cloud are kept confidential and that privacy of queries is ensured.

Condition 2. ***Sender Privacy*** The scheme is said to respect *Sender Privacy* if the advantage of any server is negligible in the $Exp_{\mathcal{A}}^{Sender\ Privacy}$ experiment, described below.

Here, $\mathcal{A}$ is a malicious opponent taking the place of $\mathcal{CL}$, and $\mathcal{C}$ is a challenger at the user side.

$Exp_{\mathcal{A}}^{\text{Sender Privacy}}$

1. (pk, sk)	$\leftarrow KeyGen(1^{\ell})$	$(\mathcal{C})$
2. $\{m_2, \ldots, m_\Omega\}$	$\leftarrow \mathcal{A}$	$(\mathcal{A})$
3. $\varphi(m_i)$	$\leftarrow Send_{\mathcal{C},\mathcal{CL}}(m_i, pk)$	$(\mathcal{C})$
4. $\{m_0, m_1\}$	$\leftarrow \mathcal{A}$	$(\mathcal{A})$
5. $\varphi(m_e)$	$\leftarrow \underset{e\in_R\{0,1\}}{Send_{\mathcal{C},\mathcal{CL}}(m_e, pk)}$	$(\mathcal{C})$
6. Repeat steps $(2, 3)$	$Send_{\mathcal{C},\mathcal{CL}}(m_e, pk)$	
7. $e\prime \in \{0, 1\}$	$\leftarrow \mathcal{A}$	$(\mathcal{A})$

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

This experiment corresponds to a first phase where the adversary receives Send requests that he chose himself. Then $\mathcal{A}$ selects a pair (m_0,m_1) of messages and the challenger $\mathcal{C}$ chooses randomly one of the two messages to be stored in the Cloud. At the end, after a polynomial number of other Send requests, the adversary tries to guess which one of m_0 or m_1 has been sent. When the advantage of the adversary is negligible, we can assume that the data stored in the Cloud remains private.

The next condition focuses on retrieve queries. We want to ensure that the Cloud does not learn information on the retrieve queries, i.e. neither on the input message *m'*, nor on the close retrieved messages.

Condition 3. ***Receiver Privacy*** The scheme is said to respect *Receiver Privacy* if the advantage of the Cloud is negligible in the experiment $Exp_{\mathcal{A}}^{\text{Receiver Privacy}}$ described below. $\mathcal{A}$ denotes the malicious opponent taking the place of $\mathcal{CL}$, and $\mathcal{C}$ the challenger at the user side.

$Exp_{\mathcal{A}}^{\text{Sender Privacy}}$

1. (pk, sk)	$\leftarrow KeyGen(1^{\ell})$	$(\mathcal{C})$
2. $\{m_1, \ldots, m_\Omega\}$	$\leftarrow \mathcal{A}$	$(\mathcal{A})$
3. $\varphi(m_i), (i \in \{1, \ldots, \Omega\})$	$\leftarrow Send_{\mathcal{C},\mathcal{CL}}(m_i, pk)$	$(\mathcal{C})$
4. $\{m'_2, \ldots, m'_p\}$	$\leftarrow \mathcal{A}$	$(\mathcal{A})$
5. $\Phi(m'_j), (j \in \{2, \ldots, p\})$	$\leftarrow Retrieve_{\mathcal{C},\mathcal{CL}}(m'_j, sk)$	$(\mathcal{C})$
6. (m'_0, m'_1)	$\leftarrow \mathcal{A}$	$(\mathcal{A})$
7. $\Phi(m\prime_e)$	$\leftarrow \underset{e\in_R\{0,1\}}{Retrieve_{\mathcal{C},\mathcal{CL}}(m'_e, sk)}$	$(\mathcal{C})$
8. *Repeat steps* $(4, 5)$		
9. $e\prime \in \{0, 1\}$	$\leftarrow \mathcal{A}$	$(\mathcal{A})$

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

This experiment begins with the adversary's choice of messages to be stored in the Cloud. Then $\mathcal{A}$ chooses a number of retrieve queries to be made by the challenger. Following this, $\mathcal{A}$ selects a pair of challenges (m'_0, m'_1) and one of them is randomly selected by $\mathcal{C}$ as input to a Retrieve query. Note that $\mathcal{A}$ should not see the result of the Retrieve queries. At the end of the experiment, $\mathcal{A}$ tries to guess which one it was.

This condition captures the privacy of the receiver $\mathcal{Y}$ when generating Retrieve queries: $\mathcal{CL}$ does not learn information on their content.

Useful technical tools

Private identification schemes

The principle of a private identification scheme is to manage nearest neighbor search in the encrypted domain. The two main sub-problems are the Approximate Nearest Neighbor (ANN) problem and Searchable Encryption

The Approximate Nearest Neighbor (ANN) problem is defined as follows: Let $\mathcal{P}$ be a set of points in a metric space (E,d_E). For an input $x \in E$ and $\epsilon \geq 0$, find a point $p_x \in \mathcal{P}$ such that

$$d_E(x, p_x) \leq (1 + \epsilon) \min_{p \in \mathcal{P}} d_E(x, p).$$

This is an approximation of the Nearest Neighbor problem as the exact case is hard to solve in large dimension spaces. Several algorithms for the ANN problem have been proposed [19] and the basic principle is to rely on sketching methods which output shorter vectors with increased stability and which enable to simplify the search: $\mathcal{P}$ is preprocessed with such sketching to end-up with a look-up table of short vectors on which the search can be realized quickly through counting the number of the exact or almost exact matches. Sketching needs there to guarantee that two close inputs would give with a good probability the same short vector. Examples of sketching methods are numerous for vector space (with Hamming distance or Euclidean distance) [20-23]; for instance random projections on small subspace. In the private identification schemes [7-9], the authors suggest to use a construction exploited in [24] for iris biometry. This is adapted to binary vectors with Hamming distance comparison. The sketching functions are restriction of n bits vectors over $r \ll n$ of their coordinates to obtain r bits vectors:

Definition 2. Let $\mathcal{F} = (f_1, \ldots, f_\mu)$ be a family of function from $\{0,1\}^n$ to $\{0,1\}^r$ such that for $x \in \{0,1\}^n$, we have for all $i \in \{1,...,\mu\}$, $f_i(x) = (x_{i_1}, \ldots, x_{i_r})$. We say that $\mathcal{F}$ is a sketching family for the Hamming distance from dimension n to dimension r.

With a sketching family where all functions are independent and if we assume that the inputs are uniformly distributed, the probability to obtain the same output with two distinct inputs can be estimated as follows.

$$\forall x, x' \in \{0,1\}^n \begin{cases} Pr_{f \in \mathcal{F}}[f(x) = f(x') \mid d(x, x') < \lambda_1] > \left(1 - \frac{\lambda_1}{n}\right)^r \\ Pr_{f \in \mathcal{F}}[f(x) = f(x') \mid d(x, x') > \lambda_2] < \left(1 - \frac{\lambda_2}{n}\right)^r \end{cases}$$

In our construction, we rely on this idea for Hamming distance approximation combined with the embedding method from [10,11] of edit distance into the Hamming space.

As far privacy and security are concerned, private identification schemes are based on searchable encryption principle. The main goal of searchable encryption [2,25] is to store messages into an encrypted database while still enabling to search the messages related to some keywords. For instance this could correspond to a remote mailing service where the user wants to retrieve his messages which contain a given keyword,

without letting the server learn information on the content of his mails. [3] also uses such technique but only in a symmetric context. Following [7]'s idea, we adapt an asymmetric searchable encryption scheme for our construction (cf. Section **Our construction**).

A general solution to design a searchable encryption scheme is to associate a message to a set of keywords and to consider each keyword as a virtual address where the receiver can recover a link toward the associated messages. To manage all these relations in an efficient way, we follow [1,26,27] by using Bloom filters. Bloom filter [28] is a notion used in membership checking applications to reduce the memory cost of the data storage. We use an extension of this notion called *Bloom filters with storage*. It enables to store identifiers of elements in each array.

Definition 3. *Bloom Filter with Storage*, [1] Let $\mathcal{S}$ be a finite subset of a space E and a set of identifiers associated to $\mathcal{S}$. For a family of v (independent and random) hash functions $\mathcal{H} = \{h_1, \ldots, h_v\}$, with each h_i:E→{1,...,k}, a (v,k)-*Bloom Filter with Storage* for indexation of $\mathcal{S}$ is $\mathcal{H}$, together with the array (t_1,...,t_k), defined recursively as:

1. $\forall i \in \{1,...,k\}$, $t_i \leftarrow \emptyset$,
2. $\forall x \in \mathcal{S}, \forall j \in \{1, \ldots, v\}$, $t_{h_j(x)} \leftarrow t_{h_j(x)} \cup \{Id(x)\}$ where *Id(x)* is the identifier of x.

In other words, the array is empty at the beginning and for each element $x \in \mathcal{S}$, we add the identifier *Id(x)* of x at the cells indexed by $h_1(x)$,...,$h_v(x)$. To recover the identifiers associated to an element y, we compute $T(y) = \bigcap_{j=1}^{v} t_{h_j(y)}$. The following lemma describes the accuracy of this storage method.

Lemma 1. [28] Let $(\mathcal{H}, t_1, \ldots, t_k)$ be a (v,k)-Bloom filter with storage indexing $\mathcal{S}$. For $x \in \mathcal{S}$, the following properties hold:

- $Id(x) \in T(x) = \bigcap_{j=1}^{v} t_{h_j(x)}$, i.e. the identifier of $x \in \mathcal{S}$ is always retrieved,
- the probability $\Pr[t \in T(y)$ and $t \neq Id(y)]$ to obtain a false positive is $\left(1 - \left(1 - \frac{v}{k}\right)^{|\mathcal{S}|}\right)^{v}$.

Edit distance approximation

Our construction is based on the embedding of edit distance into Hamming distance designed in [10]. To solve problems such as those described in Section **Private identification schemes**, data are embedded into Hamming space and then we can apply techniques dedicated to Hamming distance.

Definition 4. Let (E_1, d_{E_1}) and (E_2, d_{E_2}) be two metric spaces. An embedding $\psi : (E_1, d_{E_1}) \rightarrow (E_2, d_{E_2})$ has a distortion c if for all (x,y) ∈ E_1,

$$c^{-1} \times d_{E_1}(x, y) \leq d_{E_2}(\psi(x), \psi(y)) \leq c \times d_{E_1}(x, y)$$

[10] proves that $\{0,1\}^N$ with edit distance can be embedded into ℓ_1 with small distortion $2^{O(\sqrt{\log_2 N \log_2 \log_2 N})}$ and then shows from a previous work [20] how to end upefficiently into the Hamming space. More precisely:

Lemma 2. [10] There exists a probabilistic polynomial time algorithm π and constants c_1,$c_2 > 0$ that, for every N ∈ ℕ, for every $4^{-N} \gg \delta > 0$, and for all × ∈ $\{0,1\}^N$, computes $\pi(x) \in \ell_1^{c_2(N^2 \log_2(N/\delta))}$ and such that for all (x,y) ∈ $\{0,1\}^N$, with probability at least 1 - δ,

$$2^{-c_1(\sqrt{\log_2 N \log_2 \log_2 N})} ed(x,y) \leq L_1(\pi(x), \pi(y)) \leq 2^{c_1(\sqrt{\log_2 N \log_2 \log_2 N})} ed(x,y)$$

where L_1 denotes the distance L_1.

The principle of the algorithm is to partition a string x into about $2^{(\sqrt{\log_2 N \log_2 \log_2 N})}$ substrings. From each substring x^i, sets of all substrings (shingles) when taking a window of a fixed size t are considered (i.e. all possible substrings of x^i formed by t subsequent coordinates). By considering the metric defined by the minimum cost perfect matching algorithm between sets, [10] then explains how such sets are embedded into ℓ_1. Note that this technique introduces a lot of redundancy in the substrings which are embedded and this increases the dimension by a factor at least N^2, but this is interesting for our construction as the distortion is very low and the algorithm remains polynomial in N.

Based on [20], the authors then show that there exist $0 < \alpha < \beta < c_2$ and an embedding Ψ from $\{0,1\}^N$ with edit distance *ed* to $\{0,1\}^{c_2(\log_2(1/\delta))}$ with Hamming distance *HD* that computes $\Psi(x) = (x;t)$ for every $t \in \mathbb{N}$ and such that with probability at least 1 - \delta:

- If ed(x,y) ≤ t, then $HD(\psi(x), \psi(y)) \leq \alpha \log_2(1/\delta)$.
- If $ed(x,y) \geq 2^{c_1(\sqrt{\log_2 N \log_2 \log_2 N})} t$ then $HD(\psi(x), \psi(y)) \geq \beta \log_2(1/\delta)$.

Our construction

Technical description

Setup Let $\{0,1\}^N$ be equipped with the edit distance. Let Ψ be the embedding of $(\{0,1\}^N, ed)$ into $(\{0,1\}^{c_2(\log_2(1/\delta))}, HD)$ (cf. previous section). Let $\mathcal{F} = (f_1, \ldots, f_\mu)$ be a sketching family for the Hamming distance from dimension $c_2(\log_2(1/\delta))$ to a dimension r. Let $(\mathcal{H}, t_1, \ldots, t_k)$, with $\mathcal{H} = \{h_1, \ldots, h_v\}$, and h_i:{1,...,μ}×{0,1}^{r}*ow*{1,...,k}, be a (v, k)-*Bloom Filter with Storage*.

Let (Gen, Enc, Dec) be a semantically secure (IND-CPA, [29]) public key cryptosystem, let $Query_{DB}^{PIR}$ be the retrieve query from a database *DB* of a Private Information Retrieval protocol and let $Update_{DB}^{PIS}(val, i)$ be the write query into a database *DB* (that adds *val* to the *i*-th field) of a Private Information Storage protocol.

A Private Information Retrieval (PIR) [16] protocol enables to retrieve a specific block from a database without letting the database learn anything about the query and the answer (i.e. neither the index of the block nor the value of the block). This is done through a method $Query_{DB}^{PIR}(i)$, that allows a user to recover the element stored at index *i* in *DB* by running the PIR protocol. A Private Information Storage (PIS) protocol [17] enables to write information in a database while preventing the database from learning information on what is being stored (neither the value of the data, nor the index of the location where the data is being stored). Such a protocol provides a method $Update_{DB}^{PIS}(val, index)$, which takes as input an element and a database index, and puts the value *val* into the database entry *index*. See Section **Cryptographic primitives** for more details on these notions.

KeyGen(1^ℓ)

The function takes a security parameter ℓ as input and uses Gen to generate a public and private key pair (pk,sk). It also initializes the Bloom filter array, $(t_1,...,t_k \leftarrow (\emptyset,...,\emptyset))$, and provides it to the Cloud.

$$Send_{\mathcal{X},\mathcal{CL}}(m, pk)$$

To send a message to the Cloud, a user $\mathcal{X}$ executes the following algorithm.

1. $\mathcal{X}$ sends Enc(m,pk) to $\mathcal{CL}$ which will give him back a virtual address ϕ(m).

2. $\mathcal{X}$ computes the embedding Ψ(m) and for all i $\in$ {1,...,μ}, $f_i \circ \psi(m)$ and for all j $\in$ {1,...,v}, $\mathcal{X}$ asks to $\mathcal{CL}$ to update the Bloom filter array through queries

$$Update_{\mathcal{CL}}^{PIS}\left(Enc\left(\varphi(m), pk\right), h_j\left(i||f_i \circ \psi(m)\right)\right)$$

in order to add the identifier into the cell $t_{h_j(i||f_i \circ \psi(m))}$.

For privacy concerns, $\mathcal{X}$ will also complete the Bloom filter array with random data in order to get the same number l of elements for all cells $t_1,...,t_k$.

At the end of the algorithm, $\mathcal{CL}$ has stored the message m at a virtual address noted ϕ(m) and the Bloom filter structure has been filled of encrypted identifiers via indexation by several sketches that enable to search with approximate data.

$$Retrieve_{\mathcal{Y},\mathcal{CL}}(m\prime, sk)$$

To retrieve a message in the Cloud, a user $\mathcal{Y}$ proceeds as follows.

1. For all i $\in$ {1,...,μ} and for all j $\in$ {1,...,v}, $\mathcal{Y}$ computes $\alpha_{i,j} = h_j(i||f_i \circ \psi(m))$.

2. $\mathcal{Y}$ executes $Query_{\mathcal{CL}}^{PIR}(\alpha_{i,j})$ to retrieve the content of the cells $t_{\alpha_{i,j}}$ from the Bloom filters stored into $\mathcal{CL}$.

3. $\mathcal{Y}$ decrypts the content of the cells with Dec(.,sk) and for i $\in$ {1,...,μ}

- $\mathcal{Y}$ computes the intersection of all the decrypted version of the cells $t_{\alpha_{i,1}}, \ldots, t_{\alpha_{i,v}}$.
- If ϕ(m) is in this intersection, this means that $\mathcal{Y}$ most probably found a match $f_i \circ \psi(m) = f_i \circ \psi(m')$

4. $\mathcal{Y}$ counts the number of times an identifier is retrieved in such intersections $\bigcap_{j=1}^{v} t_{\alpha_{i,j}}$ (for i $\in$ {1,...μ}).

5. $\mathcal{Y}$ selects all the identifier which are retrieved above some threshold τ. This leads to the result $\Phi(m\prime) = \{\varphi(m_{i_1}), \ldots, \varphi(m_{i_\gamma})\}$ of the execution of Retrieve.

Note that as the queries are made through a PIR protocol, the Cloud can not learn any information. The advantage of using Bloom filters here is to permit an efficient look-up into the structure, as for classical Bloom filter (i.e. without any encryption) compared to other hash tables techniques.

Security properties

In this section, we explain why this construction achieves the security requirements of Section **Security requirements**.

Lemma 3. *Completeness* The scheme is complete up to a probability 1 - ϵ_1 with

$$\epsilon_1 \leq 1 - \left(1 - \frac{\alpha}{c_2}\right)^{r\tau}$$

Proof. (sketch of) For m,m' such that ed(m,m') ≤ λ_{min}, Section **Edit distance approximation** implies that $HD(\psi(m;\lambda_{min}), \psi(m';\lambda_{min}) \leq \alpha\log_2(1/\delta)$ with probability 1 - δ. Hence

$$Pr[f_i(\psi(m)) = f_i(\psi(m'))] > (1 - \frac{\alpha}{c_2})^r.$$

This leads to a probability lower than $1 - (1 - \frac{\alpha}{c_2})^{r\tau}$ to find less than τ times the identifier of a close message; probability that can thus be made small, cf. the example in Section **Discussion**.

More precisely, $\epsilon_1 \approx \sum_{i=0}^{\tau-1} \binom{\mu}{i} (1 - (1 - \frac{\alpha}{c_2})^r)^{\mu-i}(1 - \frac{\alpha}{c_2})^{ri}$.

Lemma 4. *Soundness* With $\lambda_{max} = 2^{c_1(\sqrt{\log_2 N \log_2 \log_2 N})}\lambda_{min}$ and provided that Bloom filter functions from $\mathcal{H}$ behave like pseudo-random functions from {1,...,μ} × {0,1}^{r} to {1,...,k}, then the scheme is sound up to a probability 1 - ϵ_2, with:

$$\epsilon_2 \approx \left((1 - \frac{\beta}{c_2})^r(1 - \frac{1}{k^v}) + \frac{1}{k^v}\right)^\tau$$

Proof. (sketch of) For m,m' such that ed(m,m') > λ_{max}, then Section **Edit distance approximation** implies that $HD(\psi(m;\lambda_{min}), \psi(m';\lambda_{min}) \geq \beta\log_2(1/\delta)$. Hence

$$Pr[f_i(\psi(m)) = f_i(\psi(m'))] < (1 - \frac{\beta}{c_2})^r.$$

The other cause for an error could come from v collisions in the Bloom filter hashes.

Lemma 5. ***Sender Privacy*** Assume that the PIS protocol achieves PIS User Privacy, the scheme ensures Sender Privacy.

Proof. (sketch of) $\mathcal{CL}$ receives only encrypted messages and Update^{PIS} queries that do not enable to distinguish between the output of Send(m_0, pk) and the output of Send (m_1, pk), after the execution of Send(m_1, pk), i ∈ {2,...,Ω} as we assume that the underlying encryption scheme is semantically secure and that the PIS protocol achieves PIS User Privacy.

Lemma 6. ***Receiver Privacy*** Assume that the PIR protocol ensures PIR User Privacy, then the scheme ensures Receiver Privacy.

Proof. (sketch of) The Cloud $\mathcal{CL}$ receives and answers only to Query^{PIR} requests, that by assumption do not leak information neither on their content nor on the outputs.

Discussion

To illustrate the error rates that one can expect, we give an example of choice of parameters. For instance, we choose a Bloom filter array of size k = 128 with v = 64 hash functions. Then we can approximate ϵ_2 as $(1 - \frac{\beta}{c_2})^{r\tau}$. We have $\epsilon_1 \approx \sum_{i=0}^{\tau-1} \binom{\mu}{i} (1 - (1 - \frac{\alpha}{c_2})^r)^{\mu-i}(1 - \frac{\alpha}{c_2})^{ri}$ where α < β. Assume that α = c_2/4 and β = c_2/2 then with μ = 128 functions in the sketching family for the Hamming distance, r = 10 and τ = 3, we obtain ϵ_2 negligible and $\epsilon_1 \approx 0.023$. With these parameters, we have μ × v = 2^{13} for the number of queries during Send and Retrieve phases.

Concerning the cost of PIR and PIS queries, the size of the Bloom filter array should remain not too large, like k = 128 here, to be efficient.

Note that in practice, the choice of λ_{min} depends on the number of errors between two words that one wants to tolerate for fuzzy search. Our embedding is made such that λ_{max} is made close to λ_{min}. The other parameters have then to be tuned to obtain small or negligible error rates ϵ_1 and ϵ_2 (cf. Lemma 3 and Lemma 4). The purpose of this paper is to introduce a new encrypted search with edit distance. At this point, our contribution is mainly theoretical. To go further, one should consider a practical use case over the cloud to be able to devise an efficient implementation.

Acknowledgements
The authors thank Céline Chevalier for her support.

Author details
[1]Morpho, Issy-les-Moulineaux, France [2]Morpho & Télécom ParisTech, Paris, France

Authors' contributions
JB and HC follow their previous work on biometric identification to extend it to the new area of application of cloud computing. Both authors read and approved the final manuscript.

Competing interests
The authors declare that they have no competing interests.

Received: 24 August 2011 Accepted: 23 February 2012 Published: 23 February 2012

References

1. Boneh D, Kushilevitz E, Ostrovsky R, Skeith WE III (2007) Public Key Encryption That Allows PIR Queries. In: Menezes A (ed) CRYPTO, Volume 4622 of Lecture Notes in Computer Science. Springer pp 50–67
2. Curtmola R, Garay JA, Kamara S, Ostrovsky R (2006) Searchable symmetric encryption: improved definitions and efficient constructions. CCS'06: Proceedings of the 13th ACM conference on Computer and communications security. ACM pp 79–88
3. Li J, Wang Q, Wang C, Cao N, Ren K, Lou W (2009) Enabling Efficient Fuzzy Keyword Search over Encrypted Data in Cloud Computing. Cryptology ePrint Archive, Report 2009/593 16
4. Daugman J (2003) The importance of being random: statistical principles of iris recognition. Pattern Recognit 36(2):279–291. doi:10.1016/S0031-3203(02)00030-4.
5. Bringer J, Despiegel V (2010) Binary feature vector fingerprint representation from minutiae vicinities. Biometrics: Theory, Applications, and Systems, 2010. BTAS'10. IEEE 4th International Conference on
6. Bringer J, Despiegel V, Favre M (2011) Adding localization information in a fingerprint binary feature vector representation. SPIE Defense, Security, Sensing
7. Bringer J, Chabanne H, Kindarji B (2009) Error-tolerant searchable encryption. IEEE ICC 2009 CISS
8. Bringer J, Chabanne H, Kindarji B (2011) Identification with encrypted biometric data. Security Comm Networks 4(5):548–562. doi:10.1002/sec.206.
9. Adjedj M, Bringer J, Chabanne H, Kindarji B (2009) Biometric Identification over Encrypted Data Made Feasible. In: Prakash A, Gupta I (ed) ICISS, Volume 5905 of Lecture Notes in Computer Science. Springer pp 86–100
10. Ostrovsky R, Rabani Y (2005) Low distortion embeddings for edit distance. In: Gabow HN, Fagin R (ed) STOC. ACM pp 218–224
11. Ostrovsky R, Rabani Y (2007) Low distortion embeddings for edit distance. J ACM 54(5)
12. Chor B, Kushilevitz E, Goldreich O, Sudan M (1998) Private Information Retrieval. J ACM 45(6):965–981. doi:10.1145/293347.293350.
13. Gertner Y, Ishai Y, Kushilevitz E, Malkin T (1998) Protecting data privacy in private information retrieval schemes. STOC 151–160
14. Gentry C, Ramzan Z (2005) Single-database private information retrieval with constant communication rate. In: Caires L, Italiano GF, Monteiro L, Palamidessi C, Yung M (ed) ICALP, Volume 3580 of Lecture Notes in Computer Science. Springer pp 803–815
15. Lipmaa H (2005) An oblivious transfer protocol with log-squared communication. In: Zhou J, Lopez J, Deng RH, Bao F (ed) ISC, Volume 3650 of Lecture Notes in Computer Science. Springer pp 314–328
16. Gasarch WI A Survey on Private Information Retrieval.http://www.cs.umd.edu/~gasarch/pir/pir.html
17. Ostrovsky R, Shoup V (1997) Private information storage (extended abstract). STOC 294–303
18. Ostrovsky R, Skeith WE III (2007) Algebraic Lower Bounds for Computing on Encrypted Data. Cryptology ePrint Archive, Report 2007/064
19. Piotr I (2004) Nearest neighbors in high-dimensional spaces. In: Goodman JE, O'Rourke J (ed) Handbook of Discrete and Computational Geometry, Chapter 39, 2nd edn. CRC Press
20. Kushilevitz E, Ostrovsky R, Rabani Y (1998) Efficient Search for approximate nearest neighbor in high dimensional spaces. Symposium on the Theory Of Computing 614–623
21. Kirsch A, Mitzenmacher M (2006) Distance-sensitive bloom filters. Algorithm Engineering & Experiments

22. Piotr I, Rajeev M (1998) Approximate nearest neighbors: towards removing the curse of dimensionality. Symposium on the Theory Of Computing 604–613
23. Andoni A, Piotr I (2008) Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. Commun ACM 51:117–122
24. Hao F, Daugman J, Zielinski P (2008) A Fast Search Algorithm for a Large Fuzzy Database. Inf Forensics Security, IEEE Trans 3(2):203–212
25. Boneh D, Di Crescenzo G, Ostrovsky R, Persiano G (2004) Public Key Encryption with Keyword Search. In: Cachin C, Camenisch J (ed) EUROCRYPT, Volume 3027 of LCNS. Springer pp 506–522
26. Goh EJ (2003) Secure indexes. Cryptology ePrint Archive, Report 2003/216
27. Bethencourt J, Song DX, Waters B (2006) New constructions and practical applications for private stream searching (extended abstract). IEEE Symposium on Security and Privacy. IEEE Computer Society pp 132–139
28. Bloom BH (1970) Space/time trade-offs in hash coding with allowable errors. Commun ACM 13(7):422–426. doi:10.1145/362686.362692.
29. Goldwasser S, Micali S (1984) Probabilistic Encryption. J Comput Syst Sci 28(2):270–299. doi:10.1016/0022-0000(84)90070-9.

doi:10.1186/PREACCEPT-1253053215890607
Cite this article as: Bringer and Chabanne: **Embedding edit distance to enable private keyword search.** *Human-centric Computing and Information Sciences* 2012 **2**:2.