A Practical Privacy-Preserving Cooperative Computation Protocol without Oblivious Transfer for Linear Systems of Equations

Ju-Sung Kang* and Dowon Hong **

Abstract: We propose several practical SMC protocols for privacy-preserving cooperative scientific computations. We consider two important scientific computations which involve linear equations: the linear systems of equations problem and the linear least-square problem. The protocols proposed in this paper achieve acceptable security in the sense of Du-Zhan's paradigm and t-wise collusion-resistance, and their communication complexity is $\boldsymbol{O}(tm)$, where t is a security parameter and m is the total number of participants. The complexity of our protocol is significantly better than the previous result $\boldsymbol{O}(m^2 l\mu)$ of [4], in which the oblivious transfer protocol is used as an important building block. .

Keywords: *SMC. Practical SMC, Privacy, Linear system of equations*

1. Introduction

The goal of multi-party computation, as advanced by Yao[1] in 1982, is to enable a set of players to evaluate an arbitrary function on their private inputs. The problem is that the computation must guarantee the correctness of the result while preserving the privacy of the players' inputs. This problem is referred to as the Secure Multi-Party Computation (SMC) problem in [1]. Hypothetically speaking, imagine we have a trusted third party to which all parties give their inputs. The trusted party computes the output and returns it to the parties. The SMC protocol enables this without the trusted third party. The general SMC problem is theoretically solvable by the circuit evaluation method[1,2,3]. While this theoretical approach is appealing in its generality and simplicity, the protocols it generated depend on the size of the circuit. This size depends on the size of the input domain, and on the expressional complexity of a computation. Goldreich points out in [3] that using the solutions derived by these general results for special cases of multi-party computation can be impractical. Hence special solutions should be developed for special cases for reasons of efficiency.

Recently, specific SMC problems for various computation domains have been identified and efficient SMC protocols to solve the corresponding problems have been proposed. Those protocols include privacy-preserving cooperative computations for scientific computations[4], database query[5], intrusion detection[6], statistical analysis[7], geometric computations[8], and data mining[9]. However, these solutions, although achieving ideal security, are still not efficient enough for practical uses.

On the other hand, Du and Zhan[10] have proposed a new paradigm, in which they aimed at developing practical solutions to SMC problems and introduced an acceptable security model that allows partial information disclosure. Their conjecture is that by lowering the restriction on the security, they can achieve a much better performance. Du and Zhan[10] have developed a number of techniques under this new paradigm.

In this paper, we propose several practical SMC protocols for privacy-preserving cooperative scientific computations under the Du-Zhan's paradigm. We consider two important scientific computations which involve linear equations: the linear systems of equations and the linear least-squares problems. Du and Atallah[4] have formally defined the privacy-preserving cooperative scientific computation problems of the 2-party model, and presented protocols to solve them. In the Du-Atallah's protocols, the 1-out-of-N oblivious transfer protocol of [11] was used as an important building block. Since all the 1-out-of-N oblivious transfer protocols are still not practical, Du-Atallah's protocol, although achieving the ideal security, is also impractical. In fact, we will show that Du-Atallah's 2-party protocol is especially inefficient when we simply extend their protocol to the m-party ($m \geq 3$) version. Based on this investigation, we present several practical m-party protocols for privacy-preserving cooperative scientific computations.

Manuscript received November 24, 2006; accepted January 7, 2007.
This work was supported by the IT R&D program of MIC/IITA. [2005-Y001-04, Development of next generation security technology]
Corresponding Author: Ju-Sung Kang
* Dept. of Mathematics, Kookmin University, Seoul, Korea (jskang@kookmin.ac.kr)
** Information Security Research Division, ETRI, Daejeon, Korea (dwhong@etri.re.kr)

2. Practical SMC

Du and Zhan[10] proposed a security paradigm to study the SMC problems based on an acceptable security model. They referred to this problem as the Practical SMC (PSMC) problem. The PSMC problem deals with computing any function on any input, in a distributed network where each party holds one of the inputs, ensuring that only partial information is revealed to a participant in the computation. An informal definition of acceptable security is as follows[10]:

A protocol achieves acceptable security, if an adversary can only narrow down all the possible values of the secret data to a domain with the following properties:

1. The number of values in this domain is infinite, or the number of values in this domain is so large that a brute-force attack is computationally infeasible.
2. The range of the domain is acceptable for the application. The definition of the acceptable range depends on specific applications.

In [10], the authors designed the Commodity-Server Model to achieve practical security. This model introduces an extra server, the commodity server, belonging to a third party. The only requirement imposed on the commodity server is that it cannot collude with any participants. In this paper we develop some protocols under this computation model. The Commodity-Server has a few appealing properties. First, this server does not participate in the computation between participants, but it does provide data for them to hide their private data. Second, the data provided by the Commodity-Server does not depend on the participants' private data. With these properties, the Commodity-Server can generate independent off-line data, and sell them as commodities to the participants. The Commodity-Server model is depicted in Figure 1.

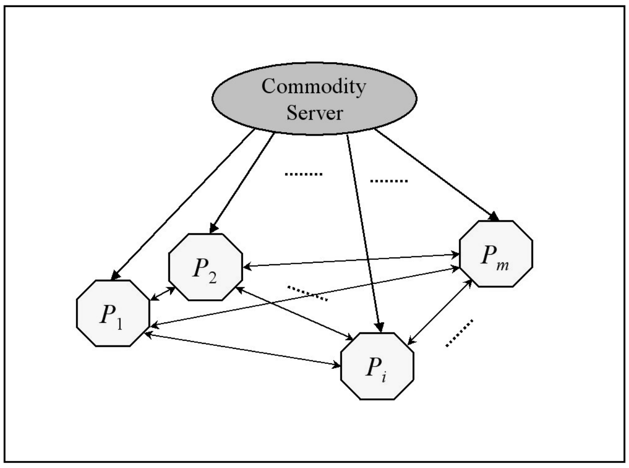

Fig. 1. Commodity-Server Model

3. Problem Description

We consider the situations that could usually be modeled as linear systems of equations problems or linear least square problems. These scientific computation problems have shown themselves to be valuable for modeling many types of problems in planning, routing, scheduling, assignment, and design. We assume that there are m parties $P_1, P_2, \dots, P_m$ engaged in the computation, where $m \geq 2$, and all parties want to jointly conduct the computations based on the private inputs.

A system of n linear equations in n unknown variables may be written as a single vector equation $\boldsymbol{Ax} = \boldsymbol{b}$, where $\boldsymbol{A}$ is the $n\times n$ coefficient matrix, and $\boldsymbol{x}$ and $\boldsymbol{b}$ are the $n\times 1$ column vectors. Assume that each party P_j $(1\leq j \leq m)$ has k_j equations as its private information and that $k_1 + \dots + k_m = n$. Then, $\boldsymbol{Ax} = \boldsymbol{b}$ can be written as $(\boldsymbol{A}_1 + \dots + \boldsymbol{A}_m)\,\boldsymbol{x} = (\boldsymbol{b}_1 + \dots + \boldsymbol{b}_m)$, where $\boldsymbol{A}_j$ and $\boldsymbol{b}_j$ $(1\leq j \leq m)$ are the $n\times n$ and $n\times 1$ matrices whose entries are all zero except for the k_j rows corresponding private information of P_j, respectively. The problem of privacy-preserving cooperative multi-party computation for the linear system of equations (PCMC-LSE) is defined as follows.

Definition 1. (PCMC-LSE) Assume that there are m parties $P_1, P_2, \dots, P_m$ $(m \geq 2)$ engaged in the computation, and that each party P_j $(1\leq j \leq m)$ provides k_j $(k_1 + \dots + k_m = n)$ equations as its private input with the corresponding matrices $\boldsymbol{A}_j$ and $\boldsymbol{b}_j$, where $\boldsymbol{A}_j$ and $\boldsymbol{b}_j$ $(1\leq j \leq m)$ are the $n\times n$ and $n\times 1$ matrices whose entries are all zero except for the k_j rows, respectively. The PCMC-LSE problem is that all parties know the solution $\boldsymbol{x}$ of $(\boldsymbol{A}_1 + \dots + \boldsymbol{A}_m)\,\boldsymbol{x} = (\boldsymbol{b}_1 + \dots + \boldsymbol{b}_m)$ while nobody knows another party's private input.

There are situations where we have more equations to satisfy than the number of unknown variables. Most often we cannot satisfy all of these equations. However we may find a solution that can satisfy them as best as we can. This problem is called the linear least-squares problem. We refer to the privacy-preserving cooperative multi-party computation version for the linear least-squares problem as PCMC-LLS.

Definition 2. (PCMC-LLS) Assume that there are m parties $P_1, P_2, \dots, P_m$ $(m \geq 2)$ engaged in the computation, and that each party P_j $(1\leq j \leq m)$ provides matrices $\boldsymbol{A}_j$ and $\boldsymbol{b}_j$ as its private input, where $\boldsymbol{A}_j$ and $\boldsymbol{b}_j$ $(1\leq j \leq m)$ are the $l \times n$ $(l > n)$ and $l\times 1$ matrices whose entries are all zero except for the k_j $(k_1 + \dots + k_m = l)$ rows, respectively. The PCMC-LLS problem here is that all parties know the linear least-squares solution $\boldsymbol{x}$ of $(\boldsymbol{A}_1 + \dots + \boldsymbol{A}_m)\,\boldsymbol{x} = (\boldsymbol{b}_1 + \dots + \boldsymbol{b}_m)$ while nobody knows any other party's private input. The linear least-squares solution $\boldsymbol{x}$ of $\boldsymbol{Ax} = \boldsymbol{b}$ minimizes $\| \boldsymbol{b} - \boldsymbol{Ax} \|$, where $\| \cdot \|$ denotes the Euclidean norm.

4. Practical Privacy-Preserving Cooperative Scientific Computations

In this section, we describe the PCMC-LSE and PCMC-LLS protocols to solve the corresponding problems of Definition 1 and 2, respectively. Firstly, we present a secure split protocol that is used as a building block of the other two protocols. The goal of the secure split protocol is to blind individual shares in such a way that no share reveals private information, but the sum total of all shares remains the same as before. This protocol is a matrix version of Protocol 1 of [12].

Protocol 1. (Secure Split Protocol)

Input: Each party P_j $(1\le j \le m)$ provides $l \times n$ matrix $\mathbf{A}_j$ as a private input.

Output: Each party P_j $(1\le j \le m)$ obtains $l \times n$ matrix $\mathbf{B}_j$ such that $\sum_{j=1}^{m} \mathbf{A}_j = \sum_{j=1}^{m} \mathbf{B}_j$.

1. All parties jointly agree on a security parameter t, such that $1\le t \le m$-1.
2. The Commodity-Server generates $t{\cdot}m$ random $l \times n$ matrices $\mathbf{R}_1^{(1)}, \dots, \mathbf{R}_t^{(1)}, \cdots, \mathbf{R}_1^{(m)}, \dots, \mathbf{R}_t^{(m)}$ whose entries are a sufficiently large random number and sends $\mathbf{R}_1^{(j)}, \dots, \mathbf{R}_t^{(j)}$ to each party P_j $(1\le j \le m)$, respectively.
3. Each party P_j $(1\le j \le m)$ sends $\mathbf{R}_1^{(j)}, \dots, \mathbf{R}_t^{(j)}$ to randomly chosen t parties from the remaining m-1 parties. Party P_j sets its share of $\mathbf{A}_j$ to be $\mathbf{R}_0^{(j)} = \mathbf{A}_j - (\mathbf{R}_1^{(j)} + \dots + \mathbf{R}_t^{(j)})$.
4. After receiving t' matrices from other parties, each party P_j computes $\mathbf{B}_j = \mathbf{R}_0^{(j)} + \sum_{i\neq j} \mathbf{R}_{k_i}^{(i)}$, where party P_i sent $\mathbf{R}_{k_i}^{(i)}$ to P_j for some $1\le k_i \le t$.

From step 4 of Protocol 1, we obtain that $\sum_{j=1}^{m} \mathbf{A}_j = \sum_{j=1}^{m} \mathbf{B}_j$, since $\sum_{j=1}^{m} \sum_{i=1}^{t} \mathbf{R}_i^{(j)} = \sum_{j=1}^{m} \sum_{i\neq j} \mathbf{R}_{k_i}^{(i)}$. By using Protocol 1, we can design a practical protocol to solve the PCMC-LSE problem under the Commodity-Server model.

Protocol 2. (PCMC-LSE Protocol)

Input: Each party P_j $(1\le j \le m)$ provides a matrix $\mathbf{A}_j$ and a column vector $\mathbf{b}_j$ as its private inputs, where $\mathbf{A}_j$ and $\mathbf{b}_j$ $(1\le j \le m)$ are $n{\times}n$ and $n{\times}1$ matrices whose entries are all zero except for the k_j $(k_1+ \dots + k_m = n)$ rows, respectively.

Output: Each party P_j $(1\le j \le m)$ obtains the solution $\mathbf{x}$ of $(\mathbf{A}_1 + \dots + \mathbf{A}_m)\,\mathbf{x} = (\mathbf{b}_1 + \dots + \mathbf{b}_m)$.

1. Using the secure split protocol (Protocol 1), each party P_j $(1\le j \le m)$ gets $\mathbf{B}_j$ and $\mathbf{c}_j$ such that $\sum_{j=1}^{m} \mathbf{A}_j = \sum_{j=1}^{m} \mathbf{B}_j$ and $\sum_{j=1}^{m} \mathbf{b}_j = \sum_{j=1}^{m} \mathbf{c}_j$.
2. All parties jointly choose two invertible random $n{\times}n$ matrices $\mathbf{T}$ and $\mathbf{S}$.
3. Each party P_j $(1\le j \le m)$ sends $\hat{\mathbf{E}}_j = \mathbf{T}\,\mathbf{B}_j\mathbf{S}$ and $\hat{\mathbf{g}}_j = \mathbf{T}\,\mathbf{c}_j$ to the Commodity-Server.
4. The Commodity-Server receives $\hat{\mathbf{E}}_j$ and $\hat{\mathbf{g}}_j$ from each party and solves the linear equations $(\sum_{j=1}^{m} \hat{\mathbf{E}}_j)\,\hat{\mathbf{y}} = \sum_{j=1}^{m} \hat{\mathbf{g}}_j$. The Commodity-Server sends $\hat{\mathbf{y}}$ to all parties.
5. Each party P_j $(1\le j \le m)$ obtains the solution $\mathbf{x}$ by computing $\mathbf{x} = \mathbf{S}\,\hat{\mathbf{y}}$.

We can confirm that the value $\mathbf{x}$ of step 5 of Protocol 2 is the solution of $(\mathbf{A}_1 + \dots + \mathbf{A}_m)\,\mathbf{x} = (\mathbf{b}_1 + \dots + \mathbf{b}_m)$, since $\sum_{j=1}^{m} \hat{\mathbf{E}}_j = \mathbf{T}\,(\sum_{j=1}^{m} \mathbf{B}_j)\,\mathbf{S} = \mathbf{T}\,(\sum_{j=1}^{m} \mathbf{A}_j)\,\mathbf{S}$, $\sum_{j=1}^{m} \hat{\mathbf{g}}_j = \mathbf{T}\,(\sum_{j=1}^{m} \mathbf{c}_j) = \mathbf{T}\,(\sum_{j=1}^{m} \mathbf{b}_j)$.

Now we consider the PCMC-LLS problem of Definition 2. Note that the linear system $\mathbf{Ax} = \mathbf{b}$, where $\mathbf{A}$ and $\mathbf{b}$ are the $l \times n$ $(l > n)$ and $l{\times}1$ matrices, is equivalent to the normal linear system $\mathbf{A}^{\mathrm{t}}\mathbf{Ax} = \mathbf{A}^{\mathrm{t}}\mathbf{b}$, where $\mathbf{A}^{\mathrm{t}}$ denotes the transposed matrix of $\mathbf{A}$. Since $\mathbf{A}^{\mathrm{t}}\mathbf{A}$ is an $n \times n$ square matrix and $\mathbf{A}^{\mathrm{t}}\,\mathbf{b}$ is an n-dimensional column vector, the linear least-squares problem can be expressed in the problem of solving the corresponding linear system of n equations. On the basis of this mathematical property, we can design a protocol which solves the PCMC-LLS problem.

Protocol 3. (PCMC-LLS Protocol)

Input: Each party P_j $(1\le j \le m)$ provides a matrix $\mathbf{A}_j$ and a column vector $\mathbf{b}_j$ as its private inputs, where $\mathbf{A}_j$ and $\mathbf{b}_j$ $(1\le j \le m)$ are the $l{\times}n$ $(l > n)$ and $l{\times}1$ matrices whose entries are all zero except for the k_j $(k_1+ \dots + k_m = l)$ rows, respectively.

Output: Each party P_j $(1\le j \le m)$ obtains the linear least-squares solution $\mathbf{x}$ of $(\mathbf{A}_1 + \dots + \mathbf{A}_m)\,\mathbf{x} = (\mathbf{b}_1 + \dots + \mathbf{b}_m)$.

1. Using the secure split protocol (Protocol 1), each party P_j $(1\le j \le m)$ gets $\mathbf{B}_j$ and $\mathbf{c}_j$ such that $\sum_{j=1}^{m} \mathbf{A}_j = \sum_{j=1}^{m} \mathbf{B}_j$ and $\sum_{j=1}^{m} \mathbf{b}_j = \sum_{j=1}^{m} \mathbf{c}_j$.
2. All parties jointly choose two invertible random $n{\times}n$ matrices $\mathbf{T}$ and $\mathbf{S}$, and share $\mathbf{B}_1, \dots, \mathbf{B}_m$ and $\mathbf{c}_1, \dots, \mathbf{c}_m$.
3. Each party P_j $(1\le j \le m)$ sends $\hat{\mathbf{E}}_j = \mathbf{T}(\,\mathbf{B}_j^{\mathrm{t}}\mathbf{B}_1 + \dots + \mathbf{B}_j^{\mathrm{t}}\mathbf{B}_m)\mathbf{S}$ and $\hat{\mathbf{g}}_j = \mathbf{T}\,(\,\mathbf{B}_j^{\mathrm{t}}\mathbf{c}_1 + \dots + \mathbf{B}_j^{\mathrm{t}}\mathbf{c}_m)$ to the Commodity-Server.
4. The Commodity-Server receives $\hat{\mathbf{E}}_j$ and $\hat{\mathbf{g}}_j$ from each party and solves the linear equations $(\sum_{j=1}^{m} \hat{\mathbf{E}}_j\,)\,\hat{\mathbf{y}} = \sum_{j=1}^{m} \hat{\mathbf{g}}_j$. The Commodity-Server sends $\hat{\mathbf{y}}$ to all parties.
5. Each party P_j $(1\le j \le m)$ obtains the solution $\mathbf{x}$ by computing $\mathbf{x} = \mathbf{S}\,\hat{\mathbf{y}}$.

At step 4 of Protocol 3, the Commodity-Server solves the following linear system of equations:

$$\{\mathbf{T}\,(\,\textstyle\sum_{j=1}^{m}\sum_{i=1}^{m} \mathbf{B}_j^{\mathrm{t}}\mathbf{B}_i)\,\mathbf{S}\}\,\mathbf{x} = \mathbf{T}\,(\,\textstyle\sum_{j=1}^{m}\sum_{i=1}^{m} \mathbf{B}_j^{\mathrm{t}}\mathbf{c}_i)\,.$$

Hence we know that the linear least-squares solution of $(\mathbf{A}_1 + \dots + \mathbf{A}_m)\,\mathbf{x} = (\mathbf{b}_1 + \dots + \mathbf{b}_m)$ is $\mathbf{S}\,\hat{\mathbf{y}}$.

5. Analysis of Security and Complexity

In Protocol 1, the method of hiding data is secure in practice, but not in the information-theoretic sense. A single private input $\mathbf{A}_j$ is distributed among t+1 parties. When this protocol is used as part of another protocol, the individual shares $\boldsymbol{B}_j$'s are revealed. However, in order for an individual $\boldsymbol{A}_j$ to be revealed, all of the t parties to whom party P_j sends messages in step 3 of Protocol 1 and all parties who send messages to party P_j in step 4 must collude. Therefore, Protocol 1 is collusion-resistant against a threshold adversarial structure that consists of at most t colluders. To clarify this property, we introduce the t-wise collusion-resistance.

Definition 3. (t-wise collusion-resistance) Let $t, i, j \in \{1, \ldots, m\}$. A multi-party computation protocol guarantees t-wise collusion-resistance if for any set $C \subset \{ P_1, , \ldots , P_m \}$, where $|C| \leq t$, and all $c \in C$ can collude together, the members in C cannot obtain any information about the private inputs of the members in $\{ P_1, , \ldots , P_m \}-C$.

It is easy to show that Protocol 1 guarantees t-wise collusion-resistance. Moreover, we can obtain the following theorem based on the acceptable security of [10].

Theorem 1. Assume that all entries of the matrices are generated from the real number domain. Then Protocol 1 is secure such that each P_j does not learn any entry of $\boldsymbol{A}_i$, a private input of P_i, for any $i \neq j$.

Proof. Note that Protocol 1 guarantees t-wise collusion-resistance. Fix two parties $P_j, P_i \in \{ P_1, , \ldots , P_m \}$, such that P_j and P_i cannot collude. Then $\boldsymbol{R}_{k_i}{}^{(i)}$ or nothing is all that P_j can obtain from P_i, and $\boldsymbol{R}_{k_i}{}^{(i)} = \boldsymbol{A}_i - \sum_{k=0}^{t} \boldsymbol{R}_k{}^{(i)} (k \neq k_i)$. Therefore, P_j cannot find out any entry of $\boldsymbol{A}_i$, because of the randomness and the secrecy of $\boldsymbol{R}_0{}^{(i)}$.

If the entries of the random matrices are not generated from the real number domain, P_j might obtain some information about $\boldsymbol{A}_i$. For example, if the entries of the random matrices are in the domain of some finite interval, then P_j can obtain some properties about the entries of $\boldsymbol{A}_i$.

In Protocols 2 and 3, everything that the Commodity-Server learns during the execution of protocols are $\hat{\boldsymbol{E}}_j$'s and $\hat{\boldsymbol{g}}_j$'s. Thus the Commodity-Server does not learn anything about the real solution $\boldsymbol{x}$, since the server cannot collude with any party and does not know the value of $\boldsymbol{S}$. Protocol 1 is used as a part of Protocols 2 and 3, so the acceptable security of Protocols 2 and 3 is guaranteed by that of Protocol 1.

Theorem 2. Assume that all entries of the matrices are generated from the real number domain. Then Protocols 2 and 3 are secure such that each P_j does not learn any entry of $\boldsymbol{A}_i$ and $\boldsymbol{b}_i$, the private inputs of P_i, for any $i \neq j$.

Protocol 1 is performed in two rounds, its total communication is $\boldsymbol{O}(tm)$, and its computational complexity at each party is $\boldsymbol{O}(t)$. Protocol 2 is carried out in four rounds including the secure split protocol. The total communication of Protocol 2 is $\boldsymbol{O}(tm)$, and the computational complexity at each party of that is $\boldsymbol{O}(t)$, where t is the security parameter for the split protocol. Protocol 3 is performed in five rounds, where its total communication is $\boldsymbol{O}(tm)$, and its computational complexity at each party is $\boldsymbol{O}(m)$.

Meanwhile, the 2-party protocol of [4] can be extended to the m-party ($m \geq 3$) protocol by performing $m(m-1)/2$ times 2-party protocol. Although achieving the ideal security, the 2-party protocol of [4] is not practical, since the oblivious transfer protocol of [11] is used as an important part of that protocol. The communication complexity of the oblivious transfer protocol is $\boldsymbol{O}(l)$, where l is the length of a number that is hard to factor. Moreover, in the 2-party protocol of [4] another security parameter μ such that 2^{μ} is computationally infeasible is needed, so the communication complexity of the extended m-party version of [4] is $\boldsymbol{O}(m^2 l \mu)$. Therefore, the cost of communication $\boldsymbol{O}(tm)$ of our protocols is significantly better than $\boldsymbol{O}(m^2 l \mu)$. Table 1 summarizes the comparison of communication complexity between the extended m-party protocol version of [4] and our protocols.

Table 1. Comparison of communication complexity

SMC protocol	Communication complexity	Parameters
m-party protocol version of [4]	$\boldsymbol{O}(m^2 l \mu)$	- l : length of a number that is hard to factor - μ : the number such that 2^{μ} is computationally infeasible
PCMC-LSE PCMC-LLS	$\boldsymbol{O}(tm)$	- t : the security parameter such that $1 \leq t \leq m-1$.

6. Conclusion

We have proposed three practical SMC protocols for privacy-preserving cooperative scientific computations. We have designed a secure split protocol (Protocol 1), and PCMC-LSE (Protocol 2) and PCMC-LLS (Protocol 3) protocols in order to solve the problems of privacy-preserving cooperative multi-party computation for the linear system of equations and the linear least-squares. The protocols proposed in this paper achieve acceptable security in the sense of Du-Zhan's paradigm and t-wise collusion-resistance, and their communication complexity is $\boldsymbol{O}(tm)$, where t **is** a security parameter and m is the total number of participants. This complexity represents a significant improvement on the previous result.

References

[1] A. C. Yao, "Protocols for secure computations", At the 23rd Annual Symposium on the Foundations of

Computer Science, *IEEE*, 1982.

[2] O. Goldreich, S. Micali, A. Wigderson, “How to play any mental game”, In the Proceedings of the 19th Annual ACM Symposium on the Theory of Computing, pp. 218-229, 1987.

[3] O. Goldreich, “Secure Multi-party Computation”, Final Draft, Version 1.4, 2002.

[4] W. Du, M. J. Atallah, “Privacy-preserving Cooperative Scientific Computations”, In the 14th IEEE Computer Security Foundations Workshop, pp. 273-282, 2001.

[5] W. Du, M. J. Atallah, “Protocols for secure remote database access with approximate matching”, In the 7th ACM Conference on Computer and Communications Security, 2000.

[6] W. Du, M. J. Atallah, “Secure multi-party computation problems and their applications: A review and open problems”, In the Proceedings of the New Security Paradigms Workshop, pp. 11-20, 2001.

[7] W. Du, M. J. Atallah, “Privacy-preserving statistical analysis”, In the Proceedings of the 17th Annual Computer Security Applications Conference, pp. 102-110, 2001.

[8] M. J. Atallah, W. Du, “Secure multi-party computational geometry”, In WADS2001: 7th International Workshop on Algorithms and Data Structures, pp. 165-179, 2001.

[9] Y. Lindell, B. Pinkas, “Privacy preserving data mining”, Advances in Cryptology – Crypto2000, LNCS 1592, pp. 402-414, 1999.

[10] W. Du, Z. Zhan, “A practical approach to solving secure multi-party computation problems”, In New Security Paradigms Workshop 2002, pp. 127-135, 2002.

[11] M. Naor, B. Pinkas, “Oblivious transfer and polynomial evaluation”, In the Proceedings of the 31st ACM Symposium on the Theory of Computing, pp. 245-254, 1999.

[12] M. Atallah, M. Bykova, J. Li, K. Frikken, M. Topkara, “Private collaborative forecasting and benchmarking”, WEPS2004, 2004.

Ju-Sung Kang
Ju-Sung Kang received B.S., M.S., and Ph.D. degrees in Mathematics from Korea University, Seoul, Korea in 1989, 1991 and 1996, respectively. From 1997 to 2004, he was a member of the technical staff at ETRI. He has been an associate professor at Kookmin University since 2004. His current research interests include cryptographic algorithms and protocols.

Dowon Hong
received B.S., M.S., and Ph.D. degrees in Mathematics from Korea University, Seoul, Korea in 1994, 1996 and 2000, respectively. He joined ETRI in 2000, and is currently with Information Security Division. His current research interests include cryptographic algorithms and protocols.