$(k - n)$ Oblivious Transfer Using Fully Homomorphic Encryption System

Mohammed Kaosar[1], Quazi Mamun[1], Rafiqul Islam[1], and Xun Yi[2]

[1] School of Computing and Mathematics, Charles Sturt University, Australia
[2] School of Engineering and Science, Victoria University, Australia
{mkaosar,qmamun,mislam}@csu.edu.au, xun.yi@vu.edu.au

Abstract. Oblivious Transfer(OT) protocol allows a client retrieving one or multiple records from a server without letting the server know about the choice of the client. OT has been one of the emerging research areas for last several years. There exist many practical applications of OT, especially in digital media subscription. In this paper, we propose a fully homomorphic encryption based secure k out of n oblivious transfer protocol. This novel protocol, first ever to use fully homomorphic encryption mechanism for integers numbers, allows the client choosing its desired records by sending encrypted indexes to the server, server works on encrypted indexes and sends back encrypted result without knowing which records the client was interested in. From the encrypted response of the server, the client only can decrypt its desired records. The security analysis demonstrates that, the desired security and privacy requirement of OT is ensured by the proposed protocol. Some optimizations are also introduced in the proposed solution to reduce transmission overhead.

Keywords: Oblivious Transfer, Homomorphic Encryption, Private Information Retrieval, Data Outsourcing.

1 Introduction

In the current world, the use of information technology has increased tremendously. Consequently, secure storage, transmission and retrieval of information become one of the top concerns in the IT era. The diversity of devices, applications and infrastructures have increased this concern by another fold. The privacy of information in any transaction is no more a small issue. Private Information Retrieval (PIR) and Oblivious Transfer (OT) are some of the cryptographic protocols that ensures the privacy of the user in retrieving information from a storage or a server. Unlike PIR, OT ensures the server security too by not allowing the user retrieving unauthorised record(s). OT has been used in many applications including certifying email and coin flipping [1], simultaneous contract signing [2], digital right management [3], e-subscription to sell digital goods [4], privacy preserving data mining in distributed environment [5] etc.

To understand the basic principle of OT protocol, let us consider an example: let us say, a server stores n number of digital contents or records of information

T. Zia et al. (Eds.): SecureComm 2013, LNICST 127, pp. 380–392, 2013.

$x_1, x_2, ...,$ *and* x_n. Clients or users need to subscribe with the server to access an item. In such e-subscription, there will be two requirements to be fulfilled from the server's and the client's point of view respectively: (i) the client should not be able to retrieve any item(s) which it did not subscribe for and (ii) the server is not allowed to know which item(s) the client retrieved. That is, if the client wants to retrieve or access item x_i, OT protocol ensures that server cannot learn the value of i and the client cannot learn any x_j for all $j \neq i$.

In this paper our proposed solution uses a secure cryptographic protocol, particularly the fully homomorphic encryption over integer numbers proposed by Dijk and Gentry [6] in 2010, to ensure data privacy of the client. The server's security is ensured by encrypting all of its records using a symmetric key encryption system such as, AES [7] or DES [8]. k out-of n OT can be achieved by repeating 1 out-of n OT protocol k times. This approach incurs extremely high overhead. In this paper, we have proposed some optimizations in the $k-n$ OT protocol. It transmits the encrypted database only once at the beginning of the protocol. The server uses separate keys to encrypt each record using a symmetric key encryption technique. The protocol only allows the desired keys to be decrypted by the client. On the other hand, the client encrypts its choices using the homomorphic encryption technique and transmits to the server. The server encrypts and manipulates keys and indexes using the same technique without being able to decrypt any of the choices of the client. The fully homomorphic encryption of Dijk and Gentry is as strong as the approximate Greatest Common Divisor (GCD) problem (more detail of approximate GCD can be found in [9]). The security analysis shows that the proposed protocol ensures both the server's and the client's requirements.

The rest of the paper is organized as follows: Section 2 describes some background knowledge on the topic of the paper including the fully homomorphic encryption system which is used in the proposed protocol, Section 3 and 4 discuss our proposed model and the protocol, Section 5 discusses the security and performance analysis and finally, Section 6 concludes the paper with some hints towards the future research directions.

2 Background and Related Work

This section discusses about various OT protocols and existing solutions and the definition of homomorphic and fully homomorphic encryption system. This section also discusses how the fully homomorphic encryption for binary digits is extended to work for integer numbers.

2.1 Types of Oblivious Transfer Protocol

OT can be of three basic types:

- 1-*out-of*-2 *(*$1-2$ *OT):*
  1 out-of 2 oblivious transfer protocol allows the client retrieving one item

out of 2 from the server. The server does not know which item was accessed by the client and the client does not know about any item it did not chose to retrieve. Rabin [10] first proposed $1-2$ OT protocol in 1981. In this RSA [11] based protocol, the server sends the message (an item) to the client with the probability of $1/2$ and hence, the server remain oblivious whether the message was received or not. Later on, $1-2$ OT was developed by Evan et al. [12] while applying it in randomized protocol for signing contracts, certifying mail and flipping coin.

- 1-*out-of-n (1-n OT):*
  1 out-of- n oblivious transfer protocol allows the client retrieving one item out of n from the server. Often times 1-n OT is used as a generalization of 1-2 OT. 1-n OT is also similar to Private Information Retrieval (PIR), first proposed by Kushilevitz and Ostrovsky [13] in 1997, with an additional condition. In PIR the client can retrieve 1 item from n items without letting the server know its choice. The client may retrieve or access other items. Whereas, 1-n OT ensures the client won't be able to access anything other than what it retrieved. Some more about $1-n$ OT protocol can be found in [14,15].
- *k-out-of-n (k-n OT):*
  k out-of- n oblivious transfer protocol allows the client retrieving k number of items out of n from the server. The client would send k number of indexes to the server. The server would return all those desired items without knowing client's choices. $k-n$ OT was first proposed by Ishai et al. in [16]. Additive homomorphic encryption based $k-n$ OT protocol is proposed in [17]. $k-n$ OT can also be achieved by repeated use of $1-n$ OT. However, this approach would be very inefficient due to huge amount of overhead transmitted from server to the client.

2.2 Fully Homomorphic Encryption System (FHES)

Homomorphic encryption is a special form of encryption where one can perform a specific algebraic operation on the plain-text by applying the same or different operation on the cipher-text. If X and Y are two numbers and E and D denote encryption and decryption function respectively, then homomorphic encryption holds following condition for an algebraic operation, such as $'+'$:

$$D[E(X) + E(Y)] = D[E(X + Y)] \tag{1}$$

Most homomorphic encryption system such as RSA [11], ElGamal [18], Benaloh [19], Paillier [20] etc. are capable to perform only one operation. But fully homomorphic encryption system can be used for many operations (such as, addition, multiplication, division etc.) at the same time. In the area of cryptography, fully homomorphic encryption system proposed by Dijk et al. in [6] is considered as a breakthrough work which can be used to solve many cryptographic problems [21]. We have used this fully homomorphic encryption technique with necessary improvements and variations in data mining [22,23] and in private information retrieval [24].

Fully Homomorphic Encryption for Binary Bits

Fully homomorphic encryption of [6] works both over binary and integer numbers. This scheme has the ability to perform both addition and multiplication over the cipher-text and these operations are represented in plain-text. Hence, a untrusted party is able to operate on private or confidential data, without the ability to know what data the untrusted party is manipulating.

The fully homomorphic scheme [6] is a simplification of an earlier work involving ideal lattices [25]. It encrypts a single bit (in the plain-text space) to an integer (in the cipher-text space). When these integers are added and multiplied, the hidden bits are added and multiplied (modulo 2). A simple encryption and decryption process of symmetric version of the scheme is as follows:

Encryption : Lets say, p is the private key, q and r are chosen random numbers, and m is a binary message *i.e.* $m \in \{0,1\}$. Then the encryption of m would be $c = pq + 2r + m$.

Decryption : The message m is recovered simply by performing following operation: $m = (c \ mod \ p) mod \ 2$.

Thus, this encryption scheme works in the bit level and underlying bits are calculated accordingly if we add or multiply on cipher-text.

Using the symmetric version of the cryptosystem, it is possible to construct an asymmetric version. The asymmetric version is more useful especially when multiple parties are involved in the computation such as in data mining, data gathering, data outsourcing, OT, PIR etc. The asymmetric version of [6] would be as follows:

$KeyGen(\lambda)$: Choose a random n-bit odd integer p as the private key. Using the private key, generate the public key as $x_i = pq_i + 2r_i$ where q_i and r_i are chosen randomly, for $i = 0, 1, ..., \tau$. Rearrange $x - i$ such that, x_0 is the largest.

$Encrypt(pk, m \in \{0,1\})$**:** Choose a random subset $S \subseteq \{1, 2, ..., \tau\}$ and a random integer r. m is encrypted to the cipher-text $c = (m + 2r + 2\sum_{i \in S} x_i)(mod \ x_0)$. Let us denote this operation as $E_{pk}(m)$.

$Decrypt(sk, c)$**:** The message m is recovered simply by performing $m = (c \ mod \ p) mod \ 2$. Let us denote this operation as $D_{sk}(c)$.

The asymmetric version works same way as the symmetric one with same correctness and security strength. We have discussed about this in [24] and [22]. The addition and the multiplication on cipher-texts are reflected as addition and multiplication being acted on the message bit respectively. This produces the correspondence between the cipher-text space and the plain-text space, as addition in the cipher-text space reduces to exclusive OR ($\oplus$) in the plain-text space and multiplication in the cipher-text space reduces to AND ($\wedge$). This correspondence (homomorphism) between these two operations, addition and multiplication, are shown in Equations 2 and 3, respectively.

$$E(m_1) + E(m_2) = E(m_1 \oplus m_2) \tag{2}$$

$$E(m_1) \cdot E(m_2) = E(m_1 \wedge m_2) \tag{3}$$

Hence, from this correspondence, it is possible to construct very complicated binary circuits to evaluate on the data, without exposing the actual data. More details regarding the implementation can be found in the original paper [6].

Fully Homomorphic Encryption for Integers

Oblivious transfer deals with the privacy and security of some numeric values being exchanged between the client and the server. Hence, we need to extend the underlying cryptosystem to accommodate integer numbers, so that integer numbers can be taken into consideration. This is achieved by representing the integer as a binary vector and encrypting each bit separately and maintaining their positions or orders. For instance, an 8-bit integer X can be encrypted and presented as cipher-text as shown in Equation 4, assuming the binary representation of X is $X_8 + X_7 + X_6 + X_5 + X_4 + X_3 + X_2 + X_1$.

$$\begin{aligned} E_{pk}(X) = & \\ & E_{pk}(X_8) + E_{pk}(X_7) + E_{pk}(X_6) + E_{pk}(X_5) + E_{pk}(X_4) + \\ & E_{pk}(X_3) + E_{pk}(X_2) + E_{pk}(X_1) \\ & where + \ represents \ concatenation \ operation. \end{aligned} \tag{4}$$

This representation of integer number allows encrypting and decrypting each bit using the fully homomorphic encryption and decryption for binary digits as discussed in section 2.2. Not only that, some binary operations such as XOR, OR, AND etc. can be performed on two encrypted integer numbers homomorphically. Let us consider two integers X and Y of ℓ-bit long each. That is, $X = \{X_\ell + ... + X_2 + X_1\}$ and $Y = \{Y_\ell + ... + Y_2 + Y_1\}$. Let us say we want to perform binary XOR operation on X and Y, i.e. $R = X \ XOR \ Y$, where $R = \{R_\ell + ... + R_2 + R_1\}$ and $R_i = X_i \ XOR \ Y_i$. Therefore, according to fully homomorphic encryption $R_i = D_{sk}(R_i\prime)$, where $R_i\prime = E_{pk}(X_i) \ XOR \ E_{pk}(Y_i)$. For all $i = 1 \ to \ \ell$.

3 Model Definition

Let us consider a client C wants to access k number of records ($k \in \{1, 2, ..., n\}$) out of n records $\{R = R_1, R_2, ..., R_n\}$ stored in a database server S. Index of the interested records $\{I = I_1, I_2, ..., I_k\}$ are known to C only. C does not want S to discover which record(s) it is interested in. On the other hand, S wants to ensure only the desired record is received by C. Figure 1 illustrates the block diagram of the proposed model in brief.

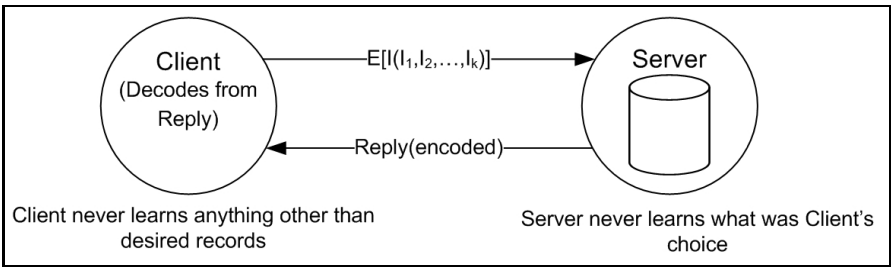

Fig. 1. Block diagram of the oblivious transfer protocol between the client and the server

The client C and the server S participates in the proposed OT protocol using the fully homomorphic encryption discussed in section 2.2. They also generate keys, encrypt, decrypt and transmit according to the protocol description discussed in following the section.

4 Proposed Solution

This section discusses our proposed OT protocol in the sequence of parameter setup, communication steps and algorithms.

4.1 Parameters and Initial Setup

Let us assume both the client C and the server S are capable to perform following operations to setup and carry on the proposed protocol:

Key generation

Client: Client C generates its private key and public key sk and pk respectively using the key generation technique discussed in section 2.2.

Server: Server S generates secret key sets $\chi = \{\kappa_1, \kappa_2, ..., \kappa_n\}$ using cryptographically secure pseudo-random number generator (CSPRNG). Standards of CSPRNG can be found in [26]. Each key is used to encrypt each record. That is, key κ_i is used to encrypt record R_i.

Encryption

Two kind of cryptosystems will be used in the proposed protocol:

FHES

Fully homomorphic encryption system based encryption and decryption functions for integer numbers works as follows:

Encryption: $E_{pk}(i)$ encrypts an ℓ-bit integer i using the public key pk, returning an encrypted ℓ-block long cipher-text c.

Decryption: $D_{sk}(c)$ decrypts an ℓ-block long cipher-text c using the private key sk, returning a plain-text ℓ-bit integer i.

Symmetric Key Cryptosystem

A secured symmetric key cryptosystem (e.g. AES [7] or DES [8]) based encryption and decryption notations are as follows:

Encryption: $R_i\prime = E\prime_{\kappa_i}(R_i)$ represents the encryption of record R_i using the key κ_i.

Decryption: $R_i = D\prime_{\kappa_i}(R_i\prime)$ represents the decryption of encrypted record $R_i\prime$ using the key κ_i.

Homomorphic XOR for Integers

Denoted as $(X \boxplus Y)$, receives two ℓ-block long cipher-text X and Y, and returns a third encrypted ℓ-block long cipher-text Z. The output is calculated bit-by-bit using the exclusive OR property of the homomorphic encryption discussed in section 2.2, that is $Z_i = X_i \ XOR \ Y_i$, where XOR is evaluated using Equation 2.

Shuffle by random permutation

Denoted as $\zeta(B\prime)$, randomly rearranges all ℓ number of blocks of the cipher-text $B\prime$, where $B\prime = E_{pk}(B)$. That is, if $B\prime = \{E_{pk}(B_\ell) + \!\!+ ... + \!\!+ E_{pk}(B_2) + \!\!+ E_{pk}(B_1)\}$, $\zeta(B\prime)$ will return $\{E_{pk}(B_i) + \!\!+ ... + \!\!+ E_{pk}(B_j) + \!\!+ E_{pk}(B_k)\}$ where values of i, j, k are non-repeating random numbers within the range of 1 *to* ℓ.

4.2 The Algorithm

Algorithm 1. Oblivious Transfer between C and S

input of $C : pk,\ sk,\ k,\ n,\ I$
input of $S : pk,\ R,\ n$
output to $C : R_{I_1}, R_{I_2}, ..., R_{I_k}$
Begin
Server S
Generate set of random keys $\chi = \{\kappa_1, \kappa_2, ..., \kappa_n\}$
Client C
for $All(i \in I)$ **do**
  $Q \leftarrow E_{pk}(I_i)$
  $SendToS(Q)$
  Server S
  $\Gamma \leftarrow \phi$ /* Initializes response string*/
  for $j = 1$ *to* n **do**
    $\alpha_j \leftarrow \zeta(Q \boxplus E_{pk}(j)) \boxplus E_{pk}(\kappa_j)$ /* ζ rearranges the order of the bits randomly*/
    $\beta_j \leftarrow E\prime_{k_j}(R_j)$
    $\Gamma_j \leftarrow \{\alpha_j \bigcup \beta_j\}$
    $\Gamma \leftarrow \Gamma \bigcup \Gamma_j$
  end for
  $SendToC(\Gamma)$
  Client C
    $\gamma \leftarrow \Gamma_{I_i}$ /* Extracts desired block from Γ. Components of γ are α and β*/
    $\alpha_i\prime \leftarrow \alpha$ /* Extracts encrypted keys*/
    $\beta_i\prime \leftarrow \beta$ /* Extracts encrypted record*/
    $\kappa_{I_i} \leftarrow D_{sk}(\alpha_i\prime)$ /* Decrypts the key to decrypt the desired record*/
    $R_{I_i} \leftarrow D\prime_{\kappa_{I_i}}(\beta_i\prime)$ /* This is the desired record of index I_i*/
end for
End

4.3 Flow Diagram

The algorithmic flow diagram for one request is shown in Figure 2.

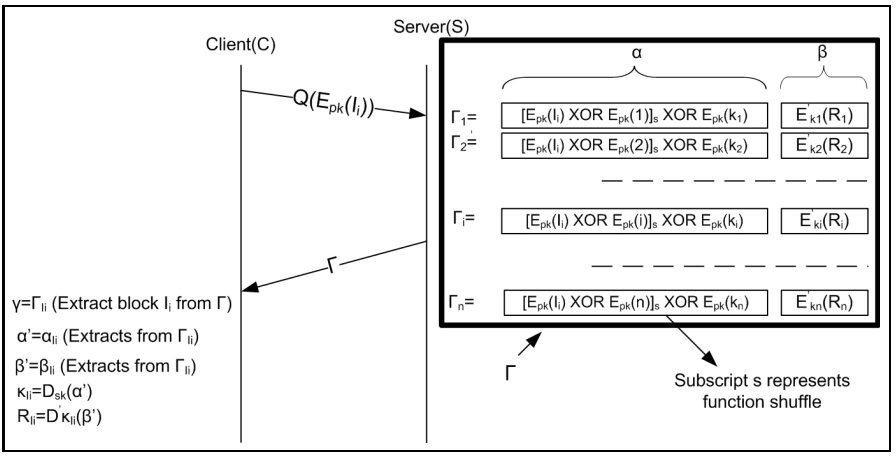

Fig. 2. Flow diagram of oblivious transfer protocol between C and S for one request

4.4 Further Optimization

In Algorithm 1, $k-n$ OT is implemented by repeated calling of $1-n$ OT k times. The client C sends k encrypted requests separately to the server S. The server returns the encrypted records each time the client requests. In the case of big size of the records, this method will be very inefficient. Alternatively, the server can transmit the whole chunk of encrypted records once at the first time and later on can transmit only the encrypted keys (the key which is used to encrypt a particular record), every time the client sends a request. In summary, the part of β in Figure 2 can be transmitted once at the beginning of the protocol and α can be transmitted k times to the client. This would reduce the transmission overhead drastically. Moreover, the client also can send all the k number of requests at once. The optimized solution is described in Algorithm 2.

Algorithm 2. Efficient Oblivious Transfer between C and S

input of $C : pk,\ sk,\ k,\ n,\ I$
input of $S : pk,\ R,\ n$
output to $C : R_I = \{R_{I_1}, R_{I_2}, ..., R_{I_k}\}$
Begin
Server S
Generate set of random keys $\chi = \{\kappa_1, \kappa_2, ..., \kappa_n\}$
$\Omega \leftarrow \phi$ /* Initializes the encrypted records*/
for $i = 1$ *to* n **do**
  $\Omega \leftarrow \Omega \bigcup E\prime_{k_i}(R_i)$
end for
Client C
$Q \leftarrow \phi$
for $All(i \in I)$ **do**
  $Q_{I_i} \leftarrow E_{pk}(I_i)$
  $Q \leftarrow Q \bigcup Q_{I_i}$
end for
$SendToS(Q, k)$ /* C sends all requests together to S*/
Server S
$\beta \leftarrow \phi$
for $j = 1$ *to* n **do**
  $\beta_j \leftarrow E\prime_{k_j}(R_j)$
  $\beta \leftarrow \beta \bigcup \beta_j$
end for
$SendToC(\beta)$ /* Sends all the encrypted records together*/
for $i = 1$ *to* k **do**
  $\alpha_i \leftarrow \phi$
  for $j = 1$ *to* n **do**
    $\alpha_j \leftarrow \zeta(Q_{I_i} \boxplus E_{pk}(j)) \boxplus E_{pk}(\kappa_j)$ /* ζ rearranges the order of the bits randomly*/
    $\alpha_i \leftarrow \alpha_i \bigcup \alpha_j$
  end for
  $SendToC(\alpha_i)$
end for
Client C
for $i = 1$ *to* k **do**
  $\alpha_{I_i}\prime \leftarrow \alpha_i$ /* Extracts blocks fromα that contain the key κ_{I_i}*/
  $\kappa_{I_i} \leftarrow D_{sk}(\alpha_{I_i})$
  $\gamma \leftarrow \beta_{I_i}$ /* Extracts desired block from β that contain R_{I_i}*/.
  $R_{I_i} \leftarrow D\prime_{\kappa_{I_i}}(\beta_i\prime)$ /* This is the desired record of index I_i*/
  $R_I \leftarrow R_I \bigcup R_{I_i}$
end for
return R_I

5 Analysis

The fully homomorphic encryption used in this protocol is as strong as approximate GCD problem, though its efficiency may not be very high. Some recent

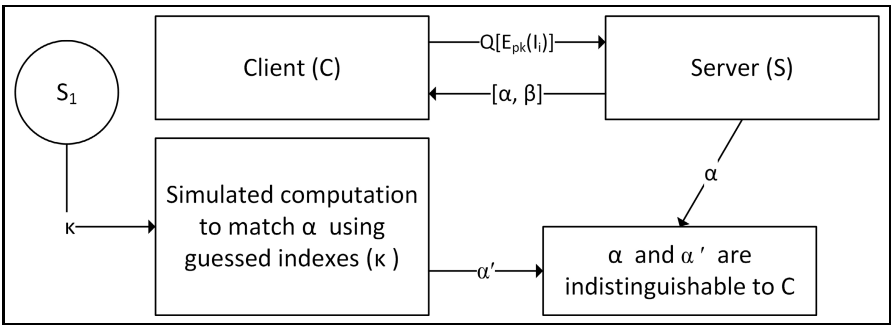

Fig. 3. C is adversary. S_1 guesses κ for C to compute a value to be equal to α.

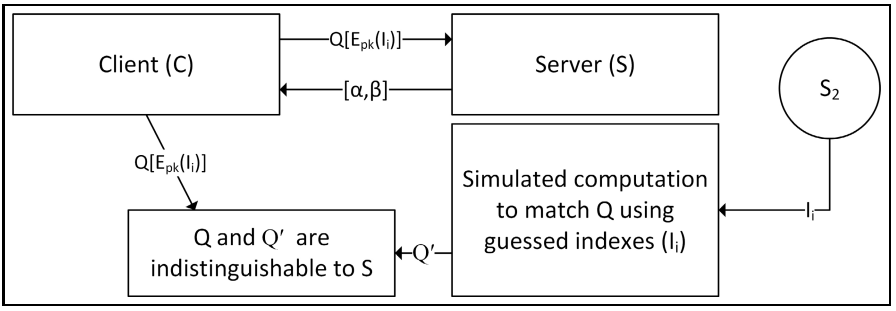

Fig. 4. S is adversary. S_2 guesses I_i for S to compute a value to be equal to $Q[E_{pk}(I_i)$.

works and ongoing research on improving this protocol, such as [27,28,21], indicates its performance to be enhanced in near future.

In this protocol, C generates and stores its secret and public keys sk and pk respectively. C does not send any data to S without being encrypted by its public key pk. Therefore, C's data is secured by the security of the fully homomorphic encryption scheme of [6]. On the other hand, S encrypts its data using same public key pk and performs operations on its own ciphe-rtext and C's cipher-text. The cipher-text is again shuffled using the function $\zeta()$ which is XORed with secret key. S then discloses this cipher-text to C. Therefore, the privacy of S's data depends on whether C can learn anything from the result sent by S and vice-versa. Let us consider C and S being adversary in two different cases:

Case1: the client C is adversary

Let us say C wants to recover a key κ_j where $j \notin I$. That is C wants to recover a key of a record which it did not retrieve. For each item I_i the client request, it receives n blocks of cipher-text, which is $\alpha_j = \zeta(Q_{I_i} \boxplus E_{pk}(j)) \boxplus E_{pk}(\kappa_j)$ for all $j = 1, 2..., n$. Any block is only meaningful, in other word C can retrieve a key from, if $I_i = j$. This condition would make $Q_{I_i} \boxplus E_{pk}(j) = E_{pk}(I_i) \boxplus E_{pk}(j) = 0$ and hence, $\alpha_j = E_{pk}(\kappa_j)$ from which C can decrypt κ_j. For any other block where $I_i \neq j$, $Q_{I_i} \boxplus E_{pk}(j)$ would be non-zero and further

shuffle by $\zeta()$ would make this part indistinguishable and unrecoverable to C. Simulated and guessed key values in C is computationally indistinguishable. Figure 3 shows C's view and simulated outcome ($\alpha \stackrel{c}{\equiv} \alpha\prime$), where,$\stackrel{c}{\equiv}$ denotes computational indistinguishability..

Case2: The server S is adversary

C encrypts all the indexes of its choices using its public key pk. Therefore, no one can know about the indexes without the secret key sk which is only possessed by C, given the fully homomorphic encryption is secure. Server S cannot know the value of C's choice by encrypting all indexes from 1 to n and and comparing with C's encrypted choices. Because asymmetric version of fully homomorphic encryption guarantees that two cipher-texts of the same bits are always different. Moreover simulated or guessed index of C's choices are indistinguishable to S. Figure 4 illustrates S's view and simulated outcome ($Q \stackrel{c}{\equiv} Q\prime$).

6 Conclusion and Future Work

In this paper, we have proposed a novel OT protocol using a fully homomorphic encryption system. The security of this protocol is as strong as approximate GCD problem. Security analysis also ensures that, the client C cannot discover any record it did not retrieve and the server S cannot learn the choice(s) of the client. The enhancement of the fully homomorphic encryption system used in this solution will influence the performance of the proposed protocol in great deal. Implementation and performance comparison with existing solution are left for the future research.

References

1. Blum, M.: Three application of oblivious transfer: Part i: Coin flipping by telephone; part ii: How to exchange secrets; part iii: How to send certified electronic mail (2001)
2. Líšková, L., Stanek, M.: Efficient Simultaneous Contract Signing. In: Deswarte, Y., Cuppens, F., Jajodia, S., Wang, L. (eds.) Security and Protection in Information Processing Systems. IFIP, vol. 147, pp. 440–455. Springer, Boston (2004)
3. Min Sun, H., Hang Wang, K., Fu Hung, C.: Towards privacy preserving digital rights management using oblivious transfer (2006)
4. Aiello, B., Ishai, Y., Reingold, O.: Priced oblivious transfer: How to sell digital goods. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 119–135. Springer, Heidelberg (2001)
5. Wang, W., Deng, B., Li, Z.: Application of oblivious transfer protocol in distributed data mining with privacy-preserving. In: Proceedings of the First International Symposium on Data, Privacy, and E-Commerce, ISDPE 2007, pp. 283–285. IEEE Computer Society, Washington, DC (2007)
6. van Dijk, M., Gentry, C., Halevi, S., Vaikuntanathan, V.: Fully homomorphic encryption over the integers. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 24–43. Springer, Heidelberg (2010)

7. FIPS-PUB.197: Advanced encryption standard. Federal Information Processing Standards Publications, US Department of Commerce/N.I.S.T., National Technical Information Service (2001)
8. FIPS-Pub.46: Data encryption standard. National Bureau of Standards, US Department of Commerce (1977)
9. Zeng, Z., Dayton, B.H.: The approximate gcd of inexact polynomials. In: Proceedings of the 2004 International Symposium on Symbolic and Algebraic Computation, ISSAC 2004, pp. 320–327. ACM, New York (2004)
10. Rabin, M.: How to Exchange Secrets by Oblivious Transfer. Technical Report TR-81, Harvard Aiken Computation Laboratory (1981)
11. Rivest, R.L., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems. Commun. ACM 21, 120–126 (1978)
12. Even, S., Goldreich, O., Lempel, A.: A randomized protocol for signing contracts. Commun. ACM 28, 637–647 (1985)
13. Kushilevitz, E., Ostrovsky, R.: Replication is not needed: Single database, computationally-private information retrieval (extended abstract). In: Proc. of the 38th Annu. IEEE Symp. on Foundations of Computer Science, pp. 364–373 (1997)
14. Naor, M., Pinkas, B.: Oblivious transfer with adaptive queries. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 573–590. Springer, Heidelberg (1999)
15. Laur, S., Lipmaa, H.: A new protocol for conditional disclosure of secrets and its applications. In: Katz, J., Yung, M. (eds.) ACNS 2007. LNCS, vol. 4521, pp. 207–225. Springer, Heidelberg (2007)
16. Ishai, Y., Kushilevitz, E.: Private simultaneous messages protocols with applications. In: Proc. of 5th ISTCS, pp. 174–183 (1997)
17. Murugesan, M., Jiang, W., Nergiz, A.E., Uzunbaz, S.: k-out-of-n oblivious transfer based on homomorphic encryption and solvability of linear equations. In: CODASPY 2011, pp. 169–178 (2011)
18. El Gamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 10–18. Springer, Heidelberg (1985)
19. Clarkson, J.B.: Dense probabilistic encryption. In: Proceedings of the Workshop on Selected Areas of Cryptography, pp. 120–128 (1994)
20. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999)
21. Naehrig, M., Lauter, K., Vaikuntanathan, V.: Can homomorphic encryption be practical? In: Proceedings of the 3rd ACM Workshop on Cloud Computing Security Workshop, pp. 113–124. ACM, New York (2011)
22. Kaosar, M., Paulet, R., Yi, X.: Fully homomorphic encryption based two-party association rule mining. Data and Knowledge Engineering 76-78, 1–15 (2012)
23. Kaosar, M., Paulet, R., Yi, X.: Secure two-party association rule mining. In: Australasian Information Security Conference, AISC 2011 (2011)
24. Yi, X., Kaosar, M., Paulet, R., Bertino, E.: Single-database private information retrieval from fully homomorphic encryption. IEEE Transactions on Knowledge and Data Engineering 25, 1125–1134 (2013)
25. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: STOC 2009: Proceedings of the 41st Annual ACM Symposium on Theory of Computing, pp. 169–178. ACM, New York (2009)

26. NIST: Recommendation for random number generation using deterministic random bit generators. U.S. Department of Commerce, National Institute of Standards and Technology (NIST) Special Publication 800-90A (January 2012)
27. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (leveled) fully homomorphic encryption without bootstrapping. In: Proceedings of the 3rd Innovations in Theoretical Computer Science Conference, ITCS 2012, pp. 309–325. ACM, New York (2012)
28. Coron, J.-S., Mandal, A., Naccache, D., Tibouchi, M.: Fully Homomorphic Encryption over the Integers with Shorter Public Keys. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 487–504. Springer, Heidelberg (2011)