Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2013, Article ID 172718, 5 pages
http://dx.doi.org/10.1155/2013/172718

Research Article

Efficient Secure Multiparty Computation Protocol for Sequencing Problem over Insecure Channel

Yi Sun,[1] Qiaoyan Wen,[1] Yudong Zhang,[2] Hua Zhang,[1] and Zhengping Jin[1]

[1] *State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China*
[2] *School of Computer Science and Technology, Nanjing Normal University, Nanjing, Jiangsu 210023, China*

Correspondence should be addressed to Yi Sun; sybupt@bupt.edu.cn

Received 2 March 2013; Accepted 2 August 2013

Academic Editor: Vishal Bhatnagar

As a powerful tool in solving privacy preserving cooperative problems, secure multiparty computation is more and more popular in electronic bidding, anonymous voting, and online auction. Privacy preserving sequencing problem which is an essential link is regarded as the core issue in these applications. However, due to the difficulties of solving multiparty privacy preserving sequencing problem, related secure protocol is extremely rare. In order to break this deadlock, this paper first presents an efficient secure multiparty computation protocol for the general privacy-preserving sequencing problem based on symmetric homomorphic encryption. The result is of value not only in theory, but also in practice.

1. Introduction

Sequencing problem is very common in our daily life, such as ranking according to the scores, queuing by the height. Informally speaking, it is about comparing and sequencing of some numbers. It is easy and convenient to get the result because it cares nothing about privacy in the scenes above. On the contrary, privacy-preserving sequencing problem (PPSP) is always a hard challenge since it requires to conduct secret numbers comparison without knowing the numbers. In this scenario, all participants distrust each other and would not like to leak their own secret information to anyone else. It is an urgent task to be solved for some important applications such as electronic bidding, anonymous voting, and online auction. Naturally, as a powerful tool in solving privacy-preserving cooperative problems, secure multiparty computation (SMC) [1] is the best choice for privacy-preserving sequencing. In fact, the classical Millionaire's problem [1–3] is the earliest example of introducing secure multiparty computation into the sequencing problem. More specifically, the millionaire's problem, with the aim to find out which one of the two Millionaires is richer without revealing their net worth, can be described as comparing two secret numbers in the perspective of sequencing, that is, the 2-party case of PPSP. In this aspect, the case of 2-party sequencing problem has already been resolved along with the advent of the solutions to Millionaire's problem and the presence of other secure two-party computation protocols [4–12]. Due to the limitation of the 2-party case in practice, the general multiparty PPSP becomes the focus in secure multiparty computation recently.

In 1962, Held and Karp [13] put forward a dynamic programming approach to multiparty sequencing problem before the advent of SMC. They concern more about some certain scenarios and aim to design schemes for the special applications such as the traveling-salesman problem. Subsequently, the research on PPSP is rare and mainly about the 2-party case. Currently, Tang et al. [14] have constructed an efficient and secure multiparty computation protocol for PPSP by making use of a secret sharing scheme based on polynomial. It is an important fruit of PPSP since it has indeed realized secure sequencing among distrusted participants. However, the cost is too high in choosing random numbers and transmitting messages. In the case of n parties with

t adversaries, it needs to choose $2n \cdot (2t + 1) + n$ polynomials and $2n{\cdot}(t{+}1)$ random numbers. What is more, the transmitted messages are up to $(2t + 1) \cdot (n - 1) \cdot n + (n - 1) \cdot n + n^2(2t + 1)$ every round.

This paper applies the fast symmetric homomorphic encryption to replace the cumbersome secret sharing based on polynomial. It no longer needs to choose so many polynomials and random numbers. Relevant complexities in computation and communication also have a great improvement. Our result is not only much simpler but also more efficient. In brief, our contributions can be summarized as follows.

(1) We first introduce symmetric homomorphic encryption to solve the privacy-preserving sequencing problem in secure multiparty computation, which brings less communications and random numbers than the method of secret sharing based on polynomial.

(2) Our protocol is appropriate for the insecure channel which allows external attackers to eavesdrop and can resist at most $t < n/2$ adversaries' corruption supposing that any two neighbor parties do not conspire.

(3) We propose a protocol for the general privacy-preserving sequencing problem, which is suitable for multiple parties to securely determine the order of a given set rather than just two parties such as the simplest sequencing problem-Millionaire's problem, or a special application such as the traveling-salesman problem.

Organization. The rest of this paper is organized as follows. In Section 2, we briefly give some related preliminaries. In Section 3, we present the new efficient secure multiparty computation protocol for privacy-preserving sequencing problem over insecure channel. In Section 4, we analyze the proposed protocol in detail including its correctness and privacy. Furthermore, we show the advantages of our protocol in the two aspects of transmitted messages and random numbers. Finally, we summarize our work of this paper in the last section.

2. Preliminaries

2.1. Secure Multiparty Computation. Secure multiparty computation is dedicated to dealing with the problem of privacy-preserving cooperative computation among distrusted participants. It was first introduced by Yao in 1982 [1] by putting forward the famous Millionaire's problem. Afterwards, SMC has become a research focus in the international cryptographic community, and a mass of research results have been published one after the other [2–12].

Generally speaking, SMC is a method to implement cooperative computation with all participants' private data, ensuring the correctness of the computation as well as not disclosing additional information except the necessary results. Assume that there are n participants $P_1, P_2, \dots, P_n$. Each has a secret, respectively, $S_1, S_2, \dots, S_n$. They want to compute the value of a public function $F(\cdot)$ on n variables at the point $(S_1, S_2, \dots, S_n)$, that is, $F(S_1, S_2, \dots, S_n)$. An SMC protocol is dubbed secure if no participant can learn more from the description of the public function and the result of the global calculation than what he can learn from his own information.

2.2. Homomorphic Encryption. In this subsection, we introduce a basic tool to design our protocol, the symmetric homomorphic encryption scheme. Allowing for security, the participants usually would not like to directly transmit their original data over insecure channel while interacting with others. They expect that other parties can perform necessary computations on the encrypted version of the data. In this way, they can encrypt their own private information and then transmit it to others without exposing the real data and finally decrypt the information sent back by others to get the target result when completing cooperative computation. To meet this demand, Rivest et al. proposed homomorphic encryption in 1978 [15]. His work sparked the research in this field. A lot of articles have been proposed and widely used in many applications since then. However, the most common homomorphic encryption schemes are mainly asymmetric, for example, ELGamal homomorphic encryption scheme and Paillier' homomorphic encryption scheme.

Although symmetric homomorphic encryption has not been used in PPSP, it is really a promising method for secure multiparty computation while dealing with the problem of privacy-preserving sequencing. The symmetry will bring high efficiency to our solution since symmetric encryption possesses the advantage of being really fast and can be used as often as possible. As illustrated in [16], a block cipher like AES is typically 100 times faster than RSA encryption and 2000 times than RSA decryption, with about 60 MB per second on a modest platform. Stream ciphers are even faster, some of them being able to encrypt/decrypt 100 MB per second or more. Therefore, asymmetric homomorphic encryptions are bound to much slower than the symmetric ones. In this paper, we will employ the superior symmetric homomorphic encryption schemes to construct our protocol.

Generally, an encryption scheme is said to be homomorphic if for any given encryption key k, the encryption function $E(\cdot)$ satisfies the following condition:

$$\forall m_1, m_2 \in P, \quad E(m_1 \odot_P m_2) = E(m_1) \odot_C E(m_2), \tag{1}$$

where $P(C)$ denotes the set of the plaintexts (ciphertexts), and $\odot_P$ and $\odot_C$ are the operators in P and C.

We say that a scheme is additively homomorphic if we consider addition operators, and it is multiplicatively homomorphic if we consider multiplication operators. Usually, multiplicative homomorphic encryption functions are more efficient than additive homomorphic encryption functions.

Herein, we will use the random symmetric homomorphic encryption function $E(\cdot)$ in this paper, which satisfies the following property:

$$\forall m_1, m_2 \in Q^+, \quad E(m_1 + m_2) = E(m_1) * E(m_2), \tag{2}$$

where $E(\cdot)$ is a random function and Q is the set of rational numbers.

It is easy to deduce that for all $m \in Q^+$, $r \in Z^+$,

$$E(r * m) = E(m)^r. \tag{3}$$

2.3. Privacy-Preserving Sequencing Problem

2.3.1. The Original Problem. Privacy-preserving sequencing problem is in fact the more universal description of the generalized secret number comparison. To be more specific, there are n distrusted participants $P_1, P_2, \ldots, P_n$. Each of them has a private number, respectively, $S_1, S_2, \ldots, S_n$. The problem is that they hope to rank the n-array $(S_1, S_2, \ldots, S_n)$ without leaking any information about $S_1, S_2, \ldots, S_n$. It requires that after executing cooperative computation, $P_1, P_2, \ldots, P_n$ know the size relations of $S_1, S_2, \ldots, S_n$ but no more other information. Formally, we can represent the whole problem as shown in Algorithm 1.

2.3.2. Equivalent Transformation of the Original Problem. In this paper, we make use of a useful theorem in the progressing procedure following reference [14] so that we can reduce the initial sequencing problem about the n-array $(S_1, S_2, \ldots, S_n)$ to the new n-array $(S_1', S_2', \ldots, S_n')$, which has the same sequence as $(S_1, S_2, \ldots, S_n)$ and is called as the pseudoarray of $(S_1, S_2, \ldots, S_n)$. Then $P_1, P_2, \ldots, P_n$ can obtain the sequence of $S_1, S_2, \ldots, S_n$ by directly comparing the pseudoarrays $(S_1', S_2', \ldots, S_n')$ in public. Along with the equivalent transformation of the problem, the aim of secure multiparty computation needs a corresponding change. It no longer has to consider how to deal with the real data $S_1, S_2, \ldots, S_n$ but only needs to securely get the pseudodata $S_1', S_2', \ldots, S_n'$. And then the subsequent work is just a piece of cake.

Theorem 1. *Arrays $(S_1, S_2, \ldots, S_n)$ and $(S_1', S_2', \ldots, S_n')$ have the same sequence, where $S_i' = r_1 * S_i + r_2 * S_i^2 + \cdots + r_n * S_i^n$, $r_i \geq 0$, $S_i \geq 0$, $i = 1, 2, \ldots, n$.*

Proof. Given for all $S_i', S_j' \in (S_1', S_2', \ldots, S_n')$

$$\begin{aligned} S_i' &= r_1 * S_i + r_2 * S_i^2 + \cdots + r_n * S_i^n, \\ S_j' &= r_1 * S_j + r_2 * S_j^2 + \cdots + r_n * S_j^n. \end{aligned} \tag{4}$$

Then,

$$\begin{aligned} S_i' - S_j' &= \left(r_1 * S_i + r_2 * S_i^2 + \cdots + r_n * S_i^n\right) \\ &= \left(r_1 * S_i + r_2 * S_i^2 + \cdots + r_n * S_i^n\right) \\ &\quad - \left(r_1 * S_j + r_2 * S_j^2 + \cdots + r_n * S_j^n\right) \\ &= r_1 * \left(S_i - S_j\right) + r_2 * \left(S_i^2 - S_j^2\right) + \cdots + r_n * \left(S_i^n - S_j^n\right) \\ &= \left(S_i - S_j\right)\left[r_1 + r_2 * \left(S_i + S_j\right)\right. \\ &\qquad + r_3 * \left(S_i^2 + S_i \cdot S_j + S_j^2\right) + \cdots \\ &\qquad \left. + r_n * \left(S_i^{n-1} + S_i^{n-2} \cdot S_j + \cdots + S_i \cdot S_j^{n-2} + S_j^{n-1}\right)\right]. \end{aligned} \tag{5}$$

Input: $(S_1, S_2, \ldots, S_n)$, S_i is the private number of P_i;
Output: $(l_1, l_2, \ldots, l_n)$, l_i is the order of S_i in the n-array.

ALGORITHM 1

Let

$$\begin{aligned} Q = r_1 + r_2 * \left(S_i + S_j\right) + r_3 * \left(S_i^2 + S_i \cdot S_j + S_j^2\right) + \cdots \\ + r_n * \left(S_i^{n-1} + S_i^{n-2} \cdot S_j + \cdots + S_i \cdot S_j^{n-2} + S_j^{n-1}\right). \end{aligned} \tag{6}$$

Then, $S_i' - S_j' = (S_i - S_j) \cdot Q$. As we know that $r_i \geq 0$, $S_i \geq 0$, $i = 1, 2, \ldots, n$. Therefore, $Q \geq 0$. That means, for all $S_i', S_j' \in (S_1', S_2', \ldots, S_n')$, S_i', S_j' and S_i, S_j have the same sequence. Obviously, $(S_1, S_2, \ldots, S_n)$ and $(S_1', S_2', \ldots, S_n')$ have the same sequence. □

3. Proposed Protocol

In this section, we present our protocol. The simplified version of the protocol is briefly illustrated in Figure 1, and the details can be described as follows.

Initialization. Assume that there are n participants $P_1, P_2, \ldots, P_n$, each P_i owning a secret number S_i and a random symmetric homomorphic encryption function $E_i(\cdot)$.

Computation

(1) P_i chooses a random number $r_i > 0$, $i = 1, 2, \ldots, n$, and computes $E_i(S_i), E_i(S_i^2), \ldots, E_i(S_i^{i-1})$, $E_i(S_i^{i+1}), \ldots, E_i(S_i^n)$ and $E_i(r_i * S_i^i)$ locally. For $i = 1, 2, \ldots, n$, $j = 1, 2, \ldots, n$, $i \neq j$, P_i sends $E_i(S_i^j)$ to P_j.

(2) After receiving $E_i(S_i^j)$, P_j computes $E_i(S_i^j)^{r_j}$, $i = 1, 2, \ldots, n$, $j = 1, 2, \ldots, n$, $i \neq j$. And then, P_j transmits $E_i(S_i^j)^{r_j}$ to P_{i+1}, $i = 1, 2, \ldots, n-1$, $j = 1, 2, \ldots, n$, $j \neq i+1$. For $i = n$, P_j transfers $E_n(S_n^j)^{r_j}$ to P_1, $j = 2, \ldots, n$.

(3) P_1 computes $S_n'' = E_n(r_1 * S_n + r_2 * S_n^2 + \cdots + r_n * S_n^n)$ and sends S_n'' to P_n; For $i = 1, 2, \ldots, n-1$, P_{i+1} computes $S_i'' = E_i(r_1 * S_i + r_2 * S_i{}^2 + \cdots + r_n * S_i^n)$ and sends S_i'' to P_i.

(4) P_i computes $S_i' = D_i(S_i'')$, $i = 1, 2, \ldots, n$ and broadcasts S_i' to obtain the sequence of the n-array $(S_1, S_2, \ldots, S_n)$ by comparing the size of the pseudoarray $(S_1', S_2', \ldots, S_n')$.

4. Analysis

In this section, we have an analysis of the proposed protocol in the aspects of security and efficiency. To guarantee that it is a secure multiparty computation protocol, we have to prove that it satisfies correctness and privacy requirements at first.

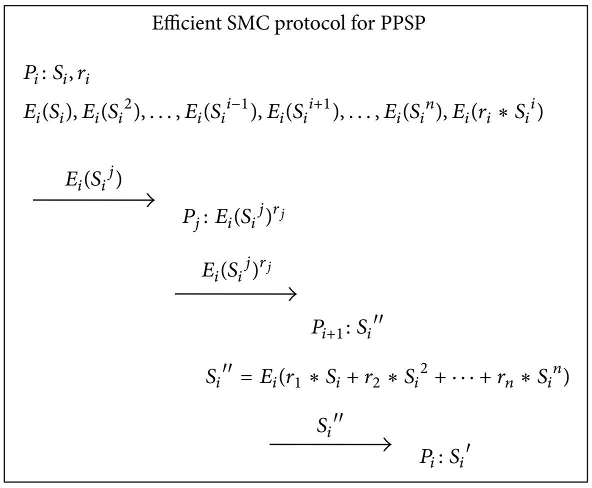
Efficient SMC protocol for PPSP
$P_i: S_i, r_i$
$E_i(S_i), E_i(S_i^2), \ldots, E_i(S_i^{i-1}), E_i(S_i^{i+1}), \ldots, E_i(S_i^n), E_i(r_i * S_i^i)$
$E_i(S_i^j)$
$P_j: E_i(S_i^j)^{r_j}$
$E_i(S_i^j)^{r_j}$
$P_{i+1}: S_i''$
$S_i'' = E_i(r_1 * S_i + r_2 * S_i^2 + \cdots + r_n * S_i^n)$
S_i''
$P_i: S_i'$

Figure 1: Simplified version of the proposed protocol.

4.1. Correctness. Assume that the attacker is passive. Then, all participants (including all attackers and honest participants) correctly follow the protocol. Therefore, we only need to examine whether the protocol will give the correct sequence for the array $(S_1, S_2, \ldots, S_n)$. From the proof of Theorem 1, we know that the array $(S_1, S_2, \ldots, S_n)$ have the same sequence with the pseudoarray $(S_1', S_2', \ldots, S_n')$. Thus, the proposed protocol can correctly achieve the aim of sequencing the array $(S_1, S_2, \ldots, S_n)$ by comparing the pseudoarrays $(S_1', S_2', \ldots, S_n')$ secretly. Hence, the protocol satisfies correctness.

4.2. Privacy. According to the definition of privacy in multiparty computation protocols in [17], the protocol is private if the protocol satisfies the following conditions.

(a) The information string viewed by each participant P_i and a random string with the same length have the same probability distribution. That is, the information string and the random string are indistinguishable.

(b) Arbitrary $t < n/2$ participants cannot jointly obtain any information about the input of any other participant.

In fact, in the proposed protocol, the viewed information strings of P_i are $E_i(S_i), E_i(S_i^2), \ldots, E_i(S_i^{i-1}), E_i(S_i^{i+1}), \ldots, E_i(S_i^n)$, and $E_i(r_i * S_i^i)$ in the first step; $E_i(r_i * S_i^i)$, $E_j(S_j^i)$, $E_j(S_j^i)^{r_i}$, $i = 1, 2, \ldots, n$, $j = 1, 2, \ldots, n$, $i \neq j$ in the second step; S_{i-1}'', $i = 2, \ldots, n$, specially, for $i = 1$, the viewed information string in this step is S_n'' for P_1; finally, in the last step, the viewed string is S_i' for P_i. All the strings are generated by the random symmetric homomorphic encryption function $E_i(\cdot)$. Therefore, the strings viewed by P_i and a random string with the same length have the same probability distribution, and (a) is satisfied.

Moreover, our protocol also satisfies that arbitrary $t < n/2$ participants cannot jointly obtain any information about the input of any other participant under the assumption that any two neighbor parties never conspire. Since we have P_{i+1} to compute S_i'' for P_i by collecting $E_i(S_i^j)^{r_j}$, $j = 1, \ldots, n$, it is obvious that it is insecure if P_i and P_{i+1} collude. It is reasonable to suppose that no neighbors collude because the two adversaries are not exactly adjacent since they cannot control the order of the parties when executing the protocol. In addition, if an adversary wants to get more information from $S_i'' = E_i(r_1 * S_i + r_2 * S_i^2 + \cdots + r_n * S_i^n)$, he must corrupt with at least $n-1$ parties since there are $n-1$ unknown coefficients as well as breaking the encryption scheme $E(\cdot)$.

What is more, in our protocol, all information strings are transmitted in the encrypted forms. The private information S_i and r_i are secret as long as the encryption function $E_i(\cdot)$ is robust. In other words, it is secure even over the insecure channel, which is better than the previous protocol based on polynomial for the sequencing problem.

In short, our protocol is correct and private.

Table 1: Efficiency comparison.

Item	Our protocol	Tang's protocol
Random numbers	n	$2n(t+1)$
Transmitted messages	$n(2n-1)$	$(2t+1)(n-1)n + (n-1)n + n^2(2t+1)$

Table 2: Efficiency comparison.

Item	Our protocol	Tang' protocol
Random numbers	n	$n(n+1)$
Transmitted messages	$n(2n-1)$	$2n^3 - n$

4.3. Efficiency. Our protocol is efficient as well as secure. It operates better than the previous one because it is independent of the secret sharing scheme based on complex polynomial. We can make a concrete comparison between the proposed protocol and the previous one on the numbers of random numbers and transmitted messages as in Table 1.

From Table 1, we can easily find that in Tang' protocol [14], it needs to choose $n \cdot (2t+1)$ polynomials for $f(\cdot)$, n polynomials for $r(\cdot)$, and $n \cdot (2t+1)$ polynomials for $h(\cdot)$, that is, totally $2n \cdot (t+1)$ random numbers as well as $2n \cdot (2t+1) + n$ polynomials; it also needs to transmit $f_{ik}(x_j)$ from P_i to P_j, $i, j = 1, 2, \ldots, n$, $k = 1, 2, \ldots, 2t+1$, $r_i(x_j)$, $h_{ik}(x_j)$ from P_i to P_j, $i = 1, 2, \ldots, 2t+1$, $j, k = 1, 2, \ldots, n$; thus, $(2t+1) \cdot (n-1) \cdot n + (n-1) \cdot n + n^2(2t+1)$ messages are needed to be transmitted totally.

In our protocol, it only needs to choose n random numbers in the whole procedure. And the messages that need to be transmitted are, respectively, $E_i(S_i^j)$, $i = 1, 2, \ldots, n$, $j = 1, 2, \ldots, n$, $i \neq j$, $E_i(S_i^j)^{r_j}$, $i = 1, 2, \ldots, n-1$, $j = 1, 2, \ldots, n$, $j \neq i+1$, and $E_n(S_n^j)^{r_j}$, $j = 2, \ldots, n$, and S_i'', $i = 1, 2, \ldots, n$, totally $n \cdot (2n-1)$ messages. It is much simpler and more appropriate for the clients who expect easier products in practice.

If there are $t = (n-1)/2$ adversaries (the upper bound of the adversaries in Tang' protocol [14]), the advantages of our protocol are more obvious as shown in Table 2.

5. Conclusion

It is always a difficult problem in the cryptographic field to construct a secure multiparty computation protocol for the privacy-preserving sequencing problem. In the present study, we have successfully designed an efficient secure multiparty computation protocol for sequencing problem over insecure channel based on symmetric homomorphic encryption, which is of great importance to the theory on this topic and of significant value in practice for its high efficiency.

Acknowledgments

This work is supported by NSFC (Grant nos. 61170270, 61100203, 60903152, 61003286, and 61121061) and the Fundamental Research Funds for the Central Universities (Grant nos. BUPT2011YB01, BUPT2011RC0505, 2011PTB-00-29, 2011RCZJ15, and 2012RC0612).

References

[1] A. C. Yao, "Protocols for secure computations," in *Proceedings of the 23rd Annual IEEE Symposium on Foundations of Computer Science*, pp. 160–164, Chicago, Ill, USA, 1982.

[2] Y. Lindell and B. Pinkas, "A proof of security of Yao's protocol for two-party computation," *Journal of Cryptology*, vol. 22, no. 2, pp. 161–188, 2009.

[3] O. S. Goldreich, S. Mical, and A. Wigderson, "How to play any mental game," in *Proceedings of the 19th Annual ACM Symposium on Theory of Computing (STOC '87)*, pp. 218–229, ACM, New York, NY, USA, 1987.

[4] R. Fagin, M. Naor, and P. Winkler, "Comparing information without leaking it," *Communications of the ACM*, vol. 39, no. 5, pp. 77–85, 1996.

[5] B. Schoenmakers and P. Tuyls, "Practical two-party computation based on the conditional gate," in *Advances in Cryptology: ASIACRYPT 2004, 10th International Conference on the Theory and Application of Cryptology and Information Security*, pp. 119–136, Springer, Jeju Island, Korea, 2004.

[6] I. F. Blake and V. Kolesnikov, "Strong conditional oblivious transfer and computing on intervals," in *Advances in Cryptology: ASIACRYPT 2004, 10th International Conference on the Theory and Application of Cryptology and Information Security*, pp. 515–529, Springer, Jeju Island, Korea, 2004.

[7] Y. L. Luo, *Some key issues in secure multi-party computation and their applied research [Ph.D. dissertation]*, University of Science and Technology of China, 2005 (Chinese).

[8] M. Fischlin, "A cost-effective pay-per-multiplication comparison method for millionaires," in *Topics in Cryptology: CT-RSA 2001, The Cryptographers' Track at RSA Conference 2001*, pp. 457–471, Springer, San Francisco, Calif, USA, 2001.

[9] C. Cachin, "Efficient private bidding and auctions with an oblivious third party," in *Proceedings of the 1999 6th ACM Conference on Computer and Communications Security*, pp. 120–127, ACM, New York, NY, USA, November 1999.

[10] J. Qin, Z.-F. Zhang, D.-G. Feng, and B. Li, "Protocol of comparing information without leaking," *Journal of Software*, vol. 15, no. 3, pp. 421–427, 2004.

[11] H. Y. Lin and W. G. Tzeng, "An efficient solution to the millionaires problem based on homomorphic encryption," ASIACRYPT 2005, http://eprint.iacr.org/2005/043.

[12] I. Ioannidis and A. Grama, "An efficient protocol for Yao's Millionaires' problem," in *Proceedings of the 36th Hawaii International Conference on System Sciences*, Maui, Hawaii, USA, 2003, Track 7.

[13] M. Held and R. M. Karp, "A dynamic programming approach to sequencing problems," *Journal of the Society for Industrial and Applied Mathematics*, vol. 10, pp. 196–210, 1962.

[14] C. Tang, G. Shi, and Z. Yao, "Secure multi-party computation protocol for sequencing problem," *Science China. Information Sciences*, vol. 54, no. 8, pp. 1654–1662, 2011.

[15] R. L. Rivest, L. Adleman, and M. L. Dertouzos, "On data banks and privacy homomorphisms," in *Foundations of Secure Computation*, pp. 169–179, Academic Press, 1978.

[16] C. Fontaine and F. Galand, "A survey of homomorphic encryption for nonspecialists," *Eurasip Journal on Information Security*, vol. 2007, Article ID 13801, 2007.

[17] M. Ben-Or, S. Goldwasser, and A. Wigderson, "Completeness theorems for non-cryptographic fault-tolerant distributed computation," in *Proceedings of the 20th Annual ACM symposium on Theory of Computing (STOC '88)*, pp. 1–11, 1988.