Secure Multiparty Computation between Distrusted Networks Terminals

Citation	S-CS, C., & Thinh, N. (2007). Secure multiparty computation between distrusted networks terminals. EURASIP Journal on Information Security, 2007: 51368. doi:10.1155/2007/51368
DOI	10.1155/2007/51368
Publisher	Springer
Version	Version of Record
Terms of Use	http://cdss.library.oregonstate.edu/sa-termsofuse

Hindawi Publishing Corporation
EURASIP Journal on Information Security
Volume 2007, Article ID 51368, 10 pages
doi:10.1155/2007/51368

Research Article

Secure Multiparty Computation between Distrusted Networks Terminals

S.-C. S. Cheung[1] and Thinh Nguyen[2]

[1] *Center for Visualization and Virtual Environments, Department of Electrical and Computer Engineering, University of Kentucky, Lexington, KY 40507, USA*
[2] *School of Electrical Engineering and Computer Science, Oregon State University, 1148 Kelley Engineering Center Corvallis, Oregon, OR 97331-5501, USA*

Correspondence should be addressed to S.-C. S. Cheung, sccheung@ieee.org

Received 7 May 2007; Accepted 12 October 2007

Recommended by Stefan Katzenbeisser

One of the most important problems facing any distributed application over a heterogeneous network is the protection of private sensitive information in local terminals. A subfield of cryptography called secure multiparty computation (SMC) is the study of such distributed computation protocols that allow distrusted parties to perform joint computation without disclosing private data. SMC is increasingly used in diverse fields from data mining to computer vision. This paper provides a tutorial on SMC for nonexperts in cryptography and surveys some of the latest advances in this exciting area including various schemes for reducing communication and computation complexity of SMC protocols, doubly homomorphic encryption and private information retrieval.

1. INTRODUCTION

The proliferation of capturing and storage devices as well as the ubiquitous presence of computer networks make sharing of data easier than ever. Such pervasive exchange of data, however, has increasingly raised questions on how sensitive and private information can be protected. For example, it is now commonplace to send private photographs or videos to the hundreds of online photoprocessing stores for storage, development, and enhancement like sharpening and red-eye removal. Few companies provide any protection of the personal pictures they receive. Hackers or employees of the store may steal the data for personal use or distribute them for personal gain without consent from the owner.

There are also security applications in which multiple parties need to collaborate with each other but do not want any of their own private data disclosed. Consider the following example: a law-enforcement agency wants to search for possible suspects in a surveillance video owned by private company A, using a proprietary software developed by another private company B. The three parties involved all have information they do not want to share with each other: the criminal biometric database from law enforcement, the surveillance tape from company A, and the proprietary software from company B.

Encryption alone cannot provide adequate protection when performing the aforementioned applications. The encrypted data needs to be decrypted at the receiver for processing and the raw data will then become vulnerable. Alternatively, the client can download the software and process her private data in a secure environment. This, however, runs the risk of having the proprietary technology of the software company pirated or reverse-engineered by hackers. The Trusted Computing (TC) Platform may solve this problem by executing the software in a secure memory space of the client machine equipped with a cryptographic coprocessor [1]. Besides the high cost of overhauling the existing PC platform, the TC concept remains highly controversial due to its unbalanced protection of the software companies over the consumers [2].

The technical challenge to this problem lies in developing a joint computation and communication protocol to be executed among multiple distrusted network terminals without disclosing any private information. Such a protocol is called a secure multiparty computation (SMC) protocol and has been an active research area in cryptography for more than twenty years [3]. Recently, researchers in other disciplines such as signal processing and data mining have begun to use SMC to solve various practical problems. The goal of this paper is to provide a tutorial on the basic theory of SMC and to survey recent advances in this area.

2. PROBLEM FORMULATION

The basic framework of SMC is as follows: there are n parties $P_1, P_2, \dots, P_n$ on a network who want to compute a joint function $f(x_1, x_2, \dots, x_n)$ based on private data x_i owned by party P_i for $i = 1, 2, \dots, n$. The goal of the SMC is that P_i will not learn anything about x_j for $j \neq i$ beyond what can be inferred from her private data x_i and the result of the computation $f(x_1, x_2, \dots, x_n)$. SMC can be trivially accomplished if there is a special server, trusted by every party with its private data, to carry out the computation. This is not a practical solution as it is too costly to protect such a server. The objective of any SMC protocol is to emulate this ideal model as much as possible by using clever transformations to conceal the private data.

Almost all SMC protocols are classified based on their models of security and adversarial behaviors. The most commonly used security models are perfect security and computational security, which will be covered in Sections 3 and 4, respectively. Adversarial behaviors are broadly classified into two types: semihonest and malicious. A dishonest party is called semihonest if she follows the SMC protocol faithfully but attempts to find out about other's private data through the communication. A malicious party, on the other hand, will modify the protocol to gain extra information. We will focus primarily on semihonest adversaries but briefly describe how the protocols can be fortified to handle malicious adversaries.

We also assume that private data are elements from a finite field F and the target function $f(\cdot)$ can be implemented as a combination of the field's addition and multiplication. This is a reasonably general computational model for two reasons: first, at the lowest level, any digital computing device can be modeled by setting F as the binary field with the XOR as addition and AND as multiplication. Second, while most signal processing and scientific computation are described using real numbers, we can approximate the real numbers with a reasonably large finite field and estimate any analytical function using a truncated version of its power series expansion, which consists of only additions and multiplications.

3. SMC WITH PERFECT SECURITY

In this section, we discuss perfectly secure multiparty computation (PSMC) in which an adversary will learn nothing about the secret numbers of the honest parties no matter how computationally powerful the adversary is. The idea is that while the adversary may control a number of parties who receive messages from other honest senders, these messages provide no useful information about the secret numbers of the senders.

One of the basic tools used in PSMC is *secret sharing*. A t-out-of-m secret-sharing scheme breaks a secret number x into *mshares* $r_1, r_2, \dots, r_m$ such that x cannot be reconstructed unless an adversary obtains more than $t - 1$ shares with $t \leq m$. The importance of a secret-sharing scheme in PSMC is illustrated by the following example: in a 2-party secure computation of $f(x_1, x_2)$, party P_i will use a 2-out-of-2 secret-sharing scheme to break x_i into r_{i1} and r_{i2}, and share r_{ij} with party P_j. Each party then computes the function using the shares received, resulting in $y_1 \triangleq f(r_{11}, r_{21})$ at P_1 and $y_2 \triangleq f(r_{12}, r_{22})$ at P_2. If the secret-sharing scheme is *homomorphic* under the function $f(\cdot)$, that is, y_1 and y_2 are themselves secret shares of the desired function $f(x_1, x_2)$, $f(x_1, x_2)$ can then be easily computed by exchanging y_1 and y_2 between the two parties. Under our computational model, all SMC problems can be solved if the secret-sharing scheme is *doubly homomorphic*—it preserves both addition and multiplication. One such scheme was invented by Adi Shamir which we will explain next [4].

In Shamir's secret-sharing scheme, a party hides her secret number x as the constant term of a secret polynomial $g(z)$ of degree $t - 1$,

$$g(z) \triangleq a_{t-1}z^{t-1} + a_{t-2}z^{t-2} + \cdots + a_1 z + x. \tag{1}$$

The coefficients a_1 to a_{t-1} are random coefficients distributed uniformly over the entire field. Given the polynomial $g(z)$, the secret number x can be recovered by evaluating it at $z = 0$. The secret shares are computed by evaluating $g(z)$ at $z = 1, 2, \dots, m$ and are distributed to m other parties. It is assumed that each party knows the degree of $g(z)$ and the value z at which her share is evaluated. We follow the convention that the share received by party P_i is evaluated at $z = i$.

If an adversary obtains any t shares $g(z_1), g(z_2), \dots, g(z_t)$ with $z_i \in \{1, 2, \dots, m\}$, the adversary can then formulate the following polynomial $\hat{g}(z)$:

$$\hat{g}(z) \triangleq \sum_{i=1}^{t} g(z_i) \frac{\prod_{j=1, j\neq i}^{t} (z - z_j)}{\prod_{j=1, j\neq i}^{t} (z_i - z_j)}. \tag{2}$$

We claim that $\hat{g}(z)$ is identical to the secret polynomial $g(z)$: first, the degree $\hat{g}(z)$ is $t - 1$, same as that of $g(z)$. Second, $\hat{g}(z) = g(z)$ for $z = z_1, z_2, \dots, z_t$ because, when evaluating $\hat{g}(z)$ at a particular $z = z_i$, every term inside the summation in (2) will go to zero except for the one that contains $g(z_i)$ it simply becomes $g(z_i)$ as the multiplier becomes one. Consequently, the $(t - 1)$th-degree polynomial $g(z) - \hat{g}(z)$ will have t roots. As the number of roots is higher than the degree, $g(z) - \hat{g}(z)$ must be identically zero or $\hat{g}(z) \equiv g(z)$. As a result, the adversary can reconstruct the secret number $x = \hat{g}(0)$.

On the other hand, the adversary will have no knowledge about x even if it possesses as many as $t - 1$ shares. This is because, for any arbitrary secret number x', there exists a polynomial $h(z)$ such that $h(0) = x'$ and $h(z_i) = g(z_i)$ for

$i = 1, 2, \ldots, t-1$. $h(z)$ is given as follows and its properties is similar to those of (2):

$$h(z) \triangleq x' \frac{\prod_{j=1}^{t-1}(z - z_j)}{\prod_{j=1}^{t-1}(-z_j)} + \sum_{i=1}^{t-1} g(z_i) \frac{z\prod_{j=1, j\neq i}^{t-1}(z - z_j)}{z_i\prod_{j=1, j\neq i}^{t-1}(z_i - z_j)}. \tag{3}$$

Shamir's secret-sharing scheme is obviously homomorphic under addition: given two secret $(t-1)$th-degree polynomials $g(z)$ and $h(z)$, the secret shares of $g(z) + h(z)$ are simply the summation of their respective secret shares $g(1) + h(1), g(2) + h(2), \ldots, g(m) + h(m)$. Secrecy is also maintained as the coefficients of $g(z) + h(z)$, except for the constant term which is the sum of all the secret numbers, are uniformly distributed and no party can gain additional knowledge about others' secret shares. On the other hand, the degree of the product polynomial $g(z)h(z)$ increases to $2(t-1)$. The locally computed shares $g(1)h(1), g(2)h(2), \ldots, g(m)h(m)$ cannot completely specify $g(z)h(z)$ unless the number of shares m is strictly larger than $2(t-1)$ or equivalently, $t \le \lceil m/2 \rceil$. Even if this condition is satisfied, a series of product can easily result in a polynomial with degree higher than m. Furthermore, the coefficients of the product polynomial is not entirely random, for example, they are related in such a way that the polynomial can be factored by the original polynomials. These problems can be solved by first assuming that $t \le \lceil m/2 \rceil$ and then replacing the product polynomial by a new $(t-1)$th-degree polynomial as follows.

P_i first computes $g(i)h(i)$ and then generates a random $(t-1)$th-degree polynomial $q_i(z)$ with $q_i(0) = g(i)h(i)$. Again, using the secret-sharing scheme, P_i sends share $q_i(j)$ to party P_j for $j = 1, 2, \ldots, m$. This step leaks no information about the local product $g(i)h(i)$. In the final step, P_i computes d_i based on all the received shares $q_j(i)$ for $j = 1, 2, \ldots, m$,

$$d_i \triangleq \sum_{j=1}^{m} \gamma_j q_j(i), \tag{4}$$

where γ_j for $j = 1, 2, \ldots, m$ solve the following equation:

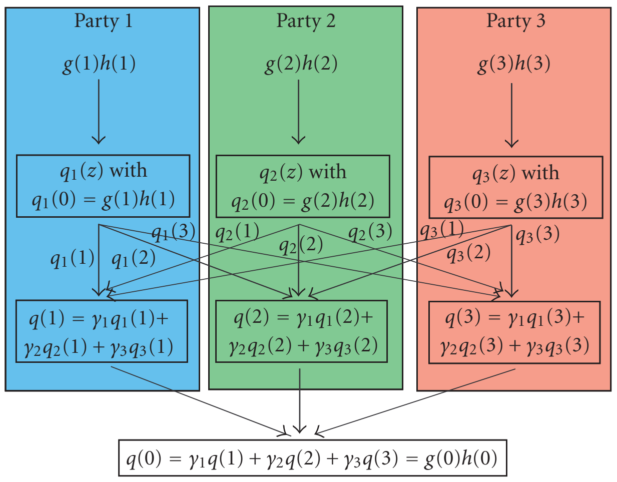
$$g(0)h(0) = \sum_{j=1}^{m} \gamma_j g(j)h(j). \tag{5}$$

Before explaining how P_i can solve (5) without knowing $g(0)h(0)$ and $g(j)h(j)$ for $j \neq i$, we first note that d_i for $i = 1, 2, \ldots, m$ are shares of a $(t-1)$th-degree polynomial $q(z)$ defined below:

$$q(z) \triangleq \sum_{j=1}^{m} \gamma_j q_j(z). \tag{6}$$

The coefficients of $q(z)$ are uniformly random as they are linear combinations of uniformly distributed coefficients of $q_j(z)$'s. Furthermore, its constant term is our target secret number $g(0)h(0)$:

$$q(0) = \sum_{j=1}^{m} \gamma_j q_j(0) = \sum_{j=1}^{m} \gamma_j g(j)h(j) = g(0)h(0). \tag{7}$$



Party 1
Party 2
Party 3
$g(1)h(1)$
$g(2)h(2)$
$g(3)h(3)$
$q_1(z)$ with $q_1(0) = g(1)h(1)$
$q_2(z)$ with $q_2(0) = g(2)h(2)$
$q_3(z)$ with $q_3(0) = g(3)h(3)$
$q_1(1)$
$q_1(2)$
$q_1(3)$
$q_2(1)$
$q_2(2)$
$q_2(3)$
$q_3(1)$
$q_3(2)$
$q_3(3)$
$q(1) = \gamma_1 q_1(1) + \gamma_2 q_2(1) + \gamma_3 q_3(1)$
$q(2) = \gamma_1 q_1(2) + \gamma_2 q_2(2) + \gamma_3 q_3(2)$
$q(3) = \gamma_1 q_1(3) + \gamma_2 q_2(3) + \gamma_3 q_3(3)$
$q(0) = \gamma_1 q(1) + \gamma_2 q(2) + \gamma_3 q(3) = g(0)h(0)$

FIGURE 1: This diagram shows how three parties can share the secret $g(0)h(0)$ based on the locally computed products $g(1)h(1), g(2)h(2)$, and $g(3)h(3)$.

The second last equality is because $g(j)h(j)$ is the secret number hidden by the polynomial $q_j(z)$. The last equality is based on (5). This implies that d_i for $i = 1, 2, \ldots, m$ are secret shares of the scalar $g(0)h(0)$. An example of the above protocol in a three-party situation is shown in Figure 1.

To address how each party can solve (5), we note that, based on our assumption $t \le \lceil m/2 \rceil$ the degree of the product polynomial $g(z)h(z)$ is strictly smaller than the number of shares m. Let $g(z)h(z) = a_{m-1}z^{m-1} + \cdots + a_0$. The coefficients a_i's are completely determined by the values $g(z)h(z)$ at $z = 1, 2, \ldots, m$. In other words, the following matrix equation has a unique solution:

$$V\mathbf{a} \triangleq \begin{pmatrix} 1^{m-1} & 1^{m-2} & \cdots & 1^0 \\ 2^{m-1} & 2^{m-2} & \cdots & 2^0 \\ \vdots & \vdots & & \vdots \\ m^{m-1} & m^{m-2} & \cdots & m^0 \end{pmatrix} \begin{pmatrix} a_{m-1} \\ a_{m-2} \\ \vdots \\ a_0 \end{pmatrix} = \begin{pmatrix} g(1)h(1) \\ g(2)h(2) \\ \vdots \\ g(m)h(m) \end{pmatrix}. \tag{8}$$

The $m \times m$ invertible matrix V is called the Vandermonde matrix and it is a constant matrix. Taking its inverse $W = V^{-1}$ and considering the last row entries W_{mi} for $i = 1, 2, \ldots, m$, we have

$$\sum_{i=1}^{m} W_{mi} g(i)h(i) = a_0 = g(0)h(0). \tag{9}$$

Comparing (9) with (5), we have $W_{mi} = \gamma_i$ for $i = 1, 2, \ldots, m$, which are constants.

The condition $t \le \lceil m/2 \rceil$ on using Shamir's scheme in PSMC posts a restriction on the number of dishonest parties tolerated—it implies that the number of honest parties must be a strict majority. In particular, we cannot use this scheme for a two-party SMC in which one party has to assume that the other party is dishonest. A surprising result in [5] shows that the condition $t \le \lceil m/2 \rceil$ is not a weakness of Shamir's scheme—in fact, except for certain trivial functions,[1] it is *impossible to compute any $f(x_1, x_2, \ldots, x_m)$ with perfect security if the number of dishonest parties equals to or exceeds $\lceil m/2 \rceil$*.

To conclude this section, we briefly describe how PSMC protocols can be modified to handle malicious parties. There are two types of disruption: first, a malicious party can output erroneous results and second, she may perform an inconsistent secret-sharing scheme such as evaluating the polynomial at random points. Provided the number of malicious parties is less than one third of the total number of parties, the first problem can be solved by replacing (2) with a robust extrapolation scheme based on Reed-Solomon codes [5]. This bound on the number of malicious parties can be raised to one half by combining interactive zero-knowledge proof with a broadcast channel [6]. The second problem can be solved by using a verifiable secret-sharing (VSS) scheme in which the sender needs to provide auxiliary information so that the receivers can verify the consistency of their shares without gaining knowledge of the secret number [5].

4. SMC WITH COMPUTATIONAL SECURITY

It is unsatisfactory that PSMC introduced in Section 3 cannot even provide secure two-party computation. Instead of relying on perfect security, modern cryptographical techniques primarily use the so-called *computational security* model. Under this model, secrets are protected by encoding them based on a mathematical function whose inverse is difficult to compute without the knowledge of a secret key. Such a function is called *one-way trapdoor function* and the concept is used in many public-key cipher: a sender who wants to send a message m to party P will first compute a ciphertext $c = E(m, k)$ based on the publicly known encryption algorithm $E(\cdot)$'s and P's advertised public key k. The encryption algorithm acts as a one-way trapdoor function because a computationally bounded eavesdropper will not be able to recover m given only c and k. On the other hand, P can recover m by applying a decoding algorithm $D(E(m, k), s) = m$ using her secret key s. Unlike perfectly secure protocols in which the adversary simply does not have any information about the secret, the adversary in the computationally secure model is unable to decrypt the secret due to the computational burden in solving the inverse problem. Even though it is still a conjecture that true one-way trapdoor functions exist and future computation platforms like quantum computer may drastically change the landscape of these functions, many one-way function candidates exist and are routinely used in practical security systems.[2]

The most fundamental result in SMC is that it is possible to design general computationally secure multiparty computation (CSMC) protocols to handle arbitrary number of dishonest parties [3]. In this section, we will discuss the basic construction of these protocols. Similar to Section 3, we consider the protocols for addition and multiplication in finite fields. We will concentrate on the canonical two-party case but our construction can be easily extended to more than two parties. Our starting point of building general CSMC is a straightforward secret-sharing scheme: each secret number is simply broken down as a sum of two uniformly distributed random numbers: $x_1 = r_{11} + r_{12}$ and $x_2 = r_{21} + r_{22}$. P_i then sends r_{ij} to P_j for $j \neq i$. This scheme is clearly homomorphic under addition

$$x_1 + x_2 = (r_{11} + r_{21}) + (r_{12} + r_{22}). \qquad (10)$$

Multiplication, on the other hand, introduces cross-term $r_{11}r_{22}$ which breaks the homomorphism the homomorphism

$$x_1 x_2 = r_{11}r_{21} + r_{12}x_2 + r_{11}r_{22}. \qquad (11)$$

While the first two terms can be locally computed by P_1 and P_2, respectively, it is impossible to compute the third term $r_{11}r_{22}$ without having one party revealed the actual secret number to the other. In order to accomplish this under the computational security model, we will make use of a general cryptographic protocol called the *oblivious transfer* (OT).

A 1-out-of-N OT protocol allows one party (the chooser) to read one entry from a table with N entries hosted by another party (the sender). Provided that both parties are computationally bounded, the OT protocol prevents the chooser from reading more than one entry and the sender from knowing the chooser's choice. We first show how the OT protocol can be used to break $r_{11}r_{22}$ in (11) into random shares u and v such that $r_{11}r_{22} = u + v$. Assume our finite field has N elements. The sender P_1 generates a random u and then creates a table T with N entries shown in Table 1.[3] Using the OT protocol, the chooser P_2 selects the entry $v \triangleq T(r_{22}) = r_{22}r_{11} - u$ without letting P_1 know her selection or inspecting any other entries in the table.

Table 1: OT table at P_1.

Key	Values
0	$-u$
1	$1r_{11} - u$
2	$2r_{11} - u$
$\vdots$	$\vdots$
r_{22}	$r_{22}r_{11} - u$
$\vdots$	$\vdots$
$N-2$	$(N-2)r_{11} - u$
$N-1$	$(N-1)r_{11} - u$

It remains to show how OT provides the security guarantee. A 1-out-of-N OT protocol consists of the following five steps.

(1) P_1 sends N randomly generated public keys $k_0, k_1, \ldots, k_{N-1}$ to P_2.

[1] The exceptions are those functions that are separable or $f(x_1, x_2, \ldots, x_m) = f_1(x_1)f_2(x_2)\cdots f_m(x_m)$.

[2] A list of one-way function candidates can be found in [7, Chapter 1].

[3] The role of P_1 and P_2 can be interchanged with proper adjustment to Table 1 entries.

(2) P_2 selects $k_{r_{22}}$ based on her secret number r_{22}, encrypts her public key k' using $k_{r_{22}}$, and sends $E(k', k_{r_{22}})$ back to P_1.
(3) As P_1 does not know P_2's key selection, P_1 decodes the incoming message using all possible keys or $\widehat{k_i'} = D(E(k', k_{r_{22}}), s_i)$ with private keys s_i for $i = 0, 1, \ldots, N-1$. Only one of $\widehat{k_i'}$'s ($\widehat{k_{r_{22}}'}$) matches the real key k' but P_1 has no knowledge of it.
(4) P_1 encrypts each table entry $T(i)$ using $\widehat{k_i'}$ and sends $E(T(i), \widehat{k_i'})$ for $i = 0, 1, \ldots, N-1$ to P_2.
(5) P_2 decrypts the r_{22}th message using her private key s': $D(E(T(r_{22}), \widehat{k_{r_{22}}'}), s') = T(r_{22})$ as $k_{r_{22}}' = k'$ is the public key corresponding to the secret key s'. P_2 then obtains her random share of $v = T(r_{22}) = r_{22}r_{11} - u$. Note that P_2 will not be able to decrypt any other message $E(T(i), \widehat{k_i'})$ for $i \neq r_{22}$ as it requires the knowledge of P_1's secret key s_i.

It is clear from the above procedure that OT can accomplish a table lookup secure to both P_1 and P_2. As the definition of the table is arbitrary, OT can support secure two-party computation of any finite field function. Following similar procedures as in Section 3, the above construction can be extended using standard zero-knowledge proof and verifiable secret-sharing scheme to handle malicious parties that do not follow the prescribed protocols [8, Chapter 7].

5. RECENT ADVANCES

In Sections 3 and 4, we present the construction of general SMC protocols under the perfect security model and the computational security model. While most of these results are established in 1980s, SMC continues to be a very active research area in cryptography and its applications begin to appear in many other disciplines. Recent advances focus on better understanding of the security strength of individual protocols and their composition, improving CSMC protocols in terms of their computation complexity [9, 10] and communication cost [11–14], relating SMC to error-correcting coding [15, 16], and introducing SMC to a variety of applications [17–22]. The rigorous study of protocol security is beyond the scope of this paper, and thus we will focus on the remaining three topics.

5.1. *Reduction of computation complexity and communication cost*

Both the computation complexity and communication cost of the 1-out-of-N OT protocol depend linearly on the size N of the sender's table that defines the function—it requires $O(N)$ invocations of a public-key cipher and $O(N)$ messages exchanged between the sender and the chooser. In many practical applications, the value of N could be very large. For example, computing a general function on 32-bit computers requires a table of $N = 2^{32}$ or more than four billion entries! This renders our basic version of OT hopelessly impractical. Improving the computation efficiency and reducing the communication requirement of OT and other CSMC protocols thus become the focus of intensive research effort.

In [9], Naor and Pinkas showed that the 1-out-of-N OT protocol can be reduced to applying a 1-out-of-2 OT protocol $\log_2 N$ times. The idea is that the two parties repeatedly use the 1-out-of-2 OT on individual bits of the binary representation of the chooser's secret number x_2: in the ith round, the sender will present two keys K_{i0} and K_{i1} to the chooser who will choose $K_{ix_2[i]}$ based on $x_2[i]$, the ith bit of x_2. The keys K_{i0} and K_{i1} for $i = 1, 2, \ldots, \log_2 N$ are used by the sender to encrypt the table entries $T(k)$ using the binary representation of k as follows:

$$E(T(k)) = T(k) \oplus \bigoplus_{i=1}^{\log_2 N} f(K_{ik[i]}), \tag{12}$$

where k is a $\log N$-bit number, $f(s)$ is a random number generated by seed s, and $\oplus$ denotes XOR. The entire encrypted table is sent to the chooser. Since the chooser already knows $K_{ix_2[i]}$ for $i = 1, 2, \ldots, \log_2 N$, she can use them to decrypt $E(T(x_2))$ as follows:

$$T(x_2) = E(T(x_2)) \oplus \bigoplus_{i=1}^{\log_2 N} f(K_{ix_2[i]}). \tag{13}$$

The same authors further improved the computation complexity of the 1-out-of-2 OT protocol in [10]. They showed that it is possible to use *one* exponentiation, the most complex operation in a public-key cipher, for any number of simultaneous invocations of the 1-out-of-2 OT at the cost of increasing the communication overhead. Their public-key cipher is based on the assumed difficulty of the Decisional Diffie-Hellman problem whose encryption process enables the sender to prepare all her encrypted messages with one exponentiation without any loss of secrecy.

An aspect that the above algorithms do not address is the communication requirement of general CSMC protocols. There are three different facets to the communication problem. First, our basic version of the 1-out-of-N OT protocol requires the sender to send N random keys and N encrypted messages to the chooser. The random keys can be considered as setup cost, provided that the sender changes her random share u and the chooser changes her key k' in every invocation of the protocol. However, it seems necessary to send the N encrypted messages every time as the messages depend on u. A closer examination reveals that all the chooser needs is one particular message that corresponds to her secret number. The entire set of N messages is sent simply to obfuscate her choice from the sender. This subproblem of obfuscating a selection from a public data collection is called private information retrieval (PIR). PIR attracts much research interest lately and is treated in Section 5.2. It suffices to know that there are techniques that can reduce the communication cost from $O(N)$ to $O(\log N)$ [23].

The second facet involves the communication cost of the original unsecured implementation of the target function. The CSMC protocols in Section 4 provide a systematic procedure to secure each addition and multiplication operation in the original implementation. However, not all operations need to be secured—local operations can be performed without any modification. As such, it is important to minimize the number of cross-party operations that need to be fortified with the OT protocol. Consider the following example: P_1 and P_2, each with $n/2$ secret numbers, want to find the median of the entire set of n numbers. The best known unsecured algorithm to find the median requires $O(n)$ comparison operations. To make this algorithm secure, we can use the 1-out-of-N OT protocol to implement each comparison,[4] resulting in communication requirement of $O(n \log N)$. This, however, is not the optimal solution—a distributed median-finding algorithm requires much less communication [13]. The idea is to have P_1 and P_2 first compared with their respective local medians. The party with the the larger median can then discard the half of the local data larger than the local median—the global median cannot be in this portion of the local data as the global median must be smaller than the larger of the two local medians. Following the same logic, the other party can discard the smaller half of her local data. The two parties again compare their local medians of the remaining data until exhaustion. Notice that all the local computation can be done without invocations of OT. As a result, this algorithm only requires $O(\log n)$ cross-party secure comparison and this results in a communication cost of $O(\log n \log N)$, a significant reduction from the naive implementation. In fact, it has been shown that if a communication-efficient unsecured implementation exists for a general function, we can always convert it into a secure one without much increase in communication [12].

The final facet of communication requirements has to do with the interactivity of the CSMC protocols. All the protocols introduced thus far require multiple rounds of communications between the parties. Such frequent interaction is undesirable in many applications such as batch processing in which one party needs to reuse many times the same secret information from another party, and asymmetric computation in which a low-complexity client wants to leverage a sophisticated server to privately perform a complex computation. Earlier work in this area showed that one round of message exchange is indeed possible for secure computation of any function [11]. However, the length of the replied message depends on the complexity of the implementation of the function. As a result, this requires the end receiver to devote much time in decoding the message even though the output can be as small as a binary decision. This problem can be resolved using a doubly homomorphic public-key encryption scheme in which arbitrary computation can be done on the encrypted data without size expansion. It is an open problem in cryptography on whether a doubly homomorphic encryption scheme exists. The closest scheme, which we will explain next, can support arbitrary numbers of additions and one multiplication on encrypted data [14].

The construction is based on two public-key ciphers defined on two different finite cyclic groups G and $\widehat{G}$ of the same size $n = q_1 q_2$, where q_1 and q_2 are large private primes. These two groups are related by a special bilinear map $e : G \times G \to \widehat{G}$ such that $e(u^\alpha, v^\beta) = e(u, v)^{\alpha\beta}$ for arbitrary $u, v \in G$ and integers α, β.[5] Furthermore, $e(g, g)$ is a generator for $\widehat{G}$ if g is a generator for G. The public keys for the cipher defined on G are a generator g and a random $h = g^{\alpha q_2}$ for some α. The public keys for the cipher on $\widehat{G}$ are $\hat{g} = e(g, g)$ and $\hat{h} = e(g, h) = \hat{g}^{\alpha q_2}$. Given a message m, the sender generates a random integer r and computes the ciphertext $C = g^m h^r \in G$. To decrypt this ciphertext, the receiver first removes the random factor by raising C to the power of the private key q_1:

$$C^{q_1} = \left(g^m h^r\right)^{q_1} = \left(g^{q_1}\right)^m g^{\alpha q_2 r q_1} = \left(g^{q_1}\right)^m, \tag{14}$$

where we use the basic fact $g^{q_1 q_2} = g^n = 1$ from group theory. Provided that the message space is small enough, the receiver can then retrieve m by computing the discrete logarithm of C^{q_1} base g^{q_1}. The security of the cipher is based on the assumed hardness of the so-called subgroup decision problem of which we refer the readers to the original paper [14]. We now focus on the homomorphic properties of this scheme. Given two ciphertext messages $C_1 = g^{m_1} h^{r_1}$ and $C_2 = g^{m_2} h^{r_2}$, it is easy to see that $C_1 C_2 = g^{m_1+m_2} h^{r_1+r_2}$ which is the ciphertext of message $m_1 + m_2$. For multiplication, we apply the bilinear map $e(\cdot, \cdot)$ on C_1 and C_2:

$$\begin{aligned} e(C_1, C_2) &= e\left(g^{m_1} h^{r_1}, g^{m_2} h^{r_2}\right) \\ &= e\left(g^{m_1+\alpha q_2 r_1}, g^{m_2+\alpha q_2 r_2}\right) \\ &= e(g, g)^{m_1 m_2 + \alpha q_2 (m_1 r_2 + m_2 r_1 + \alpha q_2 r_1 r_2)} \\ &= e(g, g)^{m_1 m_2} e(g, h)^{m_1 r_2 + m_2 r_1 + \alpha q_2 r_1 r_2} \\ &= \hat{g}^{m_1 m_2} \hat{h}^{r'}. \end{aligned} \tag{15}$$

The last expression is clearly a ciphertext for $m_1 m_2$. Unfortunately, $e(C_1, C_2)$ belongs to $\widehat{G}$, not in G. This means that one cannot further combine this with other ciphertexts in G and as such this scheme falls short of being a completely homomorphic encryption scheme.

5.2. Private information retrieval

Private information retrieval (PIR) protocols allow a party (a user) to select a record from a database owned by another party (a server) without the server knowing the selection of the user. PIR is a step in OT as explained in Section 5.1. Unlike OT, PIR does not prevent the sender from obtaining information about the collection beyond her choice. Due to its asymmetric protection, the paradigm of PIR is useful for privacy protection of ordinary citizens in using search engine, shopping at online stores, participating in public survey and electronic voting. As we have seen in Section 5.1, the simplest form of PIR is to send the entire database to the user. This imposes a communication cost in the order of the size

[4] Secure comparison is also called the Secure Millionaire Problem, one of the earliest problems studied in SMC literature [3].

[5] An example of such construction is based on the modified Weil paring on the elliptic curve $y^2 = x^3 + 1$ defined over a finite field [14].

of the database. Recent advances in PIR protocols, however, show that the goal can be accomplished with a much smaller communication overhead.

The problem of PIR was first proposed in the seminal paper by Chor et al. as follows [24]: the server has an n-bit binary string x, and a user wants to know $x[i]$, the ith bit of x, without the server knowing about i. The first important result shown in [24] is that, under the perfect security model, it is impossible to send less data than the trivial solution of sending the entire x to the user. On the other hand, if identical databases are available at $k \geq 2$ noncolluding servers, then perfect security can be achieved with the communication cost of $O(n^{1/k})$. Their results are based on the following basic two-server scheme that allows a user to privately obtain $x[i]$ by receiving a single bit from each of the two servers. Let us denote

$$S \otimes a = \begin{cases} S \cup \{a\}, & \text{if } a \notin S, \\ S \setminus \{a\}, & \text{if } a \in S. \end{cases} \tag{16}$$

The user first randomly selects the indexes $j \in \{1, 2, \ldots n\}$ with probability of 1/2 for each value of j, to form a set S. Next, the user computes $S \otimes i$, where i is the desired index. The user then sends S to server one and $S \otimes i$ to server two. Upon receiving S, server one replies to the user with a single bit which is the result of XORing of all the bits in the positions specified by S. Similarly, server two replies to the user with a single bit which is the result of XORing of all the bits in the positions specified by $S \otimes i$. The user then computes $x[i]$ by XORing the two bits received from the two servers. This scheme works because every position $j \neq i$ will appear twice—one in S and one in $S \otimes i$, therefore the result from XORing of all $x[j]$'s together will be 0. On the other hand, i appears only once in either S or $S \otimes i$, therefore the result of XORing of all $x[j]$'s and $x[i]$ will be $x[i]$. Provided the two servers do not collude, every bit is equally likely to be selected by the user. In this scheme, each server sends one bit to the user but the user has to send an n-bit message[6] to each server. Thus, the overall communication cost is still $O(n)$. With minor modification, this basic scheme can be extended to reduce the number of bits sent by the user to $O(n^{1/k})$ [24].

Recently, an interesting connection is made between PIR and a special type of forward-error-correcting codes (FEC) called locally decodable codes (LDC) and it has created a flurry of interest in the information theory community [16]. FEC is used to combat transmission errors by adding redundancy to the transmitted data. Formally, the sender uses an encoding function $C(\cdot)$ to map an n-bit message x to an m-bit message $C(x)$ with $m > n$, and then sends $C(x)$ over a noisy channel. Upon receiving a string y possibly different from $C(x)$, a receiver attempts to recover x using a decoding algorithm $D(C(x))$. In the conventional FEC, it will takes at least $O(n)$ complexity to recover an n-bit x since $O(n)$ is required just to record x. LDC, on the other hand, allows the user to inspect only a small fraction of $C(x)$, say $k \ll n$ bits, in order to fully recover a specific bit $x[i]$ in x. Furthermore, each bit in $C(x)$ can be used in a k-bit subset to recover $x[i]$. As such, the knowledge of a particular bit in $C(x)$ being used provides no information about which $x[i]$ is being recovered. To see how LDC is used in PIR, we assume that each of the k servers has the same m-bit $C(x)$ generated using an LDC encoding function on the n-bit database x. In order to retrieve $x[i]$, the user sends $q_1, q_2, \ldots, q_k \in \{1, 2, \ldots, m\}$, the locations of bits in $C(x)$ needed to recover $x[i]$, to each of the k servers, respectively. Note that these locations depend only on i and the particular LDC used. Upon receiving q_j, the jth server simply replies with $C(x)[q_j]$ for $j = 1, 2, \ldots, k$. After gathering all the k replies, the user can then run the decoding algorithm to recover $x[i]$. Using this framework, the communication cost of the PIR system is $k(l + \log m)$ with $k \log m$ and kl corresponded to the user's and server's communication costs, respectively.

In fact, the two-server basic scheme introduced earlier can be viewed as using the Hadamard code in the LDC framework. The Hadamard code $H(x)$ of an n-bit message x has 2^n bits. The kth bit of $H(x)$ for $k \in \{0, 1, \ldots, 2^n - 1\}$ is defined as follows:

$$H(x)[k] = \bigoplus_{j=1}^{n} x[j]k[j]. \tag{17}$$

To retrieve $x[i]$ from the servers, the user first randomly picks an n-bit number k, and then sends k to server one and $k \oplus e_i$ to server two, where e_i is an n-bit number with a single one in the ith position. Upon receiving k and $k \oplus e_i$, servers one and two reply with $H(x)[k]$ and $H(x)[k \oplus e_i]$, respectively. The user can then decode $x[i]$ by computing

$$\begin{aligned} &H(x)[k] \oplus H(x)[k \oplus e_i] \\ &\quad = \bigoplus_{j=1, j\neq i}^{n} x[j]k[j] \oplus x[i]k[i] \oplus \bigoplus_{j=1, j\neq i}^{n} x[j]k[j] \oplus x[i](\sim k[i]) \\ &\quad = x[i](k[i] \oplus \sim k[i]) = x[i]. \end{aligned} \tag{18}$$

The symbol $\sim$ denotes negation. This scheme is almost equivalent to the scheme by Chor et al., except that the XOR of all possible selections of bits in x are already contained in the Hadamard code $H(x)$. We mention again that the communication cost of this scheme is $O(n)$ due to the exponential code length of the Hadamard code. Nevertheless, the possibility of using better error-correcting codes in the place of the Hadamard code opens many opportunities for new PIR schemes. PIR schemes based on Reed-Solomon codes and Reed-Muller codes can be found in [16]. The best published result on PIR uses LDC to achieve a communication complexity of $O(n^{10^{-7}})$ with three noncolluding servers [25].

All of the above constructions provide PIR under the perfect security model. By making certain computational assumptions, PIR can also achieve sublinear communication complexity with only one database [23, 26]. We briefly review the scheme in [26] as follows: it is based on the assumed hardness of determining whether a number in a finite field

[6] The message is simply an n-bit number with ones indicating the desired bit.

F is a quadratic residue, that is, without knowing the prime factorization of the field size N, it is difficult to compute the following predicate:

$$QR(u) = \begin{cases} 1 & \text{if } u = v^2 \text{ for some } v \in F, \\ 0 & \text{otherwise.} \end{cases} \tag{19}$$

It is easy to see that $QR(\cdot)$ is homomorphic under multiplication, that is, $QR(xy) = QR(x)QR(y)$. The basic principle of using QR to retrieve $x[i]$ is straightforward: the user sends the server n numbers $y_1, \dots, y_n \in F$, all of them quadratic residues except y_i, that is, $QF(y_j) = 1$ for $j \neq i$ and $QF(y_i) = 0$. The server then replies with $m \in F$ computed as follows:

$$m \triangleq \Pi_{j=1}^{n} w_j, \quad \text{where } w_j = \begin{cases} y_j & \text{if } x[j] = 0, \\ y_j^2 & \text{if } x[j] = 1. \end{cases} \tag{20}$$

Since all y_j's are quadratic residues except for y_i, we have $QR(w_j) = 1$ for $j \neq i$ and $QR(w_i) = x[i]$. Combining the homomorphic property, we get the desired result $QR(m) = QR(w_i) = x[i]$. This scheme, however, is very wasteful as the user needs to send $n \log N$ bits. We can improve this by rearranging x as an $s \times t$ matrix M with $s = n^{(L-1)/L}$ and $t = n^{1/L}$ for some integer L. Assume that $x[i]$ is the entry at the ath row and the bth column of M. The user then sends the server y_j, for $j = 1, 2, \dots, t$, all quadratic residues except for y_b. The communication for this step is $O(n^{1/L})$. Using these t numbers, the server carries a similar computation as (20) for each row of M, resulting in m_k for $k = 1, 2, \dots, s$. Of all the m_k's, all the user needs is m_a from the ath row because it is sufficient to retrieve $x[i]$ as $QR(m_a) = x[i]$. Since each of the m_k is a $\log N$-bit number, this is equivalent to carrying out the PIR procedure $\log N$ times—but this time the database size shrinks from n to $s = n^{(L-1)/L}$. This observation allows the same procedure to be applied recursively with exponentially decreasing communication cost. As a result, the communication is dominated by the first step which is $O(n^{1/L})$ and we can make L as big as we want. Subsequent work by Cachin et al. showed that the communication cost can be further reduced to logarithmic complexity [23].

5.3. Practical applications of SMC

While the theoretical studies of SMC have advanced significantly in recent years, developing practical applications using SMC has been slow. The data mining community is the first to introduce SMC into practical usage. The goal is to compute aggregate statistics over private data stored in distributed databases. Using the OT protocol as the core, different SMC protocols have been developed to construct linear algebra routines [27], median computation [13], decision trees [17], neural network [19], and others. Even though these algorithms provide innovative implementations for many data mining schemes, their security relies on modular arithmetic operations on very large integers which are computationally intensive. In a recent study on PIR, the authors of [28] showed that even with the most advanced CPUs, the modular arithmetic in the SMC protocol requires more time than simply sending the entire database through a typical broadband connection.

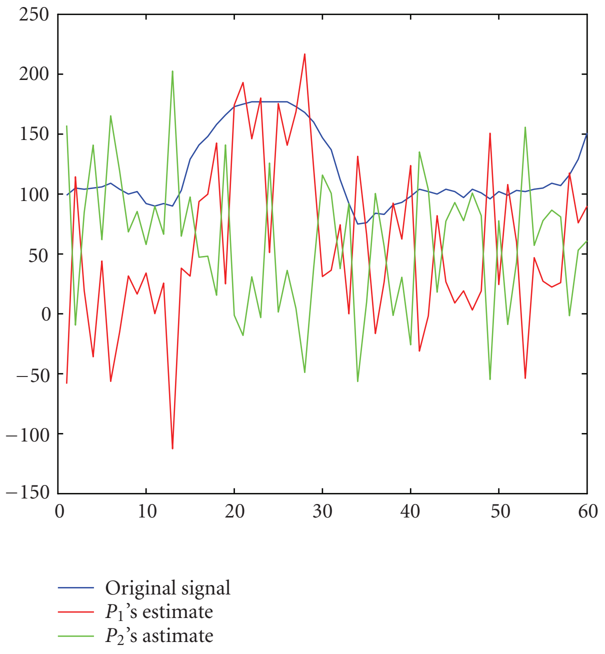


FIGURE 2: Original signal and least-square estimates in secure inner product.

While an algorithm in a typical data mining application may need to handle millions of records on a daily basis, a real-time signal processing algorithm needs to handle millions of samples within milliseconds. Very efficient algorithms have recently been developed at the expense of privacy. The pioneering work by Avidan and Moshe showed the feasibility of building a secure distributed face detector [20]. While keeping OT as the core, they provide an efficient implementation based on the assumption that certain visual features used in the detector are noninvertible and for this they do not leak important information about the images.

Another noteworthy scheme is a collection of statistical routines, developed in [18], that use linear subspace projection for privacy projection. We illustrate the idea with a simple inner product computation. Assume that two parties, P_1 and P_2, have n-dimensional vectors $\mathbf{x}_1$ and $\mathbf{x}_2$, respectively. They both know an invertible matrix M and its inverse M^{-1}. M is broken down into top and bottom halves $T \in \mathbb{R}^{\lfloor n/2 \rfloor \times n}$ and $B \in \mathbb{R}^{(n-\lfloor n/2 \rfloor)\times n}$, while M^{-1} into left and right halves $L \in \mathbb{R}^{n\times\lfloor n/2 \rfloor}$ and $R \in \mathbb{R}^{n\times(n-\lfloor n/2 \rfloor)}$. The inner product $\mathbf{x}_1^T\mathbf{x}_2$ can then be decomposed as follows:

$$\mathbf{x}_1^T\mathbf{x}_2 = \mathbf{x}_1^T M^{-1} M \mathbf{x}_2 = \mathbf{x}_1^T LT\mathbf{x}_2 + \mathbf{x}_1^T RB\mathbf{x}_2. \tag{21}$$

P_1 then sends $\mathbf{x}_1^T R$ to P_2 who computes $\mathbf{x}_1^T RB\mathbf{x}_2$ while P_2 sends $P_1 T\mathbf{x}_2$ so that she can compute $\mathbf{x}_1^T LT\mathbf{x}_2$. P_2 can then send his scalar to P_1 or vice versa to obtain the final answer. They cannot recover each other's data as the transmitted data $\mathbf{x}_1^T R$ and $T\mathbf{x}_2$ are all $n/2$-dimensional vectors. Using a randomly generated M and $\mathbf{x}_1 = \mathbf{x}_2$, Figure 2 shows the least square estimates by both parties based on the received data. Following a similar approach, we have also developed secure two-party routines for linear filtering [21] and thresholding

[22]. Even though all of the above algorithms are computationally very efficient, they all leak private information to a certain degree and thus may not be suitable for applications that demand the utmost privacy and security.

6. CONCLUSIONS

In this article, we have briefly reviewed the foundation of SMC protocols and some of the latest developments. As we do not assume any background in cryptography, we focus on the intuition rather than the rigorous treatment of the subject. Serious readers should consult the comprehensive text of [8] and the collection of papers at specialized bibliography sites [29, 30]. As the demand for secure and privacy-enhancing applications is rapidly growing, we believe that it is a great opportunity for researchers in diverse areas outside of cryptography to understand the concepts of SMC and to develop practical SMC protocols for their respective applications.

ACKNOWLEDGMENT

The authors would like to thank the constructive comments from the anonymous reviewers.

REFERENCES

[1] Trusted Computing Group, "TCG Specification Architecture Overview," April 2004, https://www.trustedcomputinggroup.org.

[2] R. Anderson, "Trusted Computing Frequently Asked Questions," August 2003, http://www.cl.cam.ac.uk/~rja14/tcpa-faq.html.

[3] 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, November 1982.

[4] Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[5] M. Ben-Or, S. Goldwasser, and A. Wigderson, "Completeness thorems for non-cryptographic fault tolerant distributed computation," in *Proceedings of the 20th ACM Symposium on the Theory of Computing*, pp. 1–10, Chicago, Ill, USA, May 1988.

[6] T. Rabin and M. Ben-Or, "Verifiable secret sharing and multiparty protocols with honest majority," in *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*, pp. 73–85, Seattle, Wash, USA, May 1989.

[7] S. Goldwasser and M. Bellare, *Lecture Notes on Cryptography*, Massachusetts Institue of Technology, Cambridge, Mass, USA, 2001.

[8] O. Goldreich, *Foundations of Cryptography: Volume II Basic Applications*, Cambridge University Press, Cambridge, Mass, USA, 2004.

[9] M. Naor and B. Pinkas, "Oblivious transfer and polynomial evaluation," in *Proceedings of the Annual ACM Symposium on Theory of Computing*, pp. 245–254, Atlanta, Ga, USA, 1999.

[10] M. Naor and B. Pinkas, "Efficient oblivious transfer protocols," in *Proceedings of the SIAM Symposium on Discrete Algorithms (SODA '01)*, pp. 448–457, Washington, DC, USA, 2001.

[11] C. Cachin, J. Camenisch, J. Kilian, and J. Muller, "One-round secure computation and secure autonomous mobile agents," in *Proceedings of the 27th International Colloquium on Automata, Languages and Programming*, pp. 512–523, Geneva, Switzerland, July 2000.

[12] M. Naor and K. Nissim, "Communication complexity and secure function evaluation," *Electronic Colloquium on Computational Complexity*, vol. 8, no. 62, 2001.

[13] G. Aggarwal, N. Mishra, and B. Pinkas, "Secure computation of the kth-ranked element," in *Proceedings of Advances in Cryptology International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT '04)*, vol. 3027 of *Lecture Notes in Computer Science*, pp. 40–55, 2004.

[14] D. Boneh, E.-J. Goh, and K. Nissim, "Evaluating 2-DNF formulas on ciphertexts," in *Proceedings of Theory of Cryptography Conference 2005*, vol. 3378 of *Lecture Notes in Computer Science*, pp. 325–341, Cambridge, Mass, USA, February 2005.

[15] W. Gasarch, "A survey on private information retrieval," *The Bulletin of the EATCS*, vol. 82, pp. 72–107, 2004.

[16] L. Trevisan, "Some applications of coding theory in computational complexity," *Quaderni di Matematica*, vol. 13, pp. 347–424, 2004.

[17] Y. Lindell and B. Pinkas, "Privacy preserving data mining," *Journal of Cryptology*, vol. 15, no. 3, pp. 177–206, 2003.

[18] W. Du, Y. S. Han, and S. Chen, "Privacy-preserving multivariate statistical analysis: linear regression and classification," in *Proceedings of the 4th SIAM International Conference on Data Mining*, pp. 222–233, Lake Buena Vista, Fla, USA, April 2004.

[19] Y.-C. Chang and C.-J. Lu, "Oblivious polynomial evaluation and oblivious neural learning," *Theoretical Computer Science*, vol. 341, no. 1–3, pp. 39–54, 2005.

[20] S. Avidan and M. Butman, "Blind vision," in *Proceedings of the 9th European Conference on Computer Vision*, vol. 3953 LNCS of *Lecture Notes in Computer Science*, pp. 1–13, Graz, Austria, May 2006.

[21] N. Hu and S.-C. Cheung, "Secure image filtering," in *Proceedings of IEEE International Conference on Image Processing (ICIP '06)*, Atlanta, Ga, USA, October 2006.

[22] N. Hu and S.-C. Cheung, "A new security model for secure thresholding," in *Proceedings of IEEE International Conference on Acoustic, Speech and Signal Processing (ICASSP '07)*, Honolulu, Hawaii, USA, April 2007.

[23] C. Cachin, S. Micali, and M. Stadler, "Computationally private information retrieval with polylogarithmic communication," in *Proceedings of Advances in Cryptology: International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT '99)*, vol. 1592, pp. 402–414, 1999.

[24] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan, "Private information retrieval," in *Proceedings of the Annual Symposium on Foundations of Computer Science*, pp. 41–50, October 1995.

[25] S. Yekhanin, "New locally decodable codes and private information retrieval schemes," Tech. Rep. 127, Electronic Colloquium on Computational Complexity, 2006.

[26] E. Kushilevitz and R. Ostrovsky, "Replication is not needed: single database, computationally-private information retrieval," in *Proceedings of the Annual Symposium on Foundations of Computer Science*, pp. 364–373, Miami Beach, Fla, USA, 1997.

[27] R. Cramer and I. Damgaard, "Secure distributed linear algebra in constant number of rounds," in *Proceedings of the 21st Annual IACR (CRYPTO '01)*, vol. 2139 of *Lecture Notes in Computer Science*, pp. 119–136, Santa Barbara, Calif, USA, August 2001.

[28] R. Sion and B. Carbunar, "On the computational practicality of prive information retrieval," in *Proceedings of the 14th ISOC Network and Distributed Systems Security Symposium*, San Diego, Calif, USA, February-March 2007.

[29] H. Lipmaa, "Oblivious Transfer or Private Information Retrieval," University College London, http://www.adastral.ucl.ac.uk/~helger/crypto/link/protocols/oblivious.php.

[30] K. Liu, "Privacy Preserving Data Mining Bibliography," University of Maryland, Baltimore County, http://www.csee.umbc.edu/~kunliu1/research/privacy_review.html.