2008 International Conference on Information Security and Assurance

Towards Empirical Aspects of Secure Scalar Product

I-Cheng Wang Chih-Hao Shen* Tsan-Sheng Hsu Churn-Chung Liao Da-Wei Wang
Institute of Information Science, Academia Sinica, Taiwan†
{icw, shench, tshsu, liaucj, wdw}@iis.sinica.edu.tw

Justin Zhan
CyLab Japan, Carnegie Mellon University, USA
justinzh@andrew.cmu.edu

Abstract

Privacy is ultimately important, and there is a fair amount of research about it. However, few empirical studies about the cost of privacy are conducted. In the area of secure multiparty computation, the scalar product has long been reckoned as one of the most promising building blocks in place of the classic logic gates. The reason is not only the scalar product complete, which is as good as logic gates, but also the scalar product is much more efficient than logic gates. As a result, we set to study the computation and communication resources needed for some of the most well-known and frequently referred secure scalar-product protocols, including the composite-residuosity, the invertible-matrix, the polynomial-sharing, and the commodity-based approaches. Besides the implementation remarks of these approaches, we analyze and compare their execution time, computation time, and random number consumption, which are the most concerned resources when talking about secure protocols. Moreover, Fairplay, the benchmark approach implementing Yao's famous circuit evaluation protocol, is included in our experiments in order to demonstrate the potential for the scalar product to replace logic gates.

1 Introduction

Secure multiparty computation is a research topic aiming at the double-edged privacy problem. The core issue of the topic is how several potentially distrustful parties can cooperate to conduct certain computations over their private data without compromising their privacy. After Yao's [20] general solution to two-party secure computation was proposed, Goldreich et al. [12] gave another general solution to multiparty computation. Both proposals are so elegant that they provide secure two-party/multiparty protocols for binary AND and XOR gates, which can be further generalized to all functions. However, despite their academic significance, the computation costs of both solutions are too prohibitive to be practical.

A function f is *complete* if a secure protocol for f implies the existence of secure protocols for all functions. Yao and Goldreich et al. propose the idea that solving the secure multiparty computation by giving secure protocols for complete functions. Extended from the idea, the scalar product has gathered more and more attention because of its completeness and integer-based computation power.

Proposals for the secure scalar product are blooming. Du et al. [9] propose the invertible-matrix and the commodity-based approaches. The former approach enables the trade-off between efficiency and privacy, and the latter is based on Beaver's commodity model [2]. Goethals et al. [10] propose the computationally secure scalar-product protocol, security of which depends on the intractability of the composite residuosity class problem. The information-theoretically secure polynomial-sharing approach proposed by Ben-Or et al.[3] is also one of the most frequently referred solutions. Furthermore, Fairplay [15], the benchmark approach implementing Yao's general solution, is included in our experiments in order to demonstrate the potential for scalar products to replace logic gates.

Despite the mass of secure scalar-product protocols, few empirical studies are conducted. Numerous approaches propose the feasibility of secure scalar product, but to construct a system for secure multiparty computation, the execution and computation time matters very much. The typical issues that we may have are as follows: (1) What is the ex-

*This work was conducted while the author visited CyLab Japan.

†This work is supported by Taiwan National Science Council (grant No. NSC96-2221-E-001-014, NSC96-2221-E-001-004, and NSC95-2221-E-001-029-MY3), Taiwan Information Security Center (TWISC), and International Collaboration for Advancing Security Technology (iCAST).

978-0-7695-3126-7 2008

DOI 10.1109/ISA.2008.78

act execution time for high-dimension inputs? How long does it take to compute a 100-, 1000-, or 10000-dimension scalar product? (2) If the approach is computationally secure, what is its throughput under typical security parameters? (3) If the approach is theoretically secure, there must be a third party. What is the workload of the third party? (4) Last but not least, what is the performance of a specific approach compared with Fairplay's?

In this work, we implement four secure scalar-product protocols, including the invertible-matrix, the composite-residuosity, the polynomial-sharing, and the commodity-based approaches. We also give specific evaluation to address the above issues. Based on our results, protocol designers can estimate their performance of their scalar-product-based designs more precisely.

This paper is organized as the following: Section 2 discusses the related works. Section 3 briefly introduces the mentioned approaches realizing secure scalar-product computation. Section 4 then gives implementation remark and analyzes the resources needed for each approach, including computation cost and random number consumptions. After that, Section 5 shows the specific data about the total execution time and CPU time of various approaches. Finally, Section 6 concludes the paper and lays out future work.

2 Related Works

After Yao's [20] and Goldreich et al.'s [12] proposals, researchers have been looking for new complete functions and the foundations of the completeness. Kilian shows that the oblivious transfer is complete [13], and so are the functions with imbedded OR [14]. Furthermore, it is believed that the integer-based scalar product is more practical to real applications than the binary-based oblivious transfer is [6].

There are many efforts on building various applications on secure scalar products. Atallah et al. reduce geometry problems to scalar products [1]. Du et al. construct secure protocols for statistical analysis [8] and scientific computations [7]. More recently, Bunn et al. [4] give a secure protocol for k-means clustering based on scalar products under the composite-residuosity approach.

There is also plenty of theoretical studies on scalar products. Based on information theory, Chiang et al. [5] propose a privacy measurement, by which they analyze various approaches. They prove that the invertible-matrix approach discloses at least half the information and the commodity-based approach is perfectly secure. Wang et al. [19] prove that there is no information-theoretically secure two-party protocol for the scalar product. Moreover, the closure property of the commodity-based approach is verified according to the security definition based on information theory [17]. More specifically, it is proven that a protocol sequentially composed of the commodity-based approach is also secure.

3 Various Proposals for Scalar-product

We begin the section with the formulation of the scalar product and the notations used hereafter. Alice has input $X_A = [x_{A_1}, \ldots, x_{A_n}]$ and Bob has $X_B = [x_{B_1}, \ldots, x_{B_n}]$. Each element of the private input vectors is in the domain $\mathcal{D}$. They want to collaboratively compute the scalar product of their private vectors and then share the result. In other words, Alice computes y_A and Bob computes y_B such that

$$y_A + y_B = X_A * X_B^T = \sum_{i=1}^{n} x_{A_i} \cdot x_{B_i}$$

where $y_A, y_B \in \mathcal{D}$, and $+$ and $\cdot$ are defined in accordance with $\mathcal{D}$. Our formulation follows Goldreich et al.'s principle [11] that the intermediate results during a protocol execution are always shared. In a protocol π composed of scalar products, outputs of the current scalar product can be inputs to the next scalar product, which is actually the intermediate results of π. Therefore, in order to imitate Goldreich et al's principle we formulate the outputs to be shared among participants.

Next, we briefly introduce aforementioned approaches.

3.1 Invertible-matrix Approach

1. Alice and Bob agree on an invertible $n \times n$ matrix M and a constant k, where k is less than n.
2. Alice computes $X'_A = X_A * M^{-1} = [x_{A'_1}, \ldots, x_{A'_n}]$ and sends $[x_{A'_1}, \ldots, x_{A'_k}]$ to Bob.
3. Bob computes $X'_B = M * X_B = [x_{B'_1}, \ldots, x_{B'_n}]$ and sends $[x_{B'_{k+1}}, \ldots, x_{B'_n}]$ to Alice.
4. Alice computes $y_A = \sum_{i=k+1}^{n} x_{A_i} \cdot x_{B'_i}$ as her output.
5. Bob computes $y_B = \sum_{i=1}^{k} x_{A'_i} \cdot x_{B_i}$ as his output.

3.2 Commodity-based Approach

1. Alice and Bob agree on a semi-trusted commodity server Carol who selects two $1 \times n$ random vectors R_A and R_B uniformly from $\mathcal{D}^n$, and selects a random number r_A uniformly from $\mathcal{D}$.
2. After computing $r_B = R_A * R_B^T - r_A$, Carol sends (R_A, r_A) and (R_B, r_B) to Alice and Bob respectively.
3. Alice computes $X'_A = X_A + R_A$ and sends it to Bob.
4. Bob computes $X'_B = X_B + R_B$ and sends it to Alice.
5. Bob selects a random number y_B uniformly from $\mathcal{D}$. After that, he computes $t = X'_A * X_B^T + r_B - y_B$ and sends t to Alice.
6. Alice computes $y_A = t - R_A * X_B'^T + r_A$.

3.3 Polynomial-sharing Approach

1. Alice, Bob, and Carol agree on three distinct numbers $\alpha_A, \alpha_B, \alpha_C \in \mathcal{D}$.
2. Alice selects r_{A_i} uniformly and randomly; evaluates the polynomial $f_{x_{A_i}}(t) = r_{x_{A_i}} \cdot t + x_{A_i}$ at α_A, α_B, and α_C; and sends $f_{x_{A_i}}(\alpha_B)$ and $f_{x_{A_i}}(\alpha_C)$ to Bob and Carol respectively, for $i = 1, \ldots, n$.
3. Bob selects r_{B_i} uniformly and randomly; evaluates the polynomial $f_{x_{B_i}}(t) = r_{x_{B_i}} \cdot t + x_{B_i}$ at α_A, α_B, and α_C; and sends $f_{x_{B_i}}(\alpha_A)$ and $f_{x_{B_i}}(\alpha_C)$ to Alice and Carol respectively, for $i = 1, \ldots, n$.
4. Bob selects r_{y_B} and y_B uniformly and randomly; evaluates the polynomial $f_{y_B}(t) = r_{y_B} \cdot t + y_B$ at α_A, α_B, and α_C; and sends $f_{y_B}(\alpha_A)$ and $f_{y_B}(\alpha_C)$ to Alice and Carol respectively.
5. Alice, Bob, and Carol compute the scalar product locally with their shares of x_{A_i}, x_{B_i}, and y_B. Then all the results are sent to Alice. In other words, Alice will have s_A, s_B, and s_C, such that

$$\begin{aligned} s_A =& \textstyle\sum_{i=1}^{n} f_{x_{A_i}}(\alpha_A) f_{x_{B_i}}(\alpha_A) - f_{y_B}(\alpha_A) \\ s_B =& \textstyle\sum_{i=1}^{n} f_{x_{A_i}}(\alpha_B) f_{x_{B_i}}(\alpha_B) - f_{y_B}(\alpha_B) \\ s_C =& \textstyle\sum_{i=1}^{n} f_{x_{A_i}}(\alpha_C) f_{x_{B_i}}(\alpha_C) - f_{y_B}(\alpha_C) \end{aligned}$$

6. It is clear that s_A, s_B, and s_C are evaluations of a 2-degree polynomial at α_A, α_B, and α_C respectively. Furthermore, the polynomial's constant coefficient is $y_A = \sum_{i=1}^{n} x_{A_i} x_{B_i} - y_B$. Thus, after collecting s_A, s_B, and s_C, Alice can reconstruct y_A by interpolation:

$$\begin{aligned} y_A =& s_A \cdot \alpha_B \cdot \alpha_C \cdot (\alpha_A - \alpha_B)^{-1} \cdot (\alpha_A - \alpha_C)^{-1} \\ &+ \alpha_A \cdot s_B \cdot \alpha_C \cdot (\alpha_B - \alpha_A)^{-1} \cdot (\alpha_B - \alpha_C)^{-1} \\ &+ \alpha_A \cdot \alpha_B \cdot s_C \cdot (\alpha_C - \alpha_A)^{-1} \cdot (\alpha_C - \alpha_B)^{-1} \end{aligned}$$

3.4 Composite-residuosity Approach

KEY GENERATION

1. Alice generates two large primes p and q. She then computes $m = p \cdot q$ and $\lambda = \text{lcm}(p-1, q-1)$.
2. Alice randomly selects $g \in \mathbb{Z}^*_{m^2}$ such that

$$\gcd(L(g^\lambda \bmod m^2), m) = 1,$$

   where $L(u) = \frac{u-1}{m}$, $\forall u \in \{u < m^2 | m = 1 \bmod m\}$.[1]
3. Alice publicizes (m, g) as the public key and keeps (p, q, λ) secret as the private key.

[1] It is proven that $g^\lambda = 1 \bmod m$, for any $g \in \mathbb{Z}^*_{m^2}$.

PROTOCOL

1. Alice randomly selects $r_i \in \mathbb{Z}^*_{m^2}$ and computes

$$c_{A_i} = g^{x_{A_i}} \cdot r_i^m \bmod m^2, \text{for } i = 1, \ldots, n.$$

   Then Alice sends $[c_{A_1}, \ldots, c_{A_n}]$ to Bob.
2. Bob randomly selects his output $y_B (0 \leq y_B < m)$ and computes $c_B = g^{m - y_B} \bmod m^2$.
3. Bob computes the scalar product in encrypted domain by $t = \left(\prod_{i=1}^{n} c_{A_i}^{x_{B_i}}\right) \cdot c_B \bmod m^2$ and sends t to Alice.
4. Alice computes her output y_A by

$$y_A = L(t^\lambda \bmod m^2) \cdot L(g^\lambda \bmod m^2)^{-1} \bmod n.$$

4 Analysis and Implementation Remark

Secure protocols are usually designed in an ideal case or under assumptions. To implement a protocol, the situation is no longer ideal, and neither are the assumptions easily satisfied. In this section, we list potential obstacles during the implementation of secure scalar-product protocols and propose feasible solutions.

4.1 Invertible-matrix Approach

According to Chiang et al.'s result [5], the invertible-matrix approach always discloses half information of the inputs. In other words, to execute this approach more than once with the same private input and different invertible matrices will expose all information of inputs. Therefore, it is recommended to use the same invertible matrix all the time.

The invertible-matrix approach provides tradeoff between privacy and efficiency, and the tradeoff is controlled by the invertible matrix M. When M is an identity matrix, the computation and space complexity is $O(n)$, and the protocol is equivalent to that both Alice and Bob reveal half of their private inputs. When M is an ordinary non-singular matrix, the algorithm for its inversion is $O(n^3)$. Even though the invertible matrices of various dimensions can be precomputed so that the computation is the multiplication between a matrix and a vector, which is of complexity $O(n^2)$, the space complexity is $O(n^2)$ as well so that 2GB memory will be exhausted when the elements are 4-byte integers and the vector dimension $n = 2^{15} (\approx 32,000)$. Therefore, instead of storing the entire matrix (n^2 elements) in memory, we recommend exploiting the matrix of special format, the elements of which can be computed from fewer attributes than n^2. For example, the element of an invertible Vandermonde matrix v_{ij} can be computed from n distinct numbers $a_1, \ldots, a_n$, by $v_{ij} = a_i^{j-1}, \forall i, j = 1, \ldots, n$.

When M is an invertible Vandermonde matrix, there is a $O(n^2)$ algorithm [18] computing $M^{-1} = U^{-1} * L^{-1}$. More specifically, if M's elements $v_{ij} = a_i^{j-1}, \forall i, j = 1, \ldots, n$, the elements l_{ij} of L^{-1} are given by the relations

$$l_{ij} = \begin{cases} 0 & i < j \\ 1 & i = j = 1 \\ \prod_{k=1, k \neq j}^{i} (a_j - a_k)^{-1} & \text{otherwise,} \end{cases}$$

and the elements u_{ij} of U^{-1} are given by

$$u_{ij} = \begin{cases} 0 & j = 1 \\ 1 & i = j \\ u_{i-1,j-1} - a_{j-1} u_{i,j-1} & \text{otherwise.} \end{cases}$$

After combining the above formulae with Horner's rule of polynomial evaluation, we have a $O(n^2)$ solution. Thus, without increasing computation complexity, the invertible Vandermonde matrix reduces the space complexity to $O(n)$.

4.2 Commodity-based Approach

A commodity server is introduced in the commodity-based scalar-product protocol. The commodity server is to the commodity-based scalar product as the certificate authority is to secure communication. The certificate authority does not participate in secure communication, but its issued digital certificate binds the public key and the key owner. Similarly, the commodity server does not participate in Alice and Bob's communication, and neither does it receive any data from both parties. The commodity server merely delivers coordinated random numbers.

However, in order to relief loads from unstable network, we recommend transferring the seed of pseudorandom number generator rather than the random numbers themselves. Although the computation loads to both Alice and Bob increase, the advantage is the constant communication between the commodity server and both of the other parties. Thus, including the random numbers r_A and y_B, pseudorandom number generator is called $4n + 2 = O(n)$ times, and the computation and communication complexity is $O(n)$.

4.3 Polynomial-sharing Approach

The polynomial-sharing approach proposed by Ben-Or et al. [3] includes addition, constant multiplication, and multiplication in integer level. Because the secret is disguised as the constant coefficient of a polynomial, to reduce the polynomial degree is important before the degree is over $\lceil \frac{t}{2} \rceil$, where t is the number of participants. The degree reduction not only is the most expensive operation but also requires synchronization. Fortunately, in our case the two-party scalar product needs no degree reduction. Without synchronization, soon after Alice and Bob share their private inputs, all three parties can locally compute their share of result. Therefore, it is recommended that to communicate in a batch in order to reduce socket connections. More specifically, Alice can send all Bob's shares of $x_{A_i}, i = 1, \ldots, n$ as a whole packet rather than one share per packet. The random number consumption, computation, and communication is of complexity $O(n)$ in this approach.

4.4 Composite-residuosity Approach

The security of composite-residuosity approach is based on the composite residuosity class problem [16], which is proven to be reducible to the RSA and the factoring large number problems. It is commonly believed that the large number should be at least 1024-bit long to be computationally intractable. Therefore, the bit length of the composite m should be more than 1024.

Let $s = (1 - \frac{1}{p})(1 - \frac{1}{q})$, then expectation number for Alice to randomly select $r_i \in \mathbb{Z}^*_{m^2}$ is $E(R)$, where

$$E(R) = \sum_{i=0}^{\infty} s(1-s)^i \cdot (i+1) \quad (1)$$

$$(1-s)E(R) = \sum_{i=0}^{\infty} s(1-s)^i \cdot i \quad (2)$$

After (1)-(2), we have

$$sE(R) = \sum_{i=0}^{\infty} s(1-s)^i$$

$$\implies E(R) = \sum_{i=0}^{\infty} (1-s)^i = \frac{1}{s} \approx 1$$

because p and q are large primes ($\approx 2^{512}$) so that $s \approx 1$. Therefore, Alice calls the random number genertor n times expectedly for $r_1, \ldots, r_n$; and Bob calls once for y_B. The computation and communication is also in the order of $O(n)$ in this approach.

4.5 Fairplay

Fairplay system includes a compiler that automatically converts an SFDL[2] program into a secure, constant-round, two-party protocol proposed by Yao [20]. More specifically, Bob generates a garbled circuit, and then Alice evaluates it according to Yao's circuit evaluation protocol, the security of which depends on the underlying cryptosystem. In the case of Fairplay, SHA-1 hash function is adopted to encryption and decryption. However, because of the implementation limit, Fairplay cannot generate a garbled circuit that correctly handles integers of more than 32 bits, and neither can it compile a scalar-product program with input vectors of more than 12 dimensions on a 2GB RAM personal computer. An "Out Of Memory Exception" is always thrown.

[2]Secure Function Definition Language is a programming language defined in Fairplay system. The language is for end users to program their applications with high-level descriptions.

5 Experimental Results

In this section, we implement all the aforementioned approaches according to the instructions in Section 3. In the implementation, we have two machines, the specification of which are AMD Opteron 2.2 GHz with 4GB DDR2 RAM and Intel Xeon 3.0 with 4GB DDR2 RAM. To minimize the probabilistic variation and to balance the different computing power, our experimental results are the average of 50 effective executions. An effective execution is the average of two protocol execution, each of which has Alice's code run on different machines and Bob's code run on the other. Our experiments focus on different resources for secure multiparty computation: random number consumption, execution time, and CPU time. It is reminded that the execution time and the CPU time are the aggregation results by summing up Alice's and Bob's execution and CPU time.

Limited by Fairplay's 32-bit precision, we define $\mathcal{D}$ as the finite field $GF(2^{31}-1)$ for the invertible-matrix, the commodity-based, and the polynomial-sharing approaches. For the composite-residuosity approach, we define $\mathcal{D}$ as the ring $\mathbb{Z}_m$, where $m = (2^{521}-1)(2^{607}-1)$.[3] Moreover, the experiment inputs to five approaches are all randomly selected from $GF(2^{31}-1)$, and the possible overflow is ignored since we care about execution time solely.

5.1 Random Number Consumption

We compare the random number consumption between Fairplay and the aforementioned approaches. Recalled that in Fairplay the scalar product is compiled into a garbled circuit composed of gates and wires, each of which needs two 80-bit random strings representing 0 and 1 value. Figure 1 shows the random bits used by different approaches.

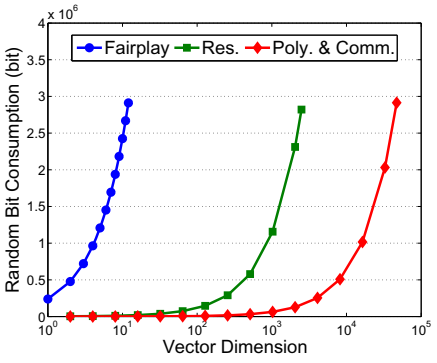

Figure 1. Random number consumption.

[3]For simplicity purpose, we use Mersenne primes $2^{31}-1$, $2^{521}-1$, and $2^{607}-1$ in our experiments.

5.2 Different Invertible Matrices

Figure 2 shows that the performance difference between protocols with Vandermonde matrices and normal nonsingular ones is less than 1 order of magnitude, but the latter cannot be applied to scalar products of more dimension than 2^{15} on our machines. The line labeled "identity" is the result that Alice and Bob simply exchange half of their inputs.

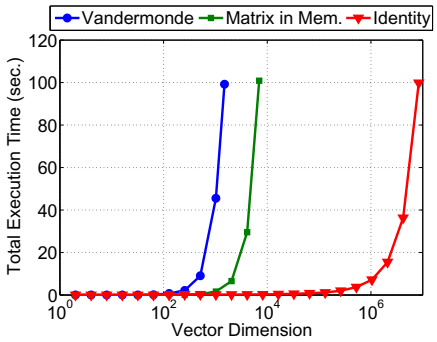

Figure 2. The total execution time of the invertible-matrix approach with different invertible matrices.

5.3 Comparison Among Approaches

Figure 3 and Figure 4 show the performance between various approaches. The invertible-matrix in the figures are the one with the invertible Vandermonde matrix. It is reminded that the execution time includes CPU time, waiting time, and communication time. From these two figures we know that the commodity-based approach has the best performance and the smallest CPU and execution time ratio.

6 Conclusion and Future Works

In this paper, we investigate several well-known and frequently referred secure scalar-product protocols, including the composite-residuosity, the invertible-matrix, the polynomial-sharing, and the commodity-base approaches. We describe the instructions to implement these approaches in detail, analyze the computation cost and random number consumption, list implementation remark, and compare their performance with the benchmark Fairplay system.

According to our experimental results, it is evident that the scalar product has great potential replacing the logic gate as a building block in two-party computation. On one hand, scalar-product protocols consume less random numbers than logic-gate protocols do. On the other hand, the

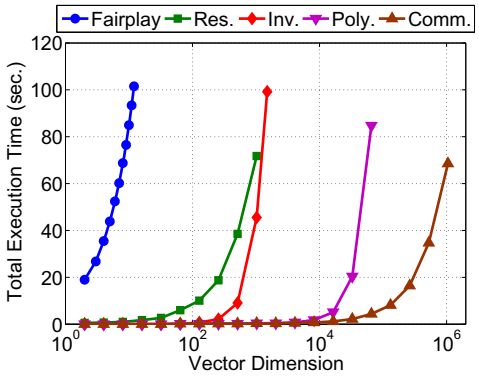

Figure 3. Comparison of the execution time.

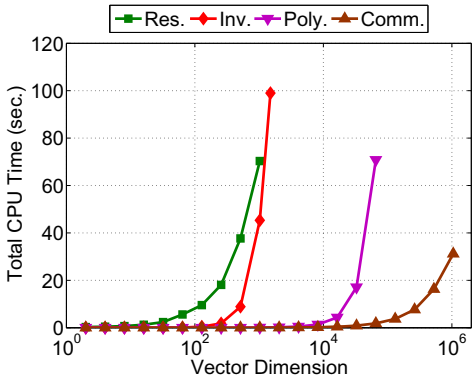

Figure 4. Comparison of the CPU time.

former has throughput at least three orders of magnitude higher than the latter's.

It is evident that scalar product outperforms logic gates in two-party computation. However, the comparison between the number-product[4] and Goldreich et al.'s approach in multiparty computation is unknown. Moreover, we are also interested in the performance of these secure scalar-product protocols when the number of participants increases, and more implementation remark will be explored in the future.

References

[1] M. J. Atallah and W. Du. Secure multi-party computational geometry. *LNCS*, 2125:165–179, 2000.

[2] D. Beaver. Commodity-based cryptography (extended abstract). In *STOC '97*, pages 446–455, New York, NY, USA, 1997. ACM Press.

[3] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *STOC '88*, pages 1–10, New York, NY, USA, 1988. ACM Press.

[4] P. Bunn and R. Ostrovsky. Secure two-party k-means clustering. In *CCS '07: Proceedings of the 14th ACM conference on Computer and communications security*, pages 486–497, New York, NY, USA, 2007. ACM.

[5] Y.-T. Chiang, D.-W. Wang, C.-J. Liau, and T.-S. Hsu. Secrecy of two-party secure computation. *Lecture Notes in Computer Science*, 3654:114–123, 2005.

[6] W. Du. *A Study of Several Specific Secure Two-party Computation Problems*. PhD thesis, Purdue Univ., August 2001.

[7] W. Du and M. J. Atallah. Privacy-preserving cooperative scientific computations. In *CSFW '01*, page 273, Washington, DC, USA, 2001. IEEE Computer Society.

[8] W. Du and M. J. Atallah. Privacy-preserving cooperative statistical analysis. In *ACSAC '01*, pages 102–110, Washington, DC, USA, 2001. IEEE Comp. Soci.

[4]The number-product is an extension of the two-party scalar product in multiparty computation. [17]

[9] W. Du and Z. Zhan. A practical approach to solve secure multi-party computation problems. In *NSPW '02*, pages 127–135, New York, NY, USA, 2002. ACM Press.

[10] B. Goethals, S. Laur, H. Lipmaa, and T. Mielikainen. On private scalar product computation for privacy-preserving data mining. *Information Security and Cryptology V ICISC 2004*, pages 104–120, 2004.

[11] O. Goldreich. *Foundations of Cryptography, Volumne II Basic Applications*. Cambridge University Press, 2004.

[12] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *STOC '87*, pages 218–229, New York, NY, USA, 1987. ACM Press.

[13] J. Kilian. Founding crytpography on oblivious transfer. In *STOC '88*, pages 20–31, New York, NY, USA, 1988. ACM.

[14] J. Kilian. A general completeness theorem for two party games. In *STOC '91*, pages 553–560, New York, NY, USA, 1991. ACM.

[15] D. Malkhi, N. Nisan, B. Pinkas, and Y. Sella. Fairplay — a secure two-party computation system. In *Proceedings of the 13th Symposium on Security, Usenix*, pages 287–302, 2004.

[16] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. *Advances in Cryptology–EUROCRYPT'99*, 1999.

[17] C.-H. Shen, J. Zhan, D.-W. Wang, T.-S. Hsu, and C.-J. Liau. Information-theoretically secure number-product protocol. *2007 International Conference on Machine Learning and Cybernetics*, 5:3006–3011, 19-22 Aug. 2007.

[18] L. R. Turner. Inverse of the vandermonde matrix with applications. Technical Report NASA-TN-D-3547, Glenn Research Center, NASA, Aug 1966.

[19] D.-W. Wang, C.-J. Liau, Y.-T. Chiang, and T.-S. Hsu. Information theoretical analysis of two-party secret computation. *Data and Applications Security XX, Lecture Notes in Computer Science*, 4127:310–317, 2006.

[20] A. C. Yao. Protocols for secure computation. In *Proceddings of the 23rd Annual IEEE Symposium on Foundations of Computer Science*, pages 160–164, November 1982.