# Fully homomorphic encryption: Searching over encrypted cloud data

**Ahmed EL-YAHYAOUI**

**Information Security Research Team, CEDOC ST2I ENSIAS, Mohammed V University in Rabat, Rabat, Morocco**

**ahmed_elyahyaoui@um5.ac.ma**

**Mohamed Dafir EC-CHRIF EL KETTANI**

**Information Security Research Team, CEDOC ST2I ENSIAS, Mohammed V University in Rabat, Rabat, Morocco**

**dafir.elkettani@um5.ac.ma**

## Abstract

Searching over encrypted data is a very important property that can offer some encryption schemes. It means performing queries on encrypted data in the absence of purpose to decryption, which permits to preserve confidentiality of sensitive data. Private Information Retrieval (PIR) is an essential protocol when selected information from remote databases is wanted to be done in secrecy. Fully homomorphic encryption is a revolutionary domain of cryptography that allows processing encrypted data without the need of any prior decryption, thus generating an encrypted result that corresponds the result of operations performed on the plaintext. In this paper we will exploit the important property that can offer the powerful fully homomorphic encryption to show how to realize a PIR protocol and how to search over encrypted data in a cloud context.

***Keywords***: search, encrypted data, PIR, protocol, cloud, database, homomorphic encryption.

## 1. Introduction:

In a context of cloud computing and outsourcing data storage. Many clients' uses cloud services to reduce operational costs of backups and infrastructure maintenance. A client stores confidential information, on outlying and anonymous cloud environment, in an encrypted form to avoid untrusted servers and third part to leak confidentiality of his sensitive data.

Law imposes sometimes encryption of some specific outsourced data. For example, legislation in many countries around the word states that electronic health records (EHRs) must be encrypted. Normal utilization of data becomes a challenge after encryption. Searching over encrypted data is an important property that can offer some encryption schemes. It is highly needed if the client want to query his encrypted database on the cloud server.

Searching over encrypted data (figure 1) was first introduced by Song, Wagner and Perrig [1]. The scheme from [1] is a searchable symmetric encryption (SSE) which is symmetric key cryptography based. It allows a single client to write and read data, i.e. only the secret key holder is able to create searchable ciphertexts and trapdoors. A second category of searchable encryption schemes will be risen after a pioneering work of Boneh et al in 2004 [2], who invented a public key encryption with keyword search (PEKS) scheme. It is a public key encryption based, for which the private key can decrypts all messages encrypted under the corresponding public key, i.e. It permits multi-clients to write (to encrypt) and only a single user to read (to decrypt).

Private Information Retrieval (PIR) [3] (Figure 2) is a protocol that allows a user to obtain an element of a database by hiding from it what element it is. While this problem admits a trivial solution – The client retrieves the entire database and query with a perfect privacy – this technique is not applicable for large databases because of the communication complexity.

Combining searching over encrypted data with private information retrieval enhances the cloud security and offers more protection to sensitive data. This solution gives more flexibility to clients to deal with their outsourced cloud data as it allows the cloud computing to do more operations on encrypted database.

Fully homomorphic encryption is a powerful tool that can allow searching over encrypted data and realizing PIR protocols. It enables cloud server to search blindly on client's encrypted databases without decrypting it or acquiring any knowledge about the plaintext data or the searched query. The first apparition of fully homomorphic encryption was in 1978 with Rivest,



BDCA'17, March 29-30, 2017, Tetouan, Morocco

ACM ISBN 978-1-4503-4852-2/17/03…$15.00
http://dx.doi.org/10.1145/3090354.3090364

Adleman and Dertozous under the name of privacy homomorphism [4], but privacy homomorphism stayed a conjecture without an effective solution until 2009. In his thesis, Craig Gentry presented the first semantic secure privacy homomorphism under the name of fully homomorphic encryption scheme [5]. The scheme from [5] uses a bootstrapping theorem to reduce the noise generated after processing ciphertexts. The cleartext space of Gentry's scheme [5] is the binary field, such a space allows us to evaluate any circuit on encrypted data using homomorphic capacities of the scheme. Several works followed Gentry's breakthrough and proposed new fully homomorphic encryption schemes [6], [7], [8]… some of this new constructions tried to use large cleartext spaces [9], [10] to reduce encryption cost.

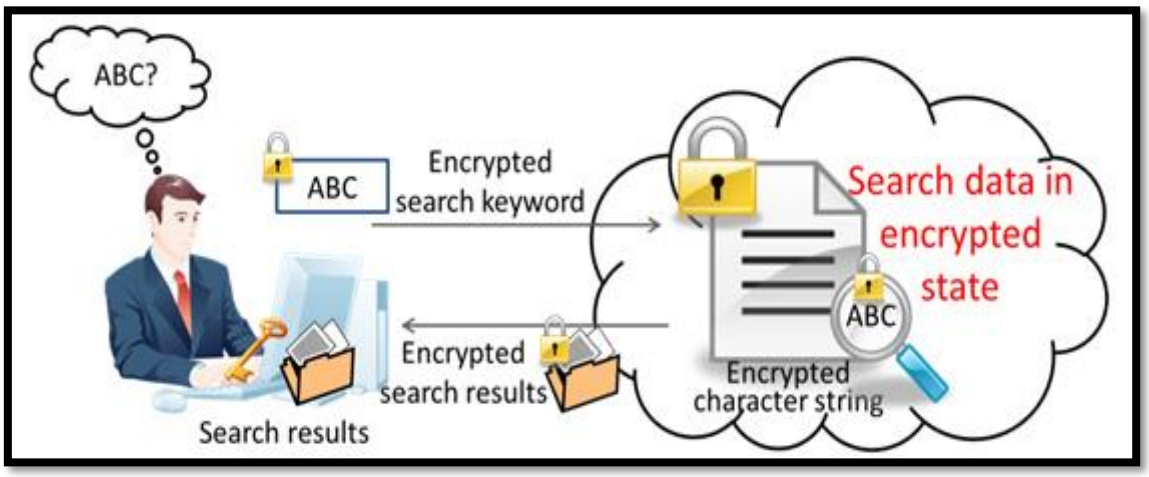


*Figure 1: Search over encrypted data*

In this paper, we propose an efficient search solution and a PIR protocol based on fully homomorphic encryption. For that reason we will focus on binary circuit based fully homomorphic encryption schemes. I.e. cryptosystems for which the cleartext space is {0,1} with XOR and AND operations.

## 2. Notations

In the following, the notation $\mathcal{E}(m)$ will denote the encryption of the message $m$ and the notation $\mathcal{D}(c)$ will denote the decryption of the ciphertext $c$. Intuitively, if we have $c = \mathcal{E}(m)$ then $m = \mathcal{D}(c)$.

## 3. Homomorphic encryption

In its large sense, a homomorphic encryption is a cryptosystem that allows a user, in addition to encryption and decryption operations, to perform computations on encrypted data. If the encryption scheme gives permission to a limited number of computations, it said partially homomorphic. For example, the unpadded RSA [11] cryptosystem is supposed to be multiplicatively homomorphic, i.e. for two cleartext $m_1$ and $m_2$ we have $\mathcal{E}(m_1) = {m_1}^e \ mod\ N$ and $\mathcal{E}(m_2) = {m_2}^e \ mod\ N$ such that $N$ is the RSA modulus and $e$ is the public key exponent, so $\mathcal{E}(m_1).\mathcal{E}(m_2) = {m_1}^e.{m_2}^e mod\ N = (m_1.m_2)^e mod\ N = \mathcal{E}(m_1.m_2)$. RSA allows us to perform only multiplications on ciphertext but no addition, an example of an additively homomorphic encryption scheme is Paillier's cryptosystem [12], if the public key is the modulus $N$ and the base $g$, then the encryption of a message $m$ is $\mathcal{E}(m) = g^m r^N mod\ N^2$, for some random $r \in \{1, \dots N-1\}$. The homomorphic property of Paillier's scheme is then $\mathcal{E}(m_1).\mathcal{E}(m_2) = (g^{m_1}{r_1}^N).(g^{m_2}{r_2}^N) mod\ N^2 = g^{m_1+m_2}(r_1 r_2)^N\ mod\ N^2 = \mathcal{E}(m_1 + m_2)$.

In the other hand a fully homomorphic encryption scheme is a cryptosystem that allows aclient, in addition to encryption and decryption operations, to perform any calculation on his encrypted data without decrypting it. Suppose that we have two cleartext $m_1$ and $m_2$, let $c_1 = \mathcal{E}(m_1)$ and $c_2 = \mathcal{E}(m_2)$ be its ciphertexts respectively. A fully homomorphic encryption permits us to obtain $c_1 + c_2 = \mathcal{E}(m_1 + m_2)$ and $c_1 \times c_2 = \mathcal{E}(m_1 \times m_2)$. For that reason, a user can delegate his complex computations to a remote cloud, which holds unlimited computation powers, to do it in his place.

Mathematically, a fully homomorphic encryption scheme is a quadruplet of polynomial algorithms $(Gen, Enc, Dec, Eval)$ verifying:

- $Gen(\lambda)$: Is an algorithm of key generation , takes as input a security parameter $\lambda$ and outputs a public and secret keys $(pk, sk)$.
- $Enc(m, pk)$: Is an encryption algorithm, takes as input a plaintext $m$ and a public key $pk$ and outputs a ciphertext $c$.
- $Dec(c, sk)$: Is a decryption algorithm, takes as input a ciphertext $c$ and a secret key $sk$ and outputs a plaintext $m$.
- $Eval(C, c_1, \dots, c_n)$: Is an evaluation algorithm, takes as input a circuit $C$ and ciphertexts $c_1, \dots, c_n$ and verifies $Dec(Eval(C, c_1, \dots, c_n), sk) = C(m_1, \dots, m_n)$. Anyone can evaluate $Eval$, since it does not require the secret key $sk$.

In the following, our scheme will be a circuit-based cryptosystem that allows a client to encrypt a cleartext bit by bit.

In addition, we will assume that we have a database DB, with $n$ encrypted entries $c_1, c_2 \dots c_n$, stored in a remote cloud-computing server.

## 4. Test of equality of two encrypted messages

With a fully homomorphic encryption scheme, one can test if two encrypted messages are equals or not without decrypting it. In this section, we will develop the operator $equal(c_1, c_2)$, which takes in input two ciphertexts $c_1 = \mathcal{E}(m_1)$ and $c_2 = \mathcal{E}(m_2)$ and permits to verify if $m_1$ and $m_2$ are equals without decrypting the given ciphertexts. To establish this operator we will need the following two operations to be done on ciphertexts:

### 4.1. Inversion

Inversion of an encrypted bit is an operation that permits to transform the encryption of a single bit to the encryption of its inverse based only on ciphertext. This operation can be realized from a fully homomorphic encryption scheme as:

Assume that $\sigma \in \{0,1\}$ and that $c = \mathcal{E}(\sigma)$, we will note $\bar{c} = \mathcal{E}(\bar{\sigma})$ such that $\bar{\sigma} = \sigma \oplus 1$ is the inverse of $\sigma$.

$\bar{c}$ can be obtained from $c$ by calculating $\bar{c} = c + \mathcal{E}(1) = \mathcal{E}(\sigma) + \mathcal{E}(1) = \mathcal{E}(\sigma \oplus 1) = \mathcal{E}(\bar{\sigma})$.

### 4.2. Ones' complement

The ones' complement of a binary number is defined as the value obtained by inverting all the bits in the binary representation of the number. It is a kind of inversion of multiple bits at the same time. To compute the ones' complement of a bit-by-bit encrypted input, we carry out the inversion of each encrypted bit.

Let $m$ be a binary cleartext (which is not necessarily a bit) encrypted to $C = \mathcal{E}(m)$ and $\bar{C} = \mathcal{E}(\bar{m})$ such that $\bar{m}$ is the ones' complement of $m$. We have $\bar{C} = \mathcal{E}(m) + \mathcal{E}(111 \dots 1) = \mathcal{E}(m \oplus 111 \dots 1) = \mathcal{E}(\bar{m})$.

### 4.3. Equality

As it precedes $equal(c_1, c_2)$ takes two ciphertexts as input. $equal$ will return an encrypted result which once decrypted we obtain a single bit. If the bit is 1 that means $m_1$ and $m_2$ are equals, else $m_1$ and $m_2$are not equals.

To test equality of $m_1$ and $m_2$, only by using $c_1$ and $c_2$ we will proceed as follows:

1. Compute the bitwise addition $c_1 \oplus c_2 = \mathcal{E}(m_1 \oplus m_2)$

If $m_1 = m_2$ we will obtain $C = c_1 \oplus c_2 = \mathcal{E}(000 \dots 0)$ otherwise, we will have among the encrypted bits of $c_1 \oplus c_2$ at least one equal to 1.

2. Compute the ones' complement of $c_1 \oplus c_2$, which is $\overline{c_1 \oplus c_2}$.
3. Multiply all the encrypted bits of the obtained result. As a consequence, we will obtain only an encrypted bit (0 or 1).
4. If the obtained result $equal(c_1, c_2) = \mathcal{E}(1)$ then we have $m_1 = m_2$, else $equal(c_1, c_2) = \mathcal{E}(0)$ then $m_1 \neq m_2$.

## 5. Search over encrypted data

Security is still a major inhibitor of cloud computing, when companies are testing cloud applications for storage and databases. Storing data in a cloud database in an encrypted form is very desirable today. The encryption allows us to protect the sensitive contents in the data, but this usually implies that one has to sacrifice functionality for security. Searching over encrypted data is the solution that should be of interest. It allows a cloud server to search blindly on client's encrypted data, based on a 'trapdoor' - a token that contains keywords to be searched for - sent by the client. Of course, client must also encrypt the keywords along the encrypted data before uploading them, which will surely increase the overhead. For 'blindly' it means the cloud server does not acquire any unnecessary knowledge about the searched keywords and the encrypted data, during the entire query process.

Fully homomorphic encryption allows us to query an encrypted database without the need for decryption key. In addition to doing computations on ciphertexts, it permits to search blindly over encrypted data.

Suppose that we have a database with $n$ entries $c_1, c_2 \dots c_n$, to look for the existence of an encrypted message $C = \mathcal{E}(m)$ among these encrypted entries we proceed as follows: After sending the query to the server, it proceeds as follows:

-It extracts the encrypted message $C$ from the query and tests its equality with each entry of the encrypted database.
Because of this phase, we will obtain $n$ encrypted bits $b_1 = equal(C, c_1)$, $b_2 = equal(C, c_2)$,… $b_n = equal(C, c_n)$.
-If C does not exist in the database, the $n$ encrypted bits will be all equals to an encrypted 0 i.e. $b_1 b_2 b_3 \dots b_n = \mathcal{E}(000 \dots 0)$. Else we will obtain at least one $b_i = \mathcal{E}(1)$ for a given $i \in [\![1, n]\!]$, i.e. $b_1 b_2 b_3 \dots b_i \dots b_n = \mathcal{E}(000 \dots 1 \dots 0)$.

So to verify the existence of $C$ in our database, the server should test the equality of the result $(b_1 b_2 b_3 \dots b_n)$ with $\mathcal{E}(000 \dots 0)$. i.e. it computes $SEARCH = equal(b_1 b_2 b_3 \dots b_n, \mathcal{E}(000 \dots 0))$.

- The obtained result will be inverted and sent to the requester. i.e. $\overline{SEARCH}$
After receiving the $\overline{SEARCH}$ encrypted bit, the requester will decrypt it. Two possible results are waited:
If he finds 0, so the searched query does not existed. Otherwise, hi will finds 1 which means that the searched query exists.

## 6. Private Information Retrieval

Assume that a client has queried a database and found a positive result ensuring that the searched data exists in the database. The protocol PIR (figure 2) allows him to retrieve his wanted information's in a safer way and without the cloud server being able to determine which element was selected. PIR protocols can be combined with searchable encryption schemes to obtain a more secure protocol which permits a clients to search on encrypted cloud data and retrieve it privately. This solution functions very well when outsourced data are sensitive. Fully homomorphic encryption is able to allow both: searching over encrypted data and private retrieval of wanted database entries.

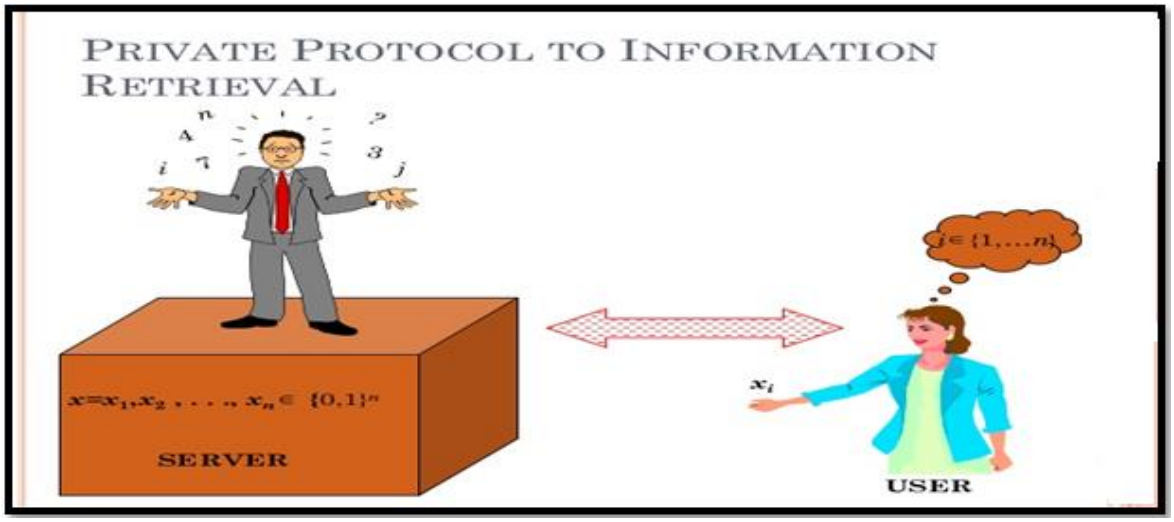


*Figure 2: Private Information Retrieval Protocol*

### 6.1. PIR from homomorphic encryption

PIR is realizable from homomorphic encryption. Assume that the database, with n entries $e_1, e_2, \dots, e_n$, is organized such that each entry is indexed by a unique integer. The entries in the database are not supposed to be encrypted, they are simply integers. We want to get the $k\ th$ element without the knowledge of the database owner. Suppose that we have a fully homomorphic encryption scheme.

The client starts by computing ciphertexts $c_i$, where $c_i$ is an encryption of 0 except for i=k, in this case $c_k$ is an encryption of 1. Because of our cryptosystem is a probabilistic scheme, 0 can be encrypted in several different ways. Therefore, all $c_i$ are different. The client sends to the cloud server the query $(c_1, c_2, \dots, c_n)$. The database administrator calculates $c_1e_1 + c_2e_2 + \cdots + c_ne_n$ and returns this result to the client. He then calculates $\mathcal{D}(c_1e_1 + c_2e_2 + \cdots + c_ne_n)$ and retrieves $e_k$. In fact, we have $c_1e_1 + c_2e_2 + \cdots + c_ne_n = \mathcal{E}(0)e_1 + \mathcal{E}(0)e_2 + \cdots + \mathcal{E}(1)e_k + \cdots + \mathcal{E}(0)e_n = \mathcal{E}(0 * e_1 + 0 * e_2 + \cdots + 1 * e_k + \cdots + 0 * e_n) = \mathcal{E}(e_k)$. (See figure 3).

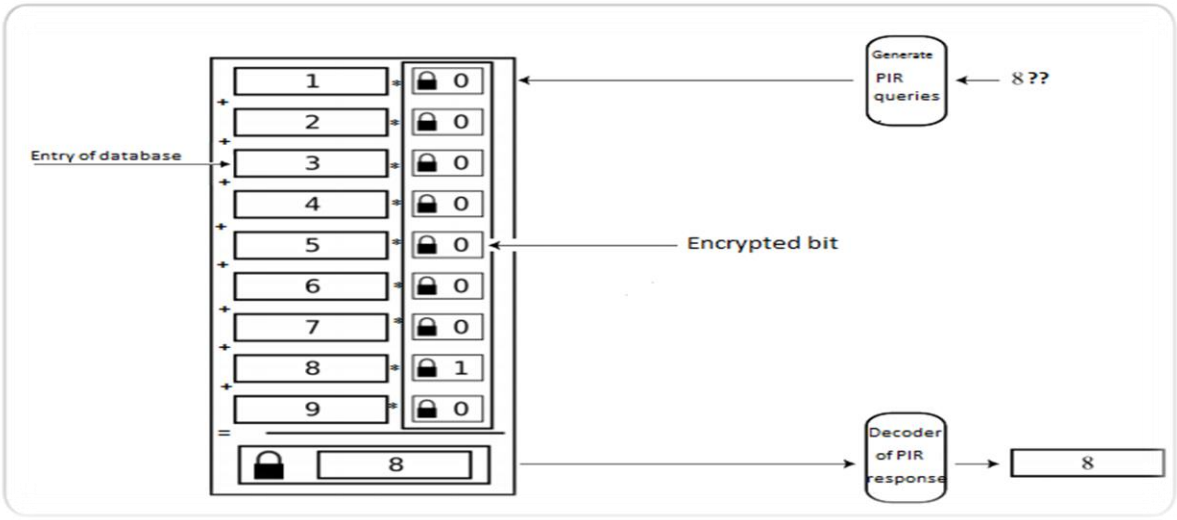


*Figure 3: PIR from homomorphic encryption*

## 7. Conclusion

In this paper, we treated the problem of searching over encrypted data and private information retrieval (PIR) protocol; all operations are done on a stored database in a remote cloud server. In the situation of an encrypted database, combining searching over encrypted data and PIR in one protocol is highly appreciated for a cloud context. Our approach to collaborate these two techniques is fully homomorphic encryption based. All operations are done, easily, under a cryptosystem that can allow computing on encrypted data without prior decryption.

# Bibliography


[1] D. Song , D. Wagner et A. Perrig, «Practical Techniques for Searches on Encrypted Data,» *IEEE Symposium on Security and Privacy*, pp. 44-55, 2000.

[2] D. Boneh, G. Crescenzo, R. Ostrovsky et G. Persiano, «Public key encryption with keyword search,» chez *EUROCRYPT (LNCS)*, 2004.

[3] B. Chor, O. Goldreich, . E. Kushilevitz et M. Suda, «Private information retrieval,» *In Proc. of the 36th Annu. IEEE Symp. on Foundations of Computer Science Pages 41–51, 1995. Journal version: J. of the ACM, 45:965–981, 1998.*

[4] R. Rivest, L. Adleman et M. Dertouzos, ««On Data Banks and Privacy Homorphisms,»,» *In Foundataions of Secure Computataion, Academic Press*, pp. 169-179, 1978.

[5] C. Gentry, «A fully homomorphic encryption scheme,» https://crypto.stanford.edu/craig/craig-thesis.pdf, September 2009.

[6] M. van Dijk, C. Gentry, S. Halevi et V. Vaikuntanan, «« Fully homomorphic encryption over the integers,»,» *Cryptology ePrint Archive, Report 2009/616, 2009. http://eprint.iacr.org/..*

[7] Z. Brakerski, C. Gentry et V. Vaikantanathan, ««« Fully Homomorphique Encryption without Bootstrapping,»,» *Available at http://eprint.iacr.org/2011/277..*

[8] J. Fan et F. Vercauteren, ««Somewhat Practical Fully Homomorphic Encryption,»,» *Available at http://eprint.iacr.org/2012/144. .*

[9] J. Kim, M. Sung Lee, A. Yun et J. Hee Cheon, «“ CRT-based Fully Homomorphic Encryption over the Integers”,» *Available at http://eprint.iacr.org/2013/057.*

[10] A. Kipnis and E. Hibshoosh, « « Efficient Methods for Practical Fully Homomorphic Symmetric-key Encrypton, Randomization and Verification »,,» *Cryptology ePrint Archive, Report 2012/637..*

[11] R. Rivest, A. Shamir et L. Adleman, «A Method for Obtaning Digital Signatures and Public Key Cryptosystems,» *Commununications of ACM, Vol. 21, pp. 120-126, April 1978..*

[12] P. Paillier, «Public-key cryptosystems based on composite degree residuosity classes.,» chez *18th Annual Eurocrypt Conference (EUROCRYPT'99)*, Prague, Czech Republic, 1999.

[13] D. Boneh, E. Goh et K. Nissim, «Evaluating 2-DNF formulas on ciphertexts. In Kilian, J., ed.: Theory of Cryptography, Second Theory of Cryptography Conference, TCC 2005, Cambridge, MA, USA, February 10-12, 2005, Proceedings. Volume 3378 of Lecture,» chez *Theory of Cryptography, Second Theory of Cryptography Conference*, Combridge, February 10-12, 2005.