Secure and Verifiable Outsourcing of Sequence Comparisons

Yansheng Feng, Hua Ma, Xiaofeng Chen, Hui Zhu

Secure and Verifiable Outsourcing of Sequence Comparisons*

Yansheng Feng[1], Hua Ma[1], Xiaofeng Chen[2], Hui Zhu[3]

[1] Department of Mathematics, Xidian University,
Xi'an 710071, P.R. China
fengyansheng1108@163.com , ma_hua@126.com
[2] State Key Laboratory of Integrated Service Networks,
Xidian University, Xi'an 710071, P.R. China
xfchen@xidian.edu.cn
[3] Network and Data Security Key Laboratory of Sichuan Province,
Chengdu 611731, P.R. China
zhuhui@xidian.edu.cn

Abstract. With the advent of cloud computing, secure outsourcing techniques of sequence comparisons are becoming increasingly valuable, especially for clients with limited resources. One of the most critical functionalities in data outsourcing is verifiability. However, there is very few secure outsourcing scheme for sequence comparisons that the clients can verify whether the servers honestly execute a protocol or not. In this paper, we tackle the problem by integrating the technique of garbled circuit with homomorphic encryption. As compared to existing schemes, our proposed solution enables clients to efficiently detect the dishonesty of servers. In particular, our construction re-garbles the circuit only for malformed responses and hence is very efficient. Besides, we also present the formal analysis for our proposed construction.

Key words: Outsourcing, Garbled Circuit, Verifiable Computation.

1 Introduction

Several trends are contributing to a growing desire to outsource computing from a device with constrained resources to a powerful computation server. This requirement would become more urgent in the coming era of cloud computing. Especially, cloud services make this desirable for clients who are unwilling or unable to do the works. Although attractive these new services are, concerns about security have prevented clients from storing their private data on the cloud. Aiming to address this problem, secure outsourcing techniques are developed.

Among the existing secure outsourcing techniques, a specific type receives great attention, namely sequence comparisons. Atallah et al. first propose this concept in [1], which achieves the nice property of allowing resource-constrained

* Corresponding author: Xiaofeng Chen (xfchen@xidian.edu.cn)

devices to enjoy the resources of powerful remote servers without revealing their private inputs and outputs. Subsequent works [2, 3] are devoted to obtain efficiency improvements of such protocols. Techniques for securely computing the edit distance have been studied in [5, 6], which partition the overall computation into multiple sub-circuits to achieve the same goal. Besides, [7, 8] introduce the Smith-Waterman sequence comparisons algorithm for enhancing data privacy. Blanton et al. [4] utilize finite automata and develop techniques for secure outsourcing of oblivious evaluation of finite automata without leaking any information. In particular, considering highly sensitive individual DNA information, it is indispensable to privately process these data, for example, [17] encrypts sensitive data before outsourcing. Furthermore, when the length of sequences is large, it is not surprising to alleviate clients from laborious tasks by outsourcing related computation to servers.

Recently, Blanton et al. [11] propose a non-interactive protocol for sequence comparisons, namely, a client obtains the edit path by transmitting the computation to two servers. However, the scheme [11] is impractical to some extent in that it does not achieve verifiability. Also, the scheme [11] has to garble each circuit when the sub-circuit is processed and hence is not efficient. As we know, there seems few available techniques for secure outsourcing of sequence comparisons, which can provide verifiability and enjoy desirable efficiency simultaneously. Among, [15] also achieves the verifiability that can be done by providing fake input labels and checking whether the output from the servers matches a precomputed value, but it verifies with some probability by using the technologies of commitment and Merkle hash tree that differ from our scheme.

Our Contribution. In this paper, we propose a construction for secure and verifiable outsourcing of sequence comparisons, which enables clients to efficiently detect the dishonesty of servers by integrating the technique of garbled circuit with homomorphic encryption. Specially, our solution is efficient in that it re-garbles the circuit only for malformed responses returned by servers. Formal analysis shows that the proposed construction is proved to achieve all the desired security notions.

2 Preliminaries

2.1 Edit Distance

We briefly review the basic algorithm for the edit distance [1]. Let $M(i, j)$, for $0 \leq i \leq m$, $0 \leq j \leq n$, be the minimum cost of transforming the prefix of λ of length j into the prefix of μ of length i. So, $M(0, 0) = 0$, $M(0, j) = \sum_{k=1}^{j} D(\lambda_k)$ for $1 \leq j \leq n$ and $M(i, 0) = \sum_{k=1}^{i} I(\mu_k)$ for $1 \leq i \leq m$. Furthermore, we can recurse to obtain the results:

$$M(i,j) = min \begin{cases} M(i-1, j-1) + S(\lambda_j, \mu_i) \\ M(i-1, j) + I(\mu_i) \\ M(i, j-1) + D(\lambda_j) \end{cases}$$

where $S(\lambda_j, \mu_i)$ shows the cost of replacing λ_j with μ_i, $D(\lambda_j)$ shows the cost of deleting λ_j, and $I(\mu_i)$ shows the cost of inserting μ_i.

2.2 Grid Graph Measures

The relevances among the entries of the M-matrix induce an $(m+1) \times (n+1)$ grid directed acyclic graph (DAG). It is apparent to see that the string editing problem can be regarded as a shortest-path problem on DAG. An $l_1 \times l_2$ DAG is a directed acyclic graph whose vertices are the $l_1 l_2$ points of an $l_1 \times l_2$ grid, and such that the only edges from point (i, j) are to points $(i, j+1)$, $(i+1, j)$, and $(i+1, j+1)$. Figure 1 shows an example of DAG and explains that the point (i, j) is at the ith row from the top and the jth column from the left. As special cases of the above definition, the meanings of $M(0, 0)$ and $M(m, n)$ are easy to be obtained.

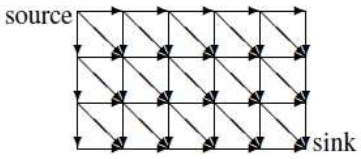

Fig. 1. Example of a 3×5 grid DAG

Such edit scripts that transform λ into μ are in one-to-one correspondence to the edit paths of G that start at the source (which represents $M(0, 0)$) and end at the sink (which represents $M(m, n)$).

2.3 Yao's Garbled Circuit

We summarize Yao's protocol for two-party computation [9], which is initiated by introducing the millionaire problem that is the same as [16]. For more details, we refer to Lindell and Pinkas' description [12].

We assume two parties, A and B, wish to compute a function F over their private inputs a and b. First, A converts F into a circuit C. A garbles the circuit and obtains $G(C)$, and sends it to B, along with garbled input $G(a)$. A and B then engage in a series of OTs so that B obtains $G(b)$ with A learning nothing about b. B then applies the garbled circuit with two garbled inputs to obtain a garbled version of the output: $G(F(a, b))$. A then maps this into the actual output.

In more detail, A constructs the garbled circuit as follows. For each wire w in the circuit, A chooses two random values $k_w^0, k_w^1 \xleftarrow{R} \{0,1\}^\lambda$ to represent 0/1 on that wire. Once A has determined every wire value, then forms a garbled version of each gate g (See Fig.2). Let g be a gate with input wires w_a and

w_a	w_b	w_z
k_a^0	k_b^0	$E_{k_a^0}(E_{k_b^0}(k_z^{g(0,0)}))$
k_a^0	k_b^1	$E_{k_a^0}(E_{k_b^1}(k_z^{g(0,1)}))$
k_a^1	k_b^0	$E_{k_a^1}(E_{k_b^0}(k_z^{g(1,0)}))$
k_a^1	k_b^1	$E_{k_a^1}(E_{k_b^1}(k_z^{g(1,1)}))$

Fig. 2. Circuit's garbled table

w_b, and output wire w_z. Then the garbled version $G(g)$ consists of simply four ciphertexts:

$$\gamma_{ij} = E_{k_a^i}(E_{k_b^j}(k_z^{g(i,j)})), where\ i \in \{0,1\}, j \in \{0,1\} \tag{1}$$

When k_a^α, k_b^β, and the four values γ_{ij} are given, it is possible to compute $k_z^{g(\alpha,\beta)}$. By then B transmits $k_z^{g(\alpha,\beta)}$ to A, A can map them back to 0/1 values and hence obtains the output of the function.

3 Secure Model and Definitions

In our work we transfer the problem of the edit distance computation by a client C for strings $\mu_1, ..., \mu_m$ and $\lambda_1, ..., \lambda_n$ to two computational servers S_1 and S_2. Furthermore, the security requirement is such that neither S_1 nor S_2 learns anything about the client's inputs or outputs except the lengths of the input strings and the alphabet size. More formally, we assume that S_1 and S_2 they are semi−honest and non−colluding, they follow the computation but might attempt to learn extra information. Here, we assume that S_2's ability is more powerful than S_1, we only prove that the attempt of S_2's attack fails in latter proof. Since the powerful adversary can not attack client successfully, neither can the weak one. Security in this case is guaranteed if both S_1's and S_2's views can be simulated by a simulator with no access to either inputs or outputs other than the basic parameters, and such simulation is indistinguishable from the real protocol. We introduce several definitions in the following:

Definition 1. *We say that a private encryption scheme (E, D) is Yao-secure if the following properties are satisfied*:

- Indistinguishable encryptions for multiple messages
- Elusive range
- Efficiently verifiable range

Definition 2. (Correctness). *A verifiable computation scheme VC_{edit} is correct if for any choice of function F, the key generation algorithm produces keys $(PK, SK) \leftarrow Keygen(F, \lambda)$ such that, $\forall x \in Domain(F)$, if $ProbGen_{SK}(x) \rightarrow \sigma_x$, $Compute_{PK}(\sigma_x) \rightarrow \sigma_y$, then $y = F(x) \leftarrow Verify_{SK}(\sigma_y)$.*

We then describe its correctness by an experiment:

Experiment $\text{Exp}_A^{veri}[VC_{edit}, F, \lambda]$
$(PK, SK) \leftarrow Keygen(F, \lambda)$;
For $i = 1, ..., l = poly(\lambda)$; $poly(\cdot)$ is a polynomial.
$x_i \leftarrow A(PK, x_1, \sigma_1, ..., x_i, \sigma_i)$;
$(\sigma_i) \leftarrow ProbGen_{SK}(x_i)$;
$(i, \hat{\sigma}_y) \leftarrow A(PK, x_1, \sigma_1, ..., x_l, \sigma_l)$;
$\hat{y} \leftarrow Verify_{SK}(\hat{\sigma}_y)$
If $\hat{y} \neq \perp$ and $\hat{y} \neq F(x_i)$, output '1', else '0';

The adversary succeeds if it produces an output that convinces the verification algorithm to accept on the wrong output for a given input.

Definition 3. (Security). *For a verifiable computation scheme VC_{edit}, we define the advantage of an adversary A in the experiment above as*:

$$Adv_A^{Verif}(VC_{edit}, F, \lambda) = Prob[Exp_A^{verif}[VC_{edit}, F, \lambda] = 1] \tag{2}$$

A verifiable computation scheme VC_{edit} is secure, if

$$Adv_A^{Verif}(VC_{edit}, F, \lambda) \leq negli(\lambda) \tag{3}$$

where negli(·) is a negligible function of its input. The F in the above descriptions is the function to calculate the $\theta(\cdot)$ in our protocol.

Definition 4. (One-time secure). *It is the same as Definition* 3 *except that in experiment Exp_A^{Verif}, the adversary can query the oracle $ProbGen_{SK}(\cdot)$ only once and must cheat on that input. VC_{Yao} is a special case of VC*[1] *when the input is single.*

Similar to formulas (2)(3), we can obtain:

$$Adv_A^{Verif}(VC_{Yao}, F, \lambda) = Prob[Exp_A^{verif}[VC_{Yao}, F, \lambda] = 1] \tag{4}$$

$$Adv_A^{Verif}(VC_{Yao}, F, \lambda) \leq negli(\lambda) \tag{5}$$

4 Secure and Verifiable Outsourcing of Sequence Comparisons

4.1 High-level Description

To gain the edit path, we can use a recursive solution: In the first round, instead of computing all elements of M, we compute the elements in the "top half" and the "bottom half" of the matrix respectively. Then calculate each $M(m/2, j)$ and determine the position with the minimum sum from top to bottom. In [11]

[1] This verifiable computation VC can be consulted in [14].

this position is expressed as $M(m/2, \theta(m/2))$. Then, we discard cells from the top right and lower left of $M(m/2, \theta(m/2))$. We recursively apply this algorithm to the remaining of the matrix. But this case exposes $M(m/2, \theta(m/2))$ to the servers. With protecting $\theta(m/2)$, we form two sub-problems of size 1/2 and 1/4 of the original [11].

We now introduce how this computation can be securely outsourced. First, client produces garbled circuit's random labels corresponding to its inputs (two labels per input bit). Then it sends all the labels to S_1 for forming the circuit and one label per wire that corresponds to its input value to S_2. Once the circuit is formed, S_2 will evaluate it using the labels. The novelty of this way is, scheme without OT protocols is also feasible.

An advanced non-interactive protocol has been proposed in [11]. Based on this paper, we achieve the multi-round inputs verifications by integrating the garbled circuit with fully homomorphic encryption [10]. Specifically, the client will encrypt labels under a fully homomorphic encryption public key. A new public key is generated for each-round input in order to prevent being reused. The server can then evaluate labels and send them to the client, who decrypts them and obtains $F(x)$. This scheme can reuse the garbled circuit until the client receives a malformed response, which is more efficient than generating the new circuit every time.

4.2 The Proposed Construction

The following is a safe and verifiable protocol to achieve the calculation of the edit path. Among, C stands for the client, S_1, S_2 as two servers. m, n are the lengths of two private strings. This allows us to obtain the overall protocol as follows:
Input: C has two private sequences as well as their cost tables, C must generate $min(m, n)$ key pairs of the fully-homomorphic encryption against all the sub-circuits. Besides, using generated keys, C encrypts the private input label's values and delivers these into the circuit.
Output: C obtains the edit path. S_1 and S_2 learn nothing.
Protocol VC_{edit}:

1. **Pre-computing**: C generates two random labels (l_0^t, l_1^t) for each bit of its input $\mu_1, ..., \mu_m, \lambda_1, ..., \lambda_n, I(\mu_i)$ for each $i \in [1, m], D(\lambda_j)$ and $S(\lambda_j, \cdot)$ for each $j \in [1, n], I(\cdot), D(\cdot)$, and $S(\cdot, \cdot)$, $t \in [1, S_\Sigma(m+n) + S_C(m+2n+n\sigma)]$. C also generates $min(m, n)$ key pairs of the fully-homomorphic encryption and runs the encryptions, $(\sigma_x \leftarrow Encrpyt(PK_{\mathbb{E}}^i, l_{bt}^t),\ i \in [1, min(m, n)])$ against all the sub-circuits.
2. **Circuit's construction**: C sends all (l_0^t, l_1^t) to S_1, S_1 uses the pairs of labels it received from C as the input labels in constructing a garbled circuit that produces $\theta(m/2)$, strings $\mu_1^{'}, ..., \mu_{m/2}^{'}, \lambda_1^{'}, ..., \lambda_n^{'}$ and the corresponding $I(\mu_i^{'}), D(\lambda_j^{'})$, and $S(\lambda_j^{'}, \cdot)$, as well as strings $\mu_1^{''}, ..., \mu_{m/2}^{''}, \lambda_1^{''}, ..., \lambda_{n/2}^{''}$ and the relevant $I(\mu_i^{''}), D(\lambda_j^{''})$, and $S(\lambda_j^{''}, \cdot)$. Let the pairs of the output labels that

correspond to $\theta(m/2)$ be denoted by $(\hat{l_0^t}, \hat{l_1^t})$, where $t \in [1, \lceil \log(n) \rceil]$, the labels corresponding to the output labels for the first sub-problem be denoted by $(l_0^{'t}, l_1^{'t})$, where $t \in [1, S_\Sigma(m/2+n) + S_C(m/2+n+n\sigma)]$ and the labels corresponding to the output labels for the second sub-problem be denoted by $(l_0^{''t}, l_1^{''t})$, where $t \in [1, S_\Sigma(m/2+n/2) + S_C(m+n+n\sigma)/2]$. Then, S_1 stores three types of labels.

3. **Keygen**: S_1 transfers $(\hat{l_0^t}, \hat{l_1^t})$ to the client as private key SK. In the pre-computing above, C has already generated key pairs of the fully-homomorphic encryption, $((PK_\mathbb{E}, SK_\mathbb{E}) \leftarrow Keygen(\lambda))$. C stores SK and $SK_\mathbb{E}$ and exposes $PK_\mathbb{E}$ to S_2.
4. **Evaluation**: C transmits all the labels σ_x in the pre-computing to S_2 for storing and evaluation. Then, S_2 uses $PK_\mathbb{E}$ to calculate $Encrypt(PK_\mathbb{E}, \gamma_i)$. Next, runs $Evaluate(C, Encrypt(PK_\mathbb{E}, \gamma_i), Encrypt(PK_\mathbb{E}, l_{bt}^t))$ with homomorphic encryption's property, we can obtain the $\sigma_y \leftarrow Encrypt(PK_\mathbb{E}, \hat{l_{bt}^t})$, which is stored in S_2 later.
5. **Sub-circuits evaluation**: S_1 and S_2 now engage in the second round of the computation, where for the first circuit S_1 uses pairs $(l_0^{'t}, l_1^{'t})$ as the input wire labels as well as the pairs of the input wire labels from C that correspond to cost tables $I(\cdot)$, $D(\cdot)$, and $S(\cdot,\cdot)$. After the circuit is formed, S_1 sends to S_2 who uses the encrypted labels stored before to evaluate this circuit. S_2 saves every result value of the evaluation as $\sigma_y^{(i)}$.
6. **Verification**: When S_1 and S_2 reach the bottom of recursion, S_2 sends the all $\sigma_y^{(i)}$ from each circuit to C. C uses $SK_\mathbb{E}$ stored before to decrypt $\sigma_y^{(i)}$ to get $\hat{l_{bt}^t}$. Further, converts the output labels into the output of the function (e.g., $F(x) = \theta(a), a = 1, ..., m$) by using SK, from which it can reconstruct the edit path.

5 Analysis of the Proposed Construction

5.1 Robust Verifiability

In practice, the view of the client will change after the evaluation. How to deal with this situation that client receives a malformed response? One option is to ask the server to run the computation again. But this repeated request informs the server that its response was malformed, server might generate forgeries. The client aborts after detecting a malformed response, but it can hinder our protocol's execution. We will consider it as follow:

There is indeed an attack if the client does not abort. Specifically, the adversary can learn the input labels one bit at a time by XOR operation [13]. So, it can generate cheating. When a malformed response come to the client, this paper needs to continue running the protocols VC_{edit} instead of terminating. we must ask the client to regarble the circuit, every time when a malformed response is received.

5.2 Security Analysis

Theorem 1. *Let E be a Yao-secure symmetric encryption scheme and $\mathbb{E}$ be a semantically secure homomorphic encryption scheme. Then protocol VC_{edit} is a secure and verifiable computation scheme.*

Proof: Since E is a Yao-secure symmetric encryption scheme, then VC_{Yao} is a one-time secure verifiable computation scheme (Proof of Theorem 3 in [13]). Our method is transformming (via a simulation) a successful adversary against the verifiable computation scheme VC_{edit} into an attacker for the one-time secure protocol VC_{Yao}. Next, for the sake of contradiction, let us assume that there is an adversary A such that $Adv_A^{Verif}(VC_{edit}, F, \lambda) \geq \varepsilon$, where ε is non-negligible in λ. We use A to build another adversary $A^{'}$ which queries the ProbGen oracle only once, and for which $Adv_{A'}^{Verif}(VC_{Yao}, F, \lambda) \geq \varepsilon^{'}$, where $\varepsilon^{'}$ is close to ε. Once we prove the Lemma 1 below, we have our contradiction and the proof of Theorem 1 is complete.

Lemma 1. $Adv_{A'}^{Verif}(VC_{Yao}, F, \lambda) \geq \varepsilon^{'}$ *where* $\varepsilon^{'}$ *is non-negligible in* λ.

Proof: This proof proceeds by defining a set of experiments.
$\mathcal{H}_A^k(VC_{edit}, F, \lambda)$: For $k = 0, ..., l-1$. Let l be an upper bound on the number of queries that A makes to its ProbGen oracle. Let i be a random index between 1 and l. In this experiment, we change the way the ProbGen oracle computes its answers. For the jth query:

- $j \leq k$ and $j \neq i$: The oracle will respond by (1) choosing a random key pair for the homomorphic encryption scheme $(PK_{\mathbb{E}}^j, SK_{\mathbb{E}}^j)$ and (2) encrypting random λ-bit strings under $PK_{\mathbb{E}}^j$.
- $j > k$ or $j = i$: The oracle will (1) generate a random key pair $(PK_{\mathbb{E}}^i, SK_{\mathbb{E}}^i)$ for the homomorphic encryption scheme and (2) encrypt σ_x (label by label) under $PK_{\mathbb{E}}^i$.

We denote with $Adv_A^k(VC_{edit}, F, \lambda)$= Prob$[\mathcal{H}_A^k(VC_{edit}, F, \lambda)$=1].

- $\mathcal{H}_A^0(VC_{edit}, F, \lambda)$ is identical to the experiment Exp_A^{Verif} $[VC_{Yao}, F, \lambda]$. Since the index i is selected at random between 1 and l, we have that

$$Adv_A^0(VC_{edit}, F, \lambda) = \frac{Adv_A^{Verif}(VC_{edit}, F, \lambda)}{l} \geq \frac{\varepsilon}{l} \tag{6}$$

- $\mathcal{H}_A^{l-1}(VC_{edit}, F, \lambda)$ equals the simulation conducted by A' above, so

$$Adv_A^{l-1}(VC_{edit}, F, \lambda) = Adv_{A'}^{Verif}(VC_{Yao}, F, \lambda) \tag{7}$$

If we prove $\mathcal{H}_A^k(VC_{edit}, F, \lambda)$ and $\mathcal{H}_A^{k-1}(VC_{edit}, F, \lambda)$ are computationally indistinguishable, that is for every A

$$\mid Adv_A^k[VC_{edit}, F, \lambda] - Adv_A^{k-1}[VC_{edit}, F, \lambda] \mid \leq negli(\lambda) \tag{8}$$

if we are done above, then that implies that

$$Adv_{A'}^{Verif}(VC_{Yao}, F, \lambda) \geq \frac{\varepsilon}{l} - l \cdot negli(\lambda) \tag{9}$$

The right of inequality is the desired non-negligible ε'. □

Remark: Eq.(8) follows from the security of the homomorphic encryption scheme. The reduction of the security of $\mathbb{E}$ with respect to Yao's garbled circuits to the basic security of $\mathbb{E}$ is trivial. For more details, please refer to [13].

6 Conclusions

This work treats the problem of secure outsourcing of sequence comparisons by a computationally limited client to two servers. To be specific, the client obtains the edit path of transforming a string of some length into another. We achieve this by integrating the techniques of garbled circuit and homomorphic encryption. In the proposed scheme, client can detect the dishonesty of servers according to a response returned from those servers. In particular, our construction re-garbles the circuit only when a malformed response comes from servers and hence is efficient. Also, the proposed construction is proved to be secure in the given security model.

Acknowledgements

We are grateful to the anonymous referees for their invaluable suggestions. This work is supported by the National Natural Science Foundation of China (Nos. 60970144 and 61272455), the Nature Science Basic Research Plan in Shaanxi Province of China (No. 2011JQ8042), and China 111 Project (No. B08038).

References

1. M. Atallah, F. Kerschbaum, and W. Du. Secure and private sequence comparisons. In *ACM Workshop on the Privacy in Electronic Society (WPES)*, 2003.
2. M. Atallah and J. Li. Secure outsourcing of sequence comparisons. In *Workshop on Privacy Enhancing Technologies (PET)*, pages 63−78, 2004.
3. M. Atallah and J. Li. Secure outsourcing of sequence comparisons. *International Journal of Information Security*, 4(4): 277−287, 2005.
4. M. Blanton and M. Aliasgari. Secure outsourcing of DNA searching via finite automata. In *DBSec*, pages 49−64, 2010.
5. Y. Huang, D. Evans, J. Katz, and L. Malka. Faster secure two-party computation using garbled circuits. In *USENIX Security Symposium*, 2011.
6. S. Jha, L. Kruger, and V. Shmatikov. Toward practical privacy for genomic computation. In *IEEE Symposium on Security and Privacy*, pages 216−230, 2008.
7. V. Kolesnikov, AR. Sadeghi, and T. Schneider. Improved garbled circuit building blocks and applications to auctions and computing minima. In *CANS*, vol. 5888, pages 1−20, 2009.

8. D. Szajda, M. Pohl, J. Owen, and B. Lawson. Toward a practical data privacy scheme for a distributed implementation of the Smith-Waterman genome sequence comparison algorithm. In *Network and Distributed System Security Symposium (NDSS)*, 2006.
9. A. Yao. How to generate and exchange secrets. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 162−167, 1986.
10. C. Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the ACM Symposium on the Theory of Computing (STOC)*, 2009.
11. M. Blanton, MJ. Atallah, KB. Frikken, and Q. Malluhi. Secure and Efficient Outsourcing of Sequence Comparisons. In *ESORICS*, LNCS, vol. 7459, pp.505−522, 2012.
12. Y. Lindell and B. Pinkas. A proof of Yao's protocol for secure two-party computation. *Journal of Cryptology*, 22(2): 161−188, 2009.
13. R. Gennaro, C. Gentry, B. Parno. Non-interactive Verifiable Computing Outsourcing Computation to Untrusted Workers. In *CRYPTO*, LNCS, vol.6223, pp.465−482, 2010.
14. B. Parno, M. Raykova, V. Vaikuntanathan. How to Delegate and Verify in Public: Verifiable Computation from Attribute-based Encryption. In *TCC*, LNCS, vol.7194, pp.422−439, 2012.
15. M. Blanton, Y. Zhang, KB. Frikken. Secure and Verifiable Outsourcing of Large-Scale Biometric Computations. In *IEEE International Conference on Information Privacy, Security, Risk and Trust (PASSAT)*, pages 1185−1191, 2011.
16. S. S. Vivek, S. S. D. Selvi, R. Venkatesan, and C. P. Rangan. A Special Purpose Signature Scheme for Secure Computation of Traffic in a Distributed Network. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA)*, 3(4): 46−60, December 2012.
17. J. Wang, H. Ma, Q. Tang, J. Li, H. Zhu, S. Ma, and X. Chen. A New Effcient Veriable Fuzzy Keyword Search Scheme. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA)*, 3(4): 61−71, December 2012.