CRYPTOGRAPHIC ALGORITHMS FOR PRIVACY-PRESERVING ONLINE APPLICATIONS

RUINIAN LI
Computer Science Department
Bowling Green State University
Bowling Green, Ohio 43401, USA

YINHAO XIAO AND CHENG ZHANG
Computer Science Department
George Washington University
2121 I St NW, Washington, DC 20052, USA

TIANYI SONG
Computer Science Department
Bowling Green State University
Bowling Green, Ohio 43401, USA

CHUNQIANG HU*
School of Big Data and Software Engineering
Chongqing University
Chongqing, China

(Communicated by Zhipeng Cai)

Abstract. Privacy in online applications has drawn tremendous attention in recent years. With the development of cloud-based applications, protecting users’ privacy while guaranteeing the expected service from the server has become a significant issue. This paper surveyed the most popular cryptographic algorithms in privacy-preserving online applications to provide a tutorial-like introduction to researchers in this area. Particularly, this paper focuses on introduction to homomorphic encryption, secret sharing, secure multi-party computation and zero-knowledge proof.

1. Introduction. Cloud computing [60] and big data [45] technology have been developed rapidly in the last decade, and privacy protection in online applications has become a more critical issue than ever before. Nowadays users are becoming more aware of their privacy issues when utilizing online service, and great efforts are made by researchers in building secure and robust applications with privacy protection. In this paper, we surveyed recent research in privacy-preserving online applications, and introduce the most popular and powerful cryptographic mechanisms in building practical applications.

Fig. 1 illustrates a general online application model, and this is also a structure for generic cloud computing, where the users outsource personal data to a cloud server. In this model, users send their personal data to the server who performs

2010 Mathematics Subject Classification. Primary: 94A60.
Key words and phrases. Security and privacy, homomorphic encryption, secret sharing, multi-party computing, zero-knowledge proof.
* Corresponding author: Chunqiang Hu.
computations and returns results to the users. The users and the server could be one-to-one, meaning that each user interacts with the server independently, or many-to-one, meaning that a group of users interact with the server together. Under some conditions, it is also possible to have multiple servers working together in the cloud. This cloud-based structure introduces serious privacy issues, because the users need to submit their personal data to the server, which exposes private information of the users. Therefore, some carefully-designed mechanisms are needed to protect users’ privacy.

There are countless online applications that adopt the structure in Fig. 1[85, 39, 81, 59, 5, 4], and each application has its own feature. For example, in an electronic voting system, voter’s votes should be kept secret such that the privacy of voters can be maintained; in a gene-testing application, not only the patient’s genome data should be kept secret, but also the doctors’ patent gene probe should be kept secret from the server. Nevertheless, a suitable cryptographic mechanism will solve these privacy issues with careful adjustments according to a specified application. In this paper we focus on some of the most popular and powerful cryptographic algorithms: homomorphic encryption, secret sharing, secure multi-party computation and zero-knowledge proof.

The concept of homomorphic encryption was firstly proposed in 1978 [72] and it raised lots of attention in the past 30 years. Homomorphic encryption enables the server to compute on the encrypted data in order to protect the privacy of users. A simple example is illustrated as follows. A user would like to compute a function $f$ of his personal data, which needs lots of computation power. Therefore, the user would like to outsource this task to a server. To do this, this user encrypts his data using homomorphic encryption, and sends the ciphertext to the server. The server will do some computations on the ciphertext, and return the results (also a ciphertext) to the user. Once the user decrypts the results from the server, he will obtain the function of his data.

Homomorphic encryption includes partially homomorphic encryption and fully homomorphic encryption. Partially homomorphic encryption was known for many
years and there are lots of public key cryptosystems that are partially homomorphic, such as unpadded RSA [73], Elgamal [25], and Paillier’s scheme [65]. These algorithms support either additive or multiplicative operations, and are not suitable for applications that need mixed operations such as signal processing. Researchers have been focused on fully homomorphic encryption in recent years, and efficiency has been largely improved from original lattice-based encryption to current learning-with-errors based encryption. However, fully homomorphic encryption is still not practical in real applications due to its huge computational cost. Partially homomorphic encryption is more efficient than fully homomorphic encryption and is well studied for real applications, though efficiency is still a critical factor in the real deployment.

Secret sharing is another significant primitive in building cryptographic mechanisms. Introduced in 1979 by Shamir [75], secret sharing has been developed in the later twenty years [10, 11, 17]. Secret sharing enables a secret to be shared by a group of users, who can reconstruct the secret together. Take an voting scheme for example, the final computation usually needs multiple administrators to process at the same time, thus secret sharing protocol is in need such that a secret key to decrypt the result could be split to multiple administrators and reconstructed when all the administrators are at present. Secret sharing is also a foundation of secure multi-party computation (MPC), which is useful in many online applications such as online lottery and auctions. While useful in many cloud-based applications, secrets sharing has its limitations. For applications with a large group of users sharing a secret, tremendous inter-communications among these users are needed, rendering huge communication cost. Thus communication cost is one key factor while considering applying secret sharing.

Zero-knowledge proof (ZKP) was proposed by Goldwasser, Micali and Rackoff in 1989 [35]. Zero-knowledge proof is used to build authentication systems where a user could be authenticated, but the credentials of the users will not be leaked to the authenticator. While this concept is attracting, there are no designs for a general zero-knowledge proof protocol. Based on different applications and requirements, there could be various designs. ZKP includes interactive ZKP and non-interactive ZKP; the former requires the participants to interact with each other while the latter does not. We give detailed explanations of these two types of ZKP in our paper.

In the following sections, we give detailed discussions of homomorphic encryption, secret sharing, secure multi-party computation and zero-knowledge proof. We also give discussions on how to apply these mechanisms in real applications.

2. Homomorphic encryption. Homomorphic encryption allows computations on ciphertext of a message, therefore the computation party cannot see the content of the message [63]. Most state-of-art homomorphic encryption schemes are public key cryptosystems [65, 25, 34, 73], and there are very few symmetric homomorphic encryption schemes [76]. Homomorphic encryption can be defined in the following way:

**Definition 2.1.** (Homomorphic Encryption) An encryption algorithm $E$ is homomorphic if

$$E(m_1 \star m_2) = E(m_1) \star E(m_2), \forall m_1, m_2 \in \mathbb{M},$$

where $\mathbb{M}$ is the set of plaintext, and $\star$ presents either additive operation or multiplicative operation.
Homomorphic encryption is characterized by four functions: Gen, Enc, Dec and Eval. Take a public key cryptosystem for example, a pair of keys pk and sk are generated by Gen (pk for encryption Enc and sk for decryption Dec separately). Eval takes a group of ciphertext and outputs a ciphertext that corresponds to a functioned plaintext. More specifically, Eval takes a group of ciphertext \((c_1, c_2, \cdots, c_n) \in \mathbb{C}\), where \((c_1, c_2, \cdots, c_n)\) are ciphertext from \(m_1, m_2, \cdots, m_n \in \mathbb{M}\), and outputs a ciphertext that corresponds to \(f(m_1, m_2, \cdots, m_n)\). Depending on the operations on the ciphertext, homomorphic can be classified as partially homomorphic encryption (FHE) and fully homomorphic encryption (PHE).

2.1. Partially homomorphic encryption. Partially homomorphic encryption supports only one kind of operation on the ciphertext, either additive or multiplicative operation. A homomorphic encryption scheme is said to be additively homomorphic if it supports additive operation, and multiplicatively homomorphic if it supports multiplicative operation. There are many designs of partially homomorphic encryptions, and we give examples of the most widely used homomorphic encryption as follows.

2.1.1. Multiplicative Homomorphic Encryption.

1. RSA

   We refer to unpadded RSA, which is also called plain RSA in this case. The detailed steps are listed as follows.
   (a) **Gen**: choose two large prime \(p\) and \(q\), and compute integer \(N = pq\). Choose an integer \(e\) such that \(gcd(e, \phi(N)) = 1\), where \(\phi(n) = (p-1)(q-1)\), and compute \(d = e^{-1} \mod \phi(n)\).
   (b) **Enc**: \(c = m^e \mod N\).
   (c) **Dec**: \(m = c^d \mod N\).
   (d) **Homomorphic Property**: 
   \[
   E(m_1) \cdot E(m_2) = (m_1^e \mod N) \cdot (m_2^e \mod N) \\
   = (m_1 \cdot m_2)^e \mod N \\
   = E(m_1 \cdot m_2)
   \]

2. **ElGamal Encryption**

   (a) **Gen**: on input \(1^n\) run \(G(1^n)\) to obtain \((G, q, g)\). Choose uniformly \(y \in \mathbb{Z}_q\) and compute \(h = g^y\). The public key is \((G, q, g, h)\), and the private key is \((G, q, g, x)\).
   (b) **Enc**: choose a uniform \(y \in \mathbb{Z}_q\), and output the ciphertext \(c = \langle g^y, h^y \cdot m \rangle\).
   (c) **Dec**: \(m = c_2/c_1^y = (h^y \cdot m)/(g^y)^y = g^{-y} \cdot m / g^{-y}\).
   (d) **Homomorphic Property**: 
   \[
   E(m_1) \cdot E(m_2) = (g^{y_1}, h^{y_1} \cdot m_1) \cdot (g^{y_2}, h^{y_2} \cdot m_2) \\
   = (g^{y_1+y_2}, m_1 \cdot m_2 \cdot h^{y_1+y_2}) \\
   = E(m_1 \cdot m_2)
   \]

2.1.2. Additive Homomorphic Encryption.

1. **Paillier Encryption**

   (a) **Gen**: choose two large primes \(p\) and \(q\) such that \(gcd(pq, (p-1), (q-1)) = 1\). Compute \(N = pq\). The public key is \(N\), and private key is \(\phi(N)\).
Table 1. Comparison of Homomorphic Schemes

| Scheme                                                                 | Additive-Homo | Multi-Homo | Full-Homo |
|-----------------------------------------------------------------------|--------------|------------|-----------|
| GM (Goldwasser) and Micali 1982 [34]                                  | ✔            |            |           |
| Exponential ElGamal [44]                                              | ✔            |            |           |
| Benaloh 1994 [9]                                                      | ✔            |            |           |
| NS (Naccache and Stern) 1998 [62]                                     | ✔            |            |           |
| OU (Okamoto and Uchiyama) 1998 [64]                                  | ✔            |            |           |
| Paillier 1999 [65]                                                    | ✔            |            |           |
| DJ (Damgad and Jurik) 2001 [24]                                      | ✔            |            |           |
| KTX (Kawachi, Tanaka and Xagawa) 2007 [46]                            | ✔            |            |           |
| RSA 1978 [73]                                                         | ✔            |            |           |
| Elgamal 1985 [25]                                                     | ✔            |            |           |
| Gentry 2009 [30]                                                      | ✔            |            |           |
| GH (Gentry and Halevi) 2011 [31]                                      | ✔            |            |           |
| Coron 2011 [21]                                                       | ✔            |            |           |
| BGV (Brakerski, Gentry and Vaikuntanathan) 2011 [82]                 | ✔            |            |           |
| LTV (Lopez-Alt, Tromer and Vaikuntanathan) 2012 [57]                  | ✔            |            |           |
| JFV (Junfeng Fan, Frederik and Vercauteren) 2012 [27]                 | ✔            |            |           |
| GSW (Gentry-Sahai-Waters) 2013 [32]                                   | ✔            |            |           |
| Gorti’s EHC (Enhanced homomorphic Cryptosystem) 2013 [70]             | ✔            |            |           |

(b) **Enc**: choose a uniform $r \in \mathbb{Z}_N^*$, and output ciphertext $c = (1 + N)^m \cdot r^N \mod N^2$.

(c) **Dec**: $m = (c^{\phi(N)} \mod N^2 - 1)/N \cdot \phi(N)^{-1} \mod N$.

(d) **Homomorphic Property**:

$$E(m_1) \cdot E(m_2) = (1 + N)^{m_1} \cdot r_1^N \mod N^2 \cdot (1 + N)^{m_2} \cdot r_2^N \mod N^2$$

$$= (1 + N)^{m_1 + m_2} \cdot (r_1 \cdot r_2)^N \mod N^2$$

$$= E(m_1 + m_2)$$

2. **Exponential ElGamal Encryption**

ElGamal encryption is originally multiplicative homomorphic, but could be changed to be additive homomorphic after some simple modifications. The idea is to encrypt $g^m$ instead of encrypting the message $m$ itself. This version
of ElGamal encryption is called exponential ElGamal. The difficulty of running exponential ElGamal is that it requires the computation authority to solve the discrete logarithm problem (DLP) (see Definition 2.2). Therefore, the algorithm usually works when \( m \) is small or the computation authority has a rough value of \( m \) and he could brute-force the value of \( m \). Although with limitations, exponential ElGamal has useful applications and there are some interesting designs in the smart grid systems[18].

**Definition 2.2.** (DLP in \( G \)) The computational DLP in a multiplicative group \( G \) with generator \( g \) is defined as follows: for an element \( Q \) in group \( G \), where \( Q = g^\alpha \), compute \( \alpha \) from \( Q \).

**2.2. Fully homomorphic encryption.** Fully Homomorphic encryption (FHE) has limited applications due to its high computational cost [7]. However, the feature of FHE attracts lots of researchers as it allows arbitrary computations. The early research on FHE is focused on lattic-based schemes [31, 77] integer-based schemes [80]. Current research on fully homomorphic encryption is mainly based on learning-with-errors (LWE) [71, 16] and ring-learning-with-errors (RLWE) [16, 15]. Optimization is still the biggest challenge in FHE and the huge costs of memory size hinders the development of FHE applications.

Tab. 1 presents the state-of-art homomorphic encryption schemes. The readers could explore each algorithm for details. Also, there are some good surveys that readers could refer to for extensive information [3, 61, 29].

**3. Secret sharing.** Secret Sharing [10] is a critical cryptographic primitive where a dealer distributes shares of a secret to parties such that a subset of the parties are able to reconstruct the secret. Most secret sharing schemes in use are threshold schemes, and a typical \((t, n)\)-threshold secret sharing scheme is defined as follows:

**Definition 3.1.** (Threshold Secret Sharing) A dealer holds a secret \( s \) to be shared to \( N \) users, and each user gets a share \( s_i \). A distribution scheme is said to be \((t, n)\)-threshold secret sharing scheme if the following requirements hold:

1. For any subset of \( T \) users where \( T \geq t \), secret \( s \) could be reconstructed from \( s_i \) of these \( T \) users.
2. For any subset of \( T \) users where \( T < t \), secret \( s \) could not be reconstructed from \( s_i \) of these \( T \) users.

We give descriptions of some mostly used secret sharing schemes here. The first two schemes we present, Sharmir’s and Blakley’s schemes, are threshold secrets sharing schemes, while the others are regular secret sharing schemes and there are no settings for threshold.

1. **Shamir’s Scheme** [75]: Shamir’s scheme is based on a well-known fact that any nonzero, degree-\( t \) polynomial over a field has most \( t \) roots. The construction is as follows:
   (a) **Pre-Construction:** Given a secret \( s \) from a finite field \( \mathbb{F} \), an integer \( t \) where \( 1 \leq t < n \).
   
   (b) **Sharing:** The dealer uniformly chooses \( a_1, a_2, \cdots, a_t \) from \( \mathbb{F} \), and defines a polynomial \( P(x) = s + \sum_{i=1}^{t} a_i X^i \). Therefore, each user’s share is \( s_i = P(x_i) \).
   
   (c) **Reconstruction:** \( t \) users are able to reconstruct the secret \( s \) by computing the unique degree-\( (t - 1) \) polynomial \( p' \) for which \( p'(x_{i_j}) = s_i \).
2. **Blakley’s Scheme**[12]: Blakley’s scheme is based on the fact that any \( n \) non-parallel \((n-1)\)-dimensional hyperplanes intersect at a specific point. Fig. 2 illustrates Blakley’s scheme in three dimensions: A secret \( x \) is encoded with all the three planes; only when the three planes intersects, the secrets can be reconstructed. The construction can be illustrated as follows:

(a) **Pre-Construction**: The dealer creates a point \( P(x_0, y_0, z_0) \) and let \( x_0 \) be the secret.

(b) **Sharing**: the dealer picks \( a \) and \( b \) randomly, and a large prime \( p \), and set 
\[ c \equiv z_0 - ax_0 - by_0 \pmod{p}, \]
then the plan is \( z = ax + by + c \). Each user gets a share of the secret.

(c) **Reconstruction**: 
\[ a_i x + b_i y - z \equiv -c_i \pmod{p} \]
will yield to a matrix equation:
\[
\begin{pmatrix}
a_1 & b_1 & -1 \\
a_2 & b_2 & -1 \\
a_3 & b_3 & -1
\end{pmatrix}
\begin{pmatrix}
x_0 \\
x_1 \\
x_2
\end{pmatrix}
\equiv
\begin{pmatrix}
-c_1 \\
-c_2 \\
-c_3
\end{pmatrix}
\pmod{p}
\]

Solution can be found to reveal the value of \( x_0 \).

![Figure 2. Blakley’s Secret Share Scheme](image)

Blakley’s protocol requires each user to keep a share of a hyper-plane, and each hyper-plane is actually a set of many different points. This indicates that, in Blakley’s protocol, each user has to keep a message space that is far greater than the secret itself. In Shamir’s scheme, however, each user only needs to keep a record of one point, which is the same size of the secret. Therefore, Shamir’s protocol is much more space-efficient than the Blakley’s scheme.

3. **Secret Splitting**

Another type of secret sharing is secret splitting, where a secret \( k \) is split to \( n \) people and has to be reconstructed with these \( n \) people. Different from threshold secret sharing, secret splitting requires a fixed number of users to reconstruct the secret. A basic secret splitting protocol involves a one-time-pad implementation and takes advantage of bit-wise exclusive-or operations.

More specifically, for a sequence of bits, the exclusive-or result is determined by all the bits. A simple example could be as follows: the dealer selects a secret key \( k = 1010 \) and splits the key into 3 shares, e.g. \( (0, 1, 0, 0) \), \( (1, 0, 0, 0) \) and \( (0, 1, 1, 0) \). Each user obtains one share and could reconstruct the secret if they are all at present:
The major drawback of secret splitting is that the number of users who share the secret is fixed. If a new user enters the set, the dealer has to recreate the shares and distribute the new shares to all users. This makes it very inconvenient for the dealer to manage the secret shares.

4. Ito, Saito, and Nishizeki’s scheme [43]: Ito, Saito, and Nishizeki’s (ISN) scheme is a special type of secret splitting, and it allows any number of users to reconstruct the secrets. ISN is based the computation of exclusive-or operations.

The construction is as follows:
(a) **Pre-Construction:** The dealer shares a secret \( k \in (0, 1) \). An authorized set that could reconstruct the secret is presented as \( s_1, s_2, s_3, \ldots, s_n \).
(b) **Sharing:** The dealer chooses \( n - 1 \) random bits \( (r_1, r_2, r_3, \ldots, r_{n-1}) \), computes \( r_n = k \oplus r_1 \oplus r_2 \oplus \cdots \oplus r_{n-1} \), and gives \( s_i \) the bit of \( r_i \).
(c) **Reconstruction:** \( k = r_1 \oplus r_2 \oplus \cdots \oplus r_{n-1} \oplus r_n \).

Note that the dealer needs to create many sets for authorized groups, and the same user could appear in any of the authorized groups. This indicates that the number of bits that a user gets is equal to the number of authorized sets that contains this user. This makes the scheme highly inefficient if the number of necessary authorized set becomes large. For example, suppose there is 1 dealer and 5 users, and the dealer would like the following three sets of users to reconstruct the secret: \{user1, user2, user3\}, \{user1, user3, user4, user5\} and \{user1, user2, user5\}. For these three sets of users, the dealer has to create three sets of shared secrets; \{r1, r2, r3\}, \{r1, r3, r4, r5\} and \{r1, r2, r5\}. In this case, if one user appears in \( n \) such sets, then he needs to remember \( n \) secret shares. When the users’ number becomes large, it becomes difficult for both the user and the dealer to manage the secret shares.

3.1. **Verifiable Secret sharing.** The secret sharing scheme introduced above are vulnerable to two attacks: 1). The dealer dishonestly gives inconsistent shares to users. 2). A malicious user presents a wrong share to construct the secret \( s \). To prevent these two attacks, a secret sharing scheme need to be improved to detect a dishonest dealer or user. Here we introduce verifiable secret sharing (VSS) [68, 79, 66]. The goal of VSS is to ensure that there is a well-defined secret that can be reconstructed even with a corrupted dealer or some dishonest users. A simple example is the algorithm introduced by Feldman [28], which is based on the Shamir’s secret sharing. The basic construction is the same as in Shamir’s scheme, but the dealer also distributes commitment to the coefficient of the polynomial \( P(x) \):

\[
A_0 = g^{a_0}, A_1 = g^{a_1}, A_2 = g^{a_2}, \ldots A_{t-1} = g^{a_{t-1}}
\]

With this construction, any user \( P_i \) is able to verify his share by computing:

\[
g^{s_i} \equiv \prod_{j=0}^{t-1} (A_j)^{i_j}.
\]
Note that the above equation holds if the dealer is honest:

\[ \prod_{j=0}^{t-1} (A_j)^{i_j} = \prod_{j=0}^{t-1} (g^{a_j})^{i_j} = g^{\sum_{j=0}^{t-1} a_j \cdot i_j} = g^{a_i}. \] (6)

The security of this scheme is based on the hardness of solving the discrete-logarithm problem. More specifically, an adversary has negligible probability to solve the following computational discrete logarithm problem (DLP) within any polynomial time. See Definition 2.2.

4. **Secure multi-party computation.** Secure Multi-party computation or multi-party computation (MPC) is a cryptographic primitive in which a group of entities are able to perform some computations without exposing individual's private information. Since the protocol needs the participants to work cooperatively without the supervision of a trusted third party, it is necessary to assume that the participants are not malicious. Therefore MPC protocols are commonly designed in an honest-but-curious model, in which the participants will try to infer other entities’ private information, but will follow the protocol honestly. A MPC protocol is defined as follows:

**Definition 4.1.** (MPC) A group of \( n \) parties \( \{p_1, p_2, \cdots , p_n\} \) would like to compute some function \( f \) based on each individual’s private data \( x_i \). A protocol \( P \) is said to be MPC protocol if it meets the two requirements after the execution:

1. **Correctness:** The execution of the protocol outputs \( f(x_1, x_2, \cdots , x_n) \).
2. **Privacy:** The execution of the protocol does not leak any private information of \( x_i \) to other parties.

A MPC is said to be *fair* if all parties can obtain the final results at the same time. Based on the number of participants, MPC protocols are classified as two-party protocol and multi-party protocol.

1. **Two-party Protocol**

   Two party protocol (2PC) was first proposed as the Millionair’s Problem: Two millionaires would like to know who is richer, but no one would like the other entity to know his property, then how to solve this problem? This problem was generalized in 1986 by Andrew Yao [83, 84] who proposed garbled circuits as a solution to 2PC. Garbled circuits takes one participant as circuits creator, and another participant as circuits evaluator. The truth table of the circuits is obscured by the creator and the execution of the circuits is run by the circuits evaluator. The evaluator needs to send the obscured result to the creator, who is able to map the obscured value to the final result. We give a detailed presentation of garbled circuits below.

   Fig. 3 illustrates the structure of garbled circuits. Suppose Alice and Bob are two participants of the protocol: Alice is the circuit creator, and Bob is the circuit evaluator. To evaluate the circuit, Alice inputs his value on the left side of each gate, while Bob inputs his value on the right side of each gate. In a non-garbled circuit, either “0” or “1” would be used as input; in garbled circuits, the inputs of “0” and “1” have been garbled and there is a mapping from each “0” or “1” to a pseudo-random looking string; to evaluate the circuit, the evaluator has to input the pseudo-random looking strings that represent either “0s” or “1s”. The detailed steps of evaluating garbled circuits are as follows.
To start the protocol, Alice as the circuit creator, creates the circuits for computing a specified function $f$, and maps “0s” and “1s” to pseudo-random looking strings, as shown in Fig. 4. For example, for a simple AND gate, Alice’s input for “0” is mapped to “$k_0 x$”, and Alice’s input for “1” is mapped to “$k_1 x$”. In the same way, Bob’s input for “0” is mapped to “$k_0 y$”, and Bob’s input for “1” is mapped to “$k_1 y$”. Therefore, to input “0” for Alice, the circuit evaluator needs to input “$k_0 x$” instead of “0”. The output of the AND gate is also garbled, and the output is produced according to the truth table, e.g, “$k_0 z$” is mapped to “0” and “$k_1 z$” is mapped to 1.

Alice then encrypt the outputs of the gate twice using the pseudo-random string as the symmetric encryption key, and obtains four values: $\alpha, \beta, \theta,$ and $\gamma$. Alice then sends the garbled circuits along with the four values to Bob. Note that Bob is not aware of how to evaluate the circuit at this moment, because the mapping is hidden from him.

After receiving the gabled circuit, Bob starts to evaluate the circuits. To do this, Bob has to get the mapping from Alice. Specifically, Bob needs two inputs: Alice’s input and his own input. To get Alice’s input is very simple, as he could just ask Alice to send him the garbled value of Alice. For example, if Alice would like to input “0”, she could send “$k_0 x$” to
Bob, and Bob will not be aware of the true value of “$k_0x$” is “0” or “1”. To get Bob’s input, Bob needs to get the mapping from Alice without letting Alice know what Bob gets from her. To do this, Bob has to run an Oblivious transfer Protocol. A simple 1-out-of-n oblivious transfer protocol is defined in the following way:

**Definition 4.2.** A sender inputs a string $x_0, x_1, \cdots, x_n$, and a receiver inputs one bit $\sigma \in (0, 1, \cdots, n)$. After the execution of the protocol, the receiver outputs $x_\sigma$ while the sender has no output.

Through the oblivious transfer, Bob could successfully gets his inputs from Alice. Take one gate for example, if Bob would like to input “1”, he could run oblivious transfer and obtains “$k_1y$” from Alice, while Alice does not know Bob asks for the mapping of “1” or “0”. After that, Bob has both his and Alice’s input to evaluate a gate. Bob needs to run oblivious transfer for each gate that he has an input, and he is able to get all the necessary inputs for each gate. Then Bob is able to evaluate the whole circuit.

(d) For each gate, Bob would get two inputs from Alice, e.g. “$k_0x$” and “$k_1y$”, and he tries to decrypt $\alpha, \beta, \theta$ and $\gamma$ using “$k_0x$” and “$k_1y$” as the keys. If “$k_0z$” and “$k_1z$” are carefully encrypted with padding, e.g. they are encrypted after padded with a sequence of “0s”, then the decryption with correct keys would be rather obvious to Bob. Hence, Bob is able to know that “$k_0x$” and “$k_1y$” are the right keys for “$\beta$”, not for “$\alpha$”, “$\theta$”, or “$\gamma$”. Hence, Bob successfully obtains “$k_0z$” and uses it as the input of the next gate.

(e) Bob will continue the process until he reaches the output wires of the entire circuit. Then Bob will tell Alice the garbled output value, and Alice will tell Bob if it represents 1 or 0.

Yao’s garbled circuit protocol is quite efficient and no private information is leaked. However, it only supports two parties, and the protocol is not fair for the participants: The circuits creator always knows the final result before the circuits evaluator, and he could maliciously choose not to tell the true result to the evaluator.

2. Multi-party Protocol

A multi-party protocol (MPC) MPCs are generally built in two ways: garbled circuits and secret sharing. Oblivious transfer, homomorphic encryption, and zero-knowledge proof and are major tools in implementing the MPC protocols. We have showed how to build a verifiable secret sharing scheme based on Shamir’s scheme, which is an example to build multi-party computation protocol. One interesting research building MPC protocol comes from Andrychowicz et al. [6], who proposed to utilize bitcoin as a platform for multi-party computation. The authors illustrated how to utilize bitcoin’s structure to build a secure lottery scheme and incentivize users to take part in computations without the involvement of a trusted third party. This work is distinctively from other work focusing on secret sharing, and opens a new research direction towards building secure MPC protocols with cryptocurrency.

Tab. 2 presents the implementations of MPC. The projects of these MPC could be found online. Readers could refer to each protocol for details.
Table 2. Multi-party Computation Implementations

| Scheme         | Feature                        | Party          |
|----------------|-------------------------------|----------------|
| FairPlay [58]  | Boolean Circuits              | Two-Party      |
| SPDZ [23]      | Arithmetic Circuits           | Two-Party      |
| MASCOT [47]    | Arithmetic Circuits           | Two-Party      |
| Tasty [40]     | Boolean & Arithmetic Circuits | Two-Party      |
| Sharemind [14] | Boolean Circuits              | Three-Party    |
| FairPlayMP [8] | Boolean Circuits              | Two or More    |
| VIFF [22]      | Arithmetic Circuits           | Two or More    |

5. **Zero-knowledge proof.** A zero-knowledge proof protocol (ZKP) [26] is a protocol with proof of statement that reveals nothing but the veracity of the statement. In a ZKP, there is a prover and a verifier, where the prover needs to prove to the verifier that he knows some secret without exposing the secret. Generally, ZKP are classified as Interactive ZKP [35] and Non-interactive ZKP [69].

1. Interactive ZKP

Interactive ZKP requires interactions between the verifier and prover. An interactive input from the verifier is called a challenge. Through responses to the challenges, the prover could prove to the verifier that he knows a secret, without exposing the secret to the verifier. A classic example of ZKP is as shown in Fig. 5. In this scenario, there is a door inside the cave, and Peggy (the woman) claims that she knows a magic word to open the door inside the cave. Now Peggy would like to prove to Victor (the man) that she indeed knows the magic word without telling Peggy the word. In this scenario Victor is the verifier and Peggy is the prover. A solution protocol will run in one way that after some interactions between Peggy and Victor, Victor will accept the fact that Peggy knows the magic word or not.

A simple protocol run in the following way [67]. Victor (Verifier) stands outside the cave while Peggy (Prover) enters the cave. Peggy has two ways to go, either A or B, while Victor does not see it. Then Victor will enter the cave and announces that he wants Peggy to show up at either A or B. Since the decisions of Victor and Peggy are independent, Peggy will only have fifty percent of chance to come to the right place followed by Peggy’s order, if she does not have the magic word to open the door. If Peggy does know the magic word, then she will definitely be able to show up at the right place. Suppose that this process is repeated for \( n \) times, then the chance that Peggy always shows up at the right place is \( 2^n \). When \( n \) becomes large, the probability becomes rather small. Therefore, if the process is repeated form many times while Peggy always successfully fulfills Victor’s requests, then it is very provable that Peggy really knows the magic word to open the door.

Based on different applications, there could be various designs of ZKPs. ZKPs are widely adopted in electronic voting systems where the voters need to prove to the voting authority that the encrypted votes are generated in the right form (e.g., contains either 1 or 0 and each candidate only appear once). Different designs are needed for different secret/credential, but generally a ZKP need to fulfill the following three requirements:
(a) Completeness: The protocol is complete if an honest verifier accept the prover’s statement when the statement is true with an overwhelming probability.

(b) Soundness: The protocol is sound if a dishonest prover is not able to convince an honest verifier with a non-negligible probability.

(c) Zero-knowledge: The protocol maintains the property of zero-knowledge if the execution of the protocol reveals no knowledge to the prover except that the statement is true or false.

A quick cryptographic example of zero-knowledge proof by Schnorr’s is introduced as follows.

(a) Public parameters: A cyclic group \( G \) of order \( q \), generator \( g \), and an element \( y \in G \).

(b) Participants: Prover: Peggy; Verifier: Victor.

(c) Statement from Victor: I know the value of \( x \) such that \( g^x = y \).

(d) Preparation of protocol: For each round, Peggy generates a random number \( r \), computes \( C = g^r \mod q \) and sends \( C \) to Victor.

(e) Challenge from Victor: For each round, Victor requests Peggy to disclose the value of \( r \), or the value of \( (x + r) \mod q \).

(f) Challenge validation: For each round, if \( r \) is requested and received from Peggy, Victor computes \( g^r \mod q \) to check if \( C \equiv g^r \mod q \); if \( (x + r) \mod q \) is requested and received from Peggy, Victor computes \( C \cdot y \equiv g^{x+r} \mod q \).

If after enough rounds of executions, the challenge validation is always successful, then Victor is convinced that Peggy knows the value of \( x \).

In this protocol, if Peggy knows the secret \( x \), he could definitely send the correct value of \( r \) or \( (x + r) \mod q \) to Victor. If Peggy does not know \( x \), he will be able to send the correct value of \( r \) or \( (x + r) \mod q \) with 50 percent of probability respectively. To understand this clearly, the two possible conditions are illustrated here if Peggy does not know the secret of \( x \): 1) In the preparation stage of the protocol, Peggy honestly sends the value of \( C = g^r \) to Victor, then he will not be able to know the value of \( x + r \) because he does not know \( x \). 2) In the preparation stage of the protocol, Peggy randomly picks a value \( r' \), computes \( C' = g^{r'} \cdot (g^x)^{-1} \), and sends \( C' \) to Peggy. In this way, if Victor requests the value of \( (x + r) \mod q \), Peggy could cheat Victor by sending the value of \( r' \) instead of \( (x + r) \mod q \), and pass the validation, as \( g^{r'} \cdot (g^x)^{-1} \cdot y = g^{r'} \mod q \). However, if Victor requests the value of \( r \), then Victor is not able to pass the validation.
(a) Completeness: Schnorr’s protocol is complete as the verifier is convinced that the prover has the secret $x$ such that $g^x = y$. The reason is that if the verifier does not have the secret $x$, then in each round he only has 50 percent of probability to pass the challenge validation. If the verifier repeats the challenge for many times, then the chance for the verifier to give the correct values is very low.

(b) Soundess: Schnorr’s protocol is sound because if the prover does not know $x$, he will not be able to convince the verifier in repeated challenges.

(c) Zero-knowledge: Schnorr’s protocol maintains the property of zero-knowledge because the verifier does not obtain any knowledge of the secret $x$.

2. Non-Interactive ZKP: Compared to interactive ZKP, there are no interactions between the verifier and the prover in the non-interactive ZKP. One general model of non-interactive ZKP consists of a prover, a verifier and a uniformly selected random string [13]. The string is selected by a trusted party who associate in constructing the protocol. If the prover and verifier share this common random string, the prover can non-interactively and yet zero-knowledge convince the verifier the validity of a theorem that he may discover. In this model, the only “interaction” is the transfer of a message from the prover to the verifier, and the message is left with the final decision of accept or decline.

Blum, Feldman and Micali were the first to propose non-interactive ZKP. It is proved by Oren et al. [33] that Non-interactive ZKP system only exists for Bounded-error Probabilistic (BPP) languages in the plain model. Non-interactive ZKP is useful in building cryptographic mechanisms, but there lacks practical applications as it is quite inefficient. Researchers have been making lots of effort improving efficiency of NIZK. In 2008, Groth et al. [38] shows how to obtain efficient NIZK proof by instantiating the proposed GS proof (Groth-Sahai) according to different application background. Two years later, Groth et al. makes further improvement based on GS proof framework to reduce computational cost of NIZK protocol [36, 37].

6. Applications. Cryptographic primitives are very useful in building real applications. Particularly, cloud-based applications benefits highly from the mechanisms we introduced in this paper. Recent development of internet of things (IoT) [78, 52, 54, 53, 20, 87, 56, 86] also demonstrates high demand of the deployment of cryptographic mechanisms. For example, in a smart grid system, the smart meters send fine-grained data to the server for data aggregation, and the fine-grained data expose the private information of the users, such as the daily routines. In order to protect user’s privacy while maintaining the service, cryptographic methods are powerful and effective methods as a solution. Security and privacy in social networks is another area that has attracted tremendous attention in recent years [50, 41, 19, 87, 39, 88]. In social network, each user has a personal profile that should be kept secret. However, some powerful and necessary functions in social networks, such as friend matching, expose users’ profile while computing the match of users’ intersts. Therefore, privacy-preserving methods are in need to protect user’s privacy while enabling popular features in social network.

We give a concrete example of Homomorphic encryption in an auction scheme here. In a privacy-preserving auction scheme [49, 48, 55, 42, 51], a group of $n$ bidders submit their bidding values $b_i$ to an auctioneer. The auctioneer needs to compute the sum of the bidding values without seeing each individual bidding value.
A generic protocol presented by Fig. 6 could be constructed on Paillier encryption could be as follows:

1. A public key \( N \) for Paillier encryption is created and publicized by an auctioneer.
2. Each bidder encrypts its bidding value \( b_i \) as \( c_i = \text{Enc}(b_i) = (1 + N)^{b_i} \cdot (r_i)^N \mod N^2 \), where \( r_i \in \mathbb{Z}_N^* \) is uniformly chosen.
3. The evaluation of \( c_i \) is computed as: \( c_{\text{sum}} = \prod_{i=1}^{n} c_i \mod N^2 \) by a trusted authority that is not colluding with the auctioneer.
4. \( c_i \) is given to the auctioneer who decrypts it to the sum of the bidder values: \( b_{\text{sum}} = \text{Dec}(c_{\text{sum}}) = \sum_{i=1}^{n} b_i \mod N \).

Through the above protocol, the auctioneer is only able to decrypt the final result without seeing each bidder’s bidding values. This prevent the auctioneer from manipulating the auction for his own good.

Another useful protocol in the auction protocol is the zero-knowledge proof. Note that all the bid values are encrypted, and there should be some way to check if the
Table 3. Recent Hot Research Topics in Privacy-Aware Computing

|                                      | Homomorphic Encryption | Secret Sharing/MPC | Zero-knowledge Proof |
|--------------------------------------|------------------------|--------------------|----------------------|
| Electronic Voting                    | ✓                      | ✓                  | ✓                    |
| Online Auction                       | ✓                      | ✓                  | ✓                    |
| Smart Grid                           | ✓                      | ✓                  | ✓                    |
| Gene Testing                         | ✓                      | ✓                  | ✓                    |
| Social network                       | ✓                      | ✓                  | ✓                    |
| Blockchain                           | ✓                      | ✓                  | ✓                    |

bid values are encrypted in a correct way. Here the auctioneer could act as the verifier, and the bidders act as the prover: the prover should prove to the verifier that the bid values are created and encrypted in a correct form.

We surveyed the recent research in privacy-preserving applications, and present the most cutting-edge research area that adopts the cryptographic primitives we introduced in Tab. 3.

7. Conclusion. In this paper, we surveyed the most popular cryptographic mechanisms for privacy-preserving applications. In particular, we give a tutorial-like introduction to homomorphic encryption, secret sharing, secure multi-party computation and zero-knowledge proof. We present the definitions, state-of-art work, technique details and applications. We also give examples of how to apply these cryptographic mechanisms, and we hope our work advances researchers in studying applied cryptography.

8. Acknowledgment. This research was partially supported by the National Natural Science Foundation of China (61702062), the Natural Science Foundation of Chongqing, China (cstc2018jcyjAX0334), and Overseas Returnees Support Program for Innovation and Entrepreneurship of Chongqing (cx2018015).

REFERENCES

[1] Secret sharing, 2018, https://en.wikipedia.org/wiki/Secret_sharing.
[2] Zero-knowledge proof, 2018, https://en.wikipedia.org/wiki/Zero-knowledge_proof.
[3] A. Acar, H. Aksu, A. S. Uluagac and M. Conti, A survey on homomorphic encryption schemes: Theory and implementation, ACM Computing Surveys (CSUR), 51 (2018), Article No. 79.
[4] A. Alhothaily, A. Alrawais, T. Song, B. Lin and X. Cheng, Quickcash: Secure transfer payment systems, Sensors, 17 (2017), 1376.
[5] A. Alhothaily, C. Hu, A. Alrawais, T. Song, X. Cheng and D. Chen, A secure and practical authentication scheme using personal devices, IEEE Access, 5 (2017), 11677–11687.
[6] M. Andrychowicz, S. Dziembowski, D. Malinowski and L. Mazurek, Secure multiparty computations on bitcoin, in Security and Privacy (SP), 2014 IEEE Symposium on, IEEE, 2014, 443–458.
[7] F. Armknecht, C. Boyd, C. Carr, K. Gijsteen, A. Jäischke, C. A. Reuter and M. Strand, A guide to fully homomorphic encryption., IACR Cryptology ePrint Archive, 2015 (2015), 1192.
[8] A. Ben-David, N. Nisan and B. Pinkas, Fairplaymp: A system for secure multi-party computation, in Proceedings of the 15th ACM Conference on Computer and Communications Security, ACM, 2008, 257–266.
[9] J. Benaloh, Dense probabilistic encryption, in Proceedings of the Workshop on Selected Areas of Cryptography, 1994, 120–128.
Cryptography Algorithms Online Applications

10. J. Benaloh and J. Leichter, Generalized secret sharing and monotone functions, in Proceedings on Advances in Cryptology, Springer-Verlag New York, Inc., 403 (1990), 27–35.
11. M. Bertilsson and I. Ingemarsson, A construction of practical secret sharing schemes using linear block codes, in International Workshop on the Theory and Application of Cryptographic Techniques, Springer, 1992, 67–79.
12. G. R. Blakley et al., Safeguarding cryptographic keys, in Proceedings of the National Computer Conference, 48 (1979), 313–317.
13. M. Blum, P. Feldman and S. Micali, Non-interactive zero-knowledge and its applications, in Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing, ACM, 1988, 103–112.
14. D. Bogdanov, S. Laur and J. Willemsen, Sharemind: A framework for fast privacy-preserving computations, in European Symposium on Research in Computer Security, Springer, 2008, 192–206.
15. Z. Brakerski and V. Vaikuntanathan, Fully homomorphic encryption from ring-lwe and security for key dependent messages, in Annual Cryptology Conference, Springer, 2011, 505–524.
16. Z. Brakerski and V. Vaikuntanathan, Efficient fully homomorphic encryption from (standard) lwe, SIAM Journal on Computing, 43 (2014), 831–871.
17. E. F. Brickell, Some ideal secret sharing schemes, in Workshop on the Theory and Application of Cryptographic Techniques, Springer, 434 (1990), 468–475.
18. N. Busom, R. Petric, F. Sebè, C. Sorge and M. Valls, Efficient smart metering based on homomorphic encryption, Computer Communications, 82 (2016), 95–101.
19. Z. Cai, Z. He, X. Guan and Y. Li, Collective data-sanitization for preventing sensitive information inference attacks in social networks, IEEE Transactions on Dependable and Secure Computing, 15 (2018), 577–590.
20. Z. Cai and X. Zheng, A private and efficient mechanism for data uploading in smart cyber-physical systems, IEEE Transactions on Network Science and Engineering, (2018), 1–1.
21. J.-S. Coron, D. Naccache and M. Tibouchi, Public key compression and modulus switching for fully homomorphic encryption over the integers, in Annual International Conference on the Theory and Applications of Cryptographic Techniques, Springer, 7237 (2012), 446–464.
22. I. Damgård, M. Geisler, M. Krøigaard and J. B. Nielsen, Asynchronous multiparty computation: Theory and implementation, in International Workshop on Public Key Cryptography, Springer, 5443 (2009), 160–179.
23. I. Damgård, V. Pastro, N. Smart and S. Zakarias, Multiparty computation from somewhat homomorphic encryption, in Advances in Cryptology–CRYPTO 2012, Springer, 7417 (2012), 643–662.
24. I. Damgard and M. Jurik, A generalisation, a simplification and some applications of Paillier’s probabilistic public-key system, in Proceedings of the 4th International Workshop on Practice and Theory in Public Key Cryptosystems, 2001, 119–136.
25. T. ElGamal, A public key cryptosystem and a signature scheme based on discrete logarithms, IEEE Transactions on Information Theory, 31 (1985), 469–472.
26. B. Ewanick, Zero knowledge proof, Google Scholar.
27. J. Fan and F. Vercauteren, Somewhat practical fully homomorphic encryption, IACR Cryptology ePrint Archive, 2012 (2012), 144.
28. P. Feldman, A practical scheme for non-interactive verifiable secret sharing, in Foundations of Computer Science, 1987., 28th Annual Symposium on, IEEE, 1987, 427–438.
29. C. Fontaine and F. Galand, A survey of homomorphic encryption for nonspecialists, EURASIP Journal on Information Security, 2007 (2007), 15.
30. C. Gentry, Fully homomorphic encryption using ideal lattices, Proceedings of the 41st Annual Acm Symposium on Symposium on Theory of Computing-stoc’09, (2009), 169–178.
31. C. Gentry and S. Halevi, Fully homomorphic encryption without squashing using depth-3 arithmetic circuits, in Foundations of Computer Science (FOCS), 2011 IEEE 52nd Annual Symposium on, IEEE, 2011, 107–116.
32. H. Gentry, A. Sahai and B. Waters, Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based, in Advances in Cryptology–CRYPTO 2012, Springer, 2013, 75–92.
33. O. Goldreich and Y. Oren, Definitions and properties of zero-knowledge proof systems, Journal of Cryptology, 7 (1994), 1–32.
[34] S. Goldwasser and S. Micali, Probabilistic encryption and how to play mental poker keeping secret all private information, in *Proceedings 14th ACM Symposium on the Theory of Computing*, vol. 4, 1982.

[35] S. Goldwasser, S. Micali and C. Rackoff, The knowledge complexity of interactive proof systems, *SIAM Journal on Computing*, 18 (1989), 186–208.

[36] J. Groth, Short pairing-based non-interactive zero-knowledge arguments, in *International Conference on the Theory and Application of Cryptology and Information Security*, Springer, 2010, 321–340.

[37] J. Groth, Efficient zero-knowledge arguments from two-tiered homomorphic commitments, in *International Conference on the Theory and Application of Cryptology and Information Security*, Springer, 2010, 451–448.

[38] J. Groth, R. Ostrovsky and A. Sahai, Perfect non-interactive zero knowledge for np, in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, Springer, 2006, 339–358.

[39] Z. He, Z. Cai and J. Yu, Latent-data privacy preserving with customized data utility for social network data, *IEEE Transactions on Vehicular Technology*, 67 (2018), 665–673.

[40] W. Henecka, A.-R. Sadeghi, T. Schneider, I. Wehrenberg et al., Tasty: Tool for automating secure two-party computations, in *Proceedings of the 17th ACM Conference on Computer and Communications Security*, ACM, 2010, 451–462.

[41] C. Hu, R. Li, W. Li, J. Yu, Z. Tian and R. Bie, Efficient privacy-preserving schemes for dot-product computation in mobile computing, in *Proceedings of the 1st ACM Workshop on Privacy-Aware Mobile Computing*, ACM, 2016, 51–59.

[42] C. Hu, R. Li, B. Mei, W. Li, A. Alrawais and R. Bie, Privacy-preserving combinatorial auction without an auctioneer, *EURASIP Journal on Wireless Communications and Networking*, 2018 (2018), 38.

[43] M. Ito, A. Saito and T. Nishizeki, Secret sharing scheme realizing general access structure, *Electronics and Communications in Japan (Part III: Fundamental Electronic Science)*, 72 (1989), 56–63.

[44] M. Jakobsson and A. Juels, Addition of el gamal plaintexts, in *International Conference on the Theory and Application of Cryptology and Information Security*, Springer, 1976 (2000), 346–358.

[45] S. John Walker, Big data: A revolution that will transform how we live, work, and think, *The Review of Marketing Communications*, 33 (2014), 181–183.

[46] A. Kawachi, K. Tanaka and K. Xagawa, Multi-bit cryptosystems based on lattice problems, in *International Workshop on Public Key Cryptography*, Springer, 4450 (2007), 315–329.

[47] M. Keller, E. Orsini and P. Scholl, Mascot: faster malicious arithmetic secure computation with oblivious transfer, in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ACM, 2016, 830–842.

[48] M. Larson, C. Hu, R. Li, W. Li and X. Cheng, Secure auctions without an auctioneer via verifiable secret sharing, in *Proceedings of the 2015 Workshop on Privacy-Aware Mobile Computing*, ACM, 2015, 1–6.

[49] M. Larson, R. Li, C. Hu, W. Li, X. Cheng and R. Bie, A bidder-oriented privacy-preserving VCG auction scheme, in *International Conference on Wireless Algorithms, Systems, and Applications*, Springer, 2015, 284–294.

[50] R. Li, H. Li, X. Cheng, X. Zhou, K. Li, S. Wang and R. Bie, Perturbation-based private profile matching in social networks, *IEEE Access*, 5 (2017), 19720–19732.

[51] R. Li, B. Mei, C. Hu, W. Li, M. Larson, X. Cheng and R. Bie, A cloud-based framework for verifiable privacy-preserving spectrum auction, *EURASIP Journal on Wireless Communications and Networking*.

[52] R. Li, T. Song, N. Capurso, J. Yu, J. Couture and X. Cheng, IoT applications on secure smart shopping system, *IEEE Internet of Things Journal*, 4 (2017), 1945–1954.

[53] R. Li, T. Song, B. Mei, H. Li, X. Cheng and L. Sun, Blockchain for large-scale internet of things data storage and protection, *IEEE Transactions on Services Computing*, (2018), 1–1.

[54] R. Li, C. Sturtivant, J. Yu and X. Cheng, A novel secure and efficient data aggregation scheme for IoT, *IEEE Internet of Things Journal*, (2018), 1–1.

[55] W. Li, M. Larson, C. Hu, R. Li, X. Cheng and R. Bie, Secure multi-unit sealed first-price auction mechanisms, *Security and Communication Networks*, 9 (2016), 3833–3843.
[56] Y. Liang, Z. Cai, J. Yu, Q. Han and Y. Li, Deep learning based inference of private information using embedded sensors in smart devices, *IEEE Network Magazine*, 32 (2018), 8–14.

[57] A. López-Alt, E. Tromer and V. Vaikuntanathan, On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption, in *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, ACM, 2012, 1219–1234.

[58] D. Malkhi, N. Nisan, B. Pinkas, Y. Sella et al., Fairplay-secure two-party computation system., in *USENIX Security Symposium*, vol. 4, San Diego, CA, USA, 2004, 9pp.

[59] B. Mei, Y. Xiao, R. Li, H. Li, X. Cheng and Y. Sun, Image and attribute based convolutional neural network inference attacks in social networks, *IEEE Transactions on Network Science and Engineering*, (2018), 1–1.

[60] P. Mell, T. Grance et al., The nist definition of cloud computing, *National Institute of Standards and Technology Special Publication 800-145*, (2011), 7pp.

[61] C. Moore, M. O’Neill, E. O’Sullivan, Y. Doroz and B. Sunar, Practical homomorphic encryption: A survey, in *Circuits and Systems (ISCAS), 2014 IEEE International Symposium on*, IEEE, 2014, 2792–2795.

[62] D. Naccache and J. Stern, A new public key cryptosystem based on higher residues, in *Proceedings of the 5th ACM Conference on Computer and Communications Security*, ACM, 1998, 59–66.

[63] M. Ogburn, C. Turner and P. Dahal, Homomorphic encryption, *Procedia Computer Science*, 20 (2013), 502–509.

[64] T. Okamoto and S. Uchiyama, A new public-key cryptosystem as secure as factoring, in *International Conference on the Theory and Applications of Cryptographic Techniques*, Springer, 1998, 308–318.

[65] P. Paillier, Public-key cryptosystems based on composite degree residuosity classes, in *International Conference on the Theory and Applications of Cryptographic Techniques*, Springer, 1999, 223–238.

[66] T. P. Pedersen, Non-interactive and information-theoretic secure verifiable secret sharing, in *Annual International Cryptology Conference*, Springer, 576 (1992), 129–140.

[67] J.-J. Quisquater, M. Quisquater, M. Quisquater, M. Quisquater, L. Guillou, M. A. Guillou, G. Guillou, A. Guillou, G. Guillou and S. Guillou, How to explain zero-knowledge protocols to your children, in *Conference on the Theory and Application of Cryptology*, Springer, 1989, 628–631.

[68] T. Rabin and M. Ben-Or, Verifiable secret sharing and multiparty protocols with honest majority, in *Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing*, ACM, 1989, 73–85.

[69] C. Rackoff and D. R. Simon, Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack, in *Annual International Cryptology Conference*, Springer, 1991, 433–444.

[70] G. V. S. Rao and G. Uma, An efficient secure message transmission in mobile ad hoc networks using enhanced homomorphic encryption scheme, *Global Journal of Computer Science and Technology*, 8 (2008), 1–10.

[71] O. Regev, On lattices, learning with errors, random linear codes, and cryptography, *Journal of the ACM (JACM)*, 56 (2009), Art. 34, 40 pp.

[72] R. L. Rivest, L. Adleman and M. L. Dertouzos, On data banks and privacy homomorphisms, *Foundations of Secure Computation*, 4 (1978), 169–179.

[73] R. L. Rivest, A. Shamir and L. Adleman, A method for obtaining digital signatures and public-key cryptosystems, *Communications of the ACM*, 21 (1978), 120–126.

[74] C.-P. Schnorr, Efficient signature generation by smart cards, *Journal of Cryptology*, 4 (1991), 161–174.

[75] A. Shamir, How to share a secret, *Communications of the ACM*, 22 (1979), 612–613.

[76] I. Sharma, Fully homomorphic encryption scheme with symmetric keys, preprint, arXiv:1310.2452.

[77] N. P. Smart and F. Vercauteren, Fully homomorphic encryption with relatively small key and ciphertext sizes, in *International Workshop on Public Key Cryptography*, Springer, 6056 (2010), 420–443.

[78] T. Song, R. Li, B. Mei, J. Yu, X. Xing and X. Cheng, A privacy preserving communication protocol for iot applications in smart homes, *IEEE Internet of Things Journal*, 4 (2017), 1844–1852.

[79] M. Stadler, Publicly verifiable secret sharing, in *International Conference on the Theory and Applications of Cryptographic Techniques*, Springer, 1996, 190–199.
[80] M. Van Dijk, C. Gentry, S. Halevi and V. Vaikuntanathan, Fully homomorphic encryption over the integers, in Annual International Conference on the Theory and Applications of Cryptographic Techniques, Springer, 6110 (2010), 24–43.

[81] J. Wang, Z. Cai, Y. Li, D. Yang, J. Li and H. Gao, Protecting query privacy with differentially private k-anonymity in location-based services, Personal and Ubiquitous Computing, 22 (2018), 453–469.

[82] M. Yagisawa, Fully homomorphic encryption without bootstrapping., IACR Cryptology ePrint Archive, 2015 (2015), 474.

[83] A. C. Yao, Protocols for secure computations, in Foundations of Computer Science, 1982. SFCS’82. 23rd Annual Symposium on, IEEE, 1982, 160–164.

[84] A. C.-C. Yao, How to generate and exchange secrets, in Foundations of Computer Science, 1986., 27th Annual Symposium on, IEEE, 1986, 162–167.

[85] X. Zheng, Z. Cai, J. Li and H. Gao, Location-privacy-aware review publication mechanism for local business service systems, in The 36th Annual IEEE International Conference on Computer Communications (INFOCOM 2017), 2017, 1–9.

[86] X. Zheng, Z. Cai and Y. Li, Data linkage in smart iot systems: A consideration from privacy perspective, IEEE Communications Magazine.

[87] X. Zheng, Z. Cai, J. Yu, C. Wang and Y. Li, Follow but no track: Privacy preserved profile publishing in cyber-physical social systems, IEEE Internet of Things Journal, 4 (2017), 1868–1878.

[88] X. Zheng, G. Luo and Z. Cai, A fair mechanism for private data publication in online social networks, IEEE Transactions on Network Science and Engineering, (2018), 1–1.

Received August 2018; revised September 2018.

E-mail address: lir@bgsu.edu
E-mail address: xyh3984@gwmail.gwu.edu
E-mail address: zhangchengcarl@gwmail.gwu.edu
E-mail address: tsong@bgsu.edu
E-mail address: chu@cqu.edu.cn