IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 18, 2021, accepted August 31, 2021, date of publication September 3, 2021, date of current version September 13, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3110055

Efficient Two-Party Integer Comparison With Block Vectorization Mechanism

THAI-HUNG NGUYEN, KOK-SENG WONG, (Member, IEEE), AND THOMAS OIKONOMOU
College of Engineering and Computer Science, VinUniversity, Hanoi 100000, Vietnam

Corresponding author: Kok-Seng Wong (wong.ks@vinuni.edu.vn)

This work was supported by the VinUniversity Seed Grant 2021.

ABSTRACT Private integer comparison has been an essential computation function for many applications, including online auction, credential identification, data mining, and joint bidding. In the setting of two-party computation, two parties with private inputs (x and y) want to jointly compare them without revealing the value of those inputs to others (also known as the Millionaires' problem) while the output should ensure correctness and preserve data privacy. The private inputs only can be revealed if they are equal, i.e., $x = y$. Many related works have been proposed to solve the integer comparison problem in various settings, focusing on different properties such as round and computation complexity. Most solutions decompose integers into bitwise representation and then securely evaluate the function in a Boolean circuit on encrypted bits. However, this type of solution is costly (especially for large integers) as each bit requires encryption and decryption. In this paper, we transform the private integer comparison into a block comparison problem. In particular, we employ a block vectorization mechanism to encode the private inputs into blocks. We show the security of our two-party protocol in the semi-honest model. Also, we implement the protocol to demonstrate its efficiency using block vectorization mechanism and homomorphic encryption. The experimental result proves that our proposed solution achieves high efficiency, particularly for large integer comparisons.

INDEX TERMS Private integer comparison, data privacy, secure two-party computation, semi-honest protocol.

I. INTRODUCTION

Secure two-party computation has been an important area of research in cryptography. The idea is to allow two parties to securely evaluate a function on their private inputs without leaking extra information to any party. This scenario was first studied and introduced by Yao in the millionaires' problem (often abstracted as "*Greater Than*" or *GT* problem) [1] where two individuals are allowed to compare their richness without revealing their wealth to each other. The *GT* problem plays an important role in a variety of privacy-preserving protocols, such as privacy-preserving data mining [2], [3], private e-voting systems [4], [5] and secure e-commerce [6].

Private integer comparison is fundamental to *GT* problem such that given two private integer inputs x and y, the protocol outputs true if $x > y$, false otherwise. When privacy is not a concern, the integer comparison problem becomes straightforward to implement. Each party can send their private inputs to a trusted third party (TTP) that functions as the central repository or data warehouse to perform the comparison. This is an ideal approach to support secure computations if the data being used is not sensitive information (e.g., sharing of project member's names, email addresses, and contact numbers). With privacy concerns, none of them should reveal their private data to any party, including the TTP (e.g., sales analysis, product costs, and stocks information). However, in practice, a TTP does not exist in the real world. In the two-party setting, secure integer comparison allows two mistrusting parties to jointly compare their private inputs without the presence of a trusted third party (TTP).

The usage of a two-party integer comparison is essential in many applications such as online auction, credential identification, data mining, and joint bidding. For example, in the sealed-bid auction system, a group of bidders simultaneously (or within an agreed period) submit their sealed-bid to the auctioneer without knowing the bidding price of the others [7]. In particular, the sealed-bid auctions require submitted bids to remain private to protect the privacy of the losing bidders. To ensure bid privacy, the auctioneer can utilize a private integer comparison protocol to filter those who submitted a bid lower than the minimum bidding price (assuming the price is correct to the nearest integer). Also,

The associate editor coordinating the review of this manuscript and approving it for publication was Sedat Akleylek.

the private integer comparison can be used to prevent collusion between bidders and auctioneer in public procurement and project tendering operations.

In general, this paper addresses two goals. First, we want to protect the privacy of the private inputs from two parties before and after the computation except when they are equal. This goal is essential in many applications to prevent other parties, e.g., competitors, from gaining an advantage. For example, the leakage of bidding price in sealed-bid auctions allows the attacker to use such information in future auctions and negotiations for similar products or services. To achieve this, we utilize homomorphic encryption to hide the private inputs in our protocol design. Second, we propose an efficient two-party protocol to compare the private inputs, especially for large integer comparisons. The existing works in the literature are not efficient for large integer comparison.

A. OUR CONTRIBUTIONS

Our contributions can be summarized as follows:

1) Motivated by the vectorization method proposed in [8], we transform the private integer comparison into a block comparison problem. We encode the private inputs into blocks to improve the protocol efficiency for large integers comparison.
2) We propose an efficient two-party integer comparison protocol in the semi-honest model. By using a block vectorization mechanism, our protocol significantly reduces the execution time of the protocol in [8] and achieves low communication and computational complexities for large integer inputs. The computational complexity of our protocol is $2(\sqrt{M}+2)\lg N$ modular multiplications, where M is the size of the original encoding vector.

B. PAPER ORGANIZATION

The rest of this paper is organized as follows. We discuss the related work in Section II and technical preliminaries are presented in Section III. We present our solution in Section IV, followed by security and performance analysis in Section V. We then provide our discussions and conclusions in Section VI.

II. RELATED WORK

Private integer comparison has been a popular research topic in cryptography. A large body of related works has been proposed to solve the integer comparison problem in various settings and focus on different properties such as round and computation complexity. In general, we can classify the existing works based on the employed techniques such as garbled circuits [9], oblivious transfers [10], [11], homomorphic encryptions [12], and secret sharing.

The garbled circuit is a cryptographic protocol that was first proposed to solve private integer comparison. The idea is to decompose integers into bitwise representation and then securely evaluate the function in a Boolean circuit [1]. Garbled circuit-based solutions are somehow inefficient because they require many rounds of interaction between the two parties to output the comparison result. As proved by Micali *et al.* in [13], there exists a secure solution for any function which can be represented as a combinatorial circuit. However, the circuit evaluation is somehow inefficient for many parties because the cost for large input can be very high.

In contrast to bitwise comparison, Carlton *et al.* [14] proposed a scheme that is based on a special RSA modulus to compare multiple input bits simultaneously within a single ciphertext. However, a plaintext equality test (PET) subprotocol is required to determine the comparison outcome and causes the increase of the computational cost. Later, Bourse *et al.* [12] improved the scheme in [14] by replacing PET subprotocol with a control value sent to one party, i.e., for Alice to determine the comparison result. However, the upper bound of the private inputs in this scheme is significantly small compared with the solution proposed by Carlton.

Homomorphic encryption (e.g., Paillier [15] and ElGamal [16]) is another essential approach that can be used to compare integers without revealing their actual values. The implementation of homomorphic encryption is straightforward. Hence, it can lower the overall communication costs as compared to the garbled circuit approach. However, it is typically less computationally efficient because of the additional cryptosystem used. Fishlin first introduced this approach in [17] to compare two integers using a Boolean circuit and semantically secure cryptosystem [18]. Lin and Tzeng [19] proposed a protocol for integers comparison by combining multiplicative homomorphism of the ElGamal encryption scheme with 0-encoding and 1-encoding. While their solution is efficient but only works when the bit numbers of x and y are the same, which is impractical in real-world applications. Blake and Kolesnikov [20] employed the additive homomorphism of the Paillier encryption scheme to construct a GT protocol for Boolean evaluation of bitwise encrypted values. However, these solutions cannot classify between $x = y$ and $x < y$. Various homomorphic encryption-based solutions have been proposed in the literature by using fully homomorphic [21], and somewhat practical fully homomorphic encryption [22]. However, these solutions are still inefficient and unpractical. Note that some works require encryption on both inputs and outputs [23], [24] while some only deal with encrypted inputs and allow the output in clear [25].

In another work, Liu *et al.* [8] propose a secure two-party protocol for integer comparison problem using the vectorization method and Paillier cryptosystem. This work aims to encode the private inputs into two vectors and then transform the integer comparison problem into a vector-element-selection problem. The computational complexity of the protocol in [8] depends on the encoding vector size to encode both x and y, i.e., $|U|$. The authors claimed that the domain size is usually small in practical applications, but it is not always the case for some real-world applications. For example, two similar companies can have huge value differences (e.g., revenue and asset valuation). Therefore, it requires a

large vector to encode the two values. The authors also confirm that their protocol becomes impractical when the vector size is large.

In this work, we aim to bridge the gap between secure two-party computation and efficient protocol for large integer comparison problem. We improve the two-party *GT* protocol in [8] by using a block vectorization mechanism. Our protocol significantly reduces the size of the encoding vectors and can classify between $x > y$, $x = y$, and $x < y$. We will present our proposed solution in Section IV.

III. TECHNICAL PRELIMINARIES AND DEFINITIONS

A secure two-party computation must ensure the correctness of the output while protecting data privacy. The primary cryptography tool we use to construct our protocol is a homomorphic cryptosystem (e.g., Paillier cryptosystem) and a ciphertext comparison approach. We will briefly describe them in this section to give some ideas on how they can be used to perform computation in an encrypted form. Also, we will formally define the adversary model we use in our protocols design and the definition of privacy for secure two-party computation.

A. SEMI-HONEST ADVERSARY MODEL

In our protocol design, we assume both the client and the server are semi-honest (''honest-but-curious'') [28] parties. They follow the prescribed actions in the protocol but might be interested to learn some extra information from the data or intermediate outputs they received during the protocol execution or from the final output.

B. SECURITY MODEL

In most of the computation protocols, the test output is either kept private to only one party (active player) or shared with all other parties. Fair exchange is needed in some two-party computation protocols to ensure that both parties will receive the same result. The protocol is said symmetric if both parties receive the same information or result.

Generally, a two-party computation problem is cast by specifying a random process that maps pairs of inputs to pairs of outputs [26]. However, this paper considers the asymmetric case where only one party receives the output at the end of the protocol execution while the other party learns nothing. In the setting of a two-party computation, the client (with input X) and the server (with input Y) jointly compute for the function $f(X, Y)$ while preserving some security properties such as the correctness of the output and the data privacy [27].

Let Π be a protocol between Alice (with input X) and Bob (with input Y). Then, we can denote Alice's output by $\Pi_A(X, Y)$ and Bob's output by $\Pi_B(X, Y)$. Since only Alice gets the output in our case, we can simply denote $\Pi(X, Y) = \Pi_A(X, Y)$. The perspective of Alice and Bob during the execution of protocol Π on input (X, Y) can be denoted as $VIEW_A^{\Pi}(X, Y)$ and $VIEW_B^{\Pi}(X, Y)$, respectively. Note that each party's view includes their local input, their output, and their messages received from the other party.

We now formally define our usage of the term privacy in our protocol (adapted from [28]) as follows:

Definition 1 (Privacy w.r.t. Semi-Honest Behavior): Let $f : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^* \times \{0,1\}^*$ be a probabilistic polynomial-time function. We say that a two-party computation protocol Π securely computes f in the presence of semi-honest adversaries if the following holds:

- For every $X, Y \in \{0,1\}^* : \Pi(X, Y) = f(X, Y)$.
- There exists a probabilistic polynomial-time algorithm S_A, such that

$$\{S_A(X, f(X, Y))\}_{X,Y\in\{0,1\}^*} \overset{c}{\equiv} \{VIEW_A^{\Pi}(X, Y)\}_{X,Y\in\{0,1\}^*} \quad (1)$$

- There exists a probabilistic polynomial-time algorithm S_B, such that

$$\{S_B(Y)\}_{X,Y\in\{0,1\}^*} \overset{c}{\equiv} \{VIEW_B^{\Pi}(X, Y)\}_{X,Y\in\{0,1\}^*} \quad (2)$$

where $\overset{c}{\equiv}$ denotes computational indistinguishability according to families of polynomial-size circuits. We refer the reader to [29] for the definition of computational indistinguishability.

We can simulate each party's view using a probabilistic polynomial-time algorithm, only given access to the party's input and output. Thus, we only need to show the existence of a simulator for each party that satisfies the requirements of Eq.1 and Eq.2.

C. PAILLIER CRYPTOSYSTEM

The Paillier cryptosystem [15] is defined as follows. Let p and q are prime numbers where p does not divide $q - 1$ and $p < q$. The public key pk is set to $N = pq$ and the private key pr is set to (λ, N) such that λ is the lowest common multiplier of $p-1$ and $q-1$. To encrypt a message m, a random number r is chosen from 1 to $N - 1$ and the encryption is computed as $Enc_{pk}(m) = (1 + N)^m r^N \bmod N^2$. The decryption of the ciphertext $c_1 = Enc_{pk}(m_1)$ can be computed as $m_1 = \frac{(c_1^{\lambda} \bmod N^2)^{-1}}{N} \lambda^{-1} \bmod N$ where λ^{-1} is actually the inverse of λ in modulo N.

1) ADDITIVE PROPERTY

Given two ciphertexts $Enc_{pk}(m_1)$ and $Enc_{pk}(m_2)$, we can compute $Enc_{pk}(m_1 + m_2)$ as follows:

$$\begin{aligned} &Enc_{pk}(m_1) +_h Enc_{pk}(m_2) \bmod N^2 \\ &= ((1 + N)^{m_1} r_1^N) +_h (1 + N)^{m_2} r_2^N) \bmod N^2 \\ &= ((1 + N)^{m_1+m_2} (r_1 + r_2)^N \bmod N^2 \\ &= Enc_{pk}(m_1 + m_2) \bmod N^2 \end{aligned} \quad (3)$$

2) SCALAR MULTIPLICATION

Given a constant c_1 and a ciphertext $Enc_{pk}(m_1)$, we can compute $Enc_{pk}(c_1 \cdot m_1)$ as follows:

$$\begin{aligned} &c_1 \cdot_h Enc_{pk}(m_1) \\ &= Enc_{pk}(m_1)^{c_1} \bmod N^2 \end{aligned}$$

$$= ((1+N)^{m_1} r_1^N)^{c_1} \bmod N^2$$
$$= (1+N)^{c_1 m_1} r_1^{c_1 N} \bmod N^2$$
$$= Enc_{pk}((c_1 \cdot m_1) \bmod N^2) \quad (4)$$

The security of the Paillier cryptosystem is based on the difficulty of factorizing N. Therefore, for simplicity, we will omit the modulus and keys representation hereafter in this section.

D. NOTATION USED

The notations used hereafter in this paper are summarized in Table 1.

TABLE 1. Notations Used.

Notation	Definition
x	Alice's private input
y	Bob's private input
$P(x, y)$	Comparison result between x and y
U	Encoding vector (contains x and y)
U_i	ith block of U, $U = \cup_{i=1}^{s} U_i$
s	Number of blocks in U
k	Block size
M	size of the encoding vector, i.e., $\|U\|$
$\mathcal{E}$	The event that x and y in the same block
$\mathcal{P}(\mathcal{E})$	Probability that event $\mathcal{E}$ occurs
$s*$	Optimum number of blocks
$k*$	Optimum block size
$E(\cdot)$	Encryption operation
$D(\cdot)$	Decryption operation
$\widetilde{U}$	New encoding vector
$\widetilde{u}_i$	Element i in $\widetilde{U}$
X, X'	Encoded vectors
δ	Threshold value

IV. PROPOSED SOLUTION

In this section, we first explain the vectorization method and the idea of our blocking mechanism. Next, we discuss the selection criteria to determine the parameters (k_i and s) followed by the stopping condition of our protocol. Last, we give the full protocol of the proposed solution.

A. PROTOCOL IDEA

1) VECTORIZATION METHOD

The main idea of the vectorization method [8] is to encode a private input into a vector. For instance, given an input $a \in U = \{u_1, u_2, \ldots, u_d\}$, a will be encoded into an d-dimensional vector $m_1, m_2, \ldots, m_d$ such that

$$m_i = \begin{cases} \alpha & \text{if } 1 \le u_i < x \\ \gamma & \text{if } u_i = x \\ \beta & \text{if } u_i > x \end{cases} \qquad (\alpha \neq \beta \neq \gamma)$$

2) BLOCK PREPARATION

For formulation purposes we will consider the original encoding vector U as an ordered set $U = \{u_1, u_2, \ldots, u_M\} \subset \mathbb{N}$ of length $|U| = M < \infty$, with $u_{\ell+1} = u_\ell + 1, \ell = 1, \ldots, M-1$. The secret values x and y are members of the set, $x, y \in U$. Then, we partition U into subsets $U_i = \{u_i^{(1)}, \ldots, u_i^{(k_i)}\}$ of length $|U_i| = k_i$ such that $U = \cup_{i=1}^{s} U_i$ with $U_i \cap U_j = \emptyset$ for $i \neq j$, $u_1^{(1)} = u_1$, $u_{i+1}^{(1)} = u_i^{(k_i)} + 1$ and $u_s^{(k_s)} = M = \sum_{i=1}^{s} k_i$. We denote the subsets U_i as blocks. Apparently, we generally have $x \in U_i$ and $y \in U_j$. If x and y are in the same block, then $i = j$. The next step is to optimize the values of the $s+1$ involved parameters, $\{s, k_1, \ldots, k_s\}$, in order to automatize our algorithm and achieve higher efficiency. For simplicity, we require $k_i = k_j = k$, reducing the parameter set to $\{s, k\}$, and $M = ks$. The latter condition is generally not satisfied since M cannot always be exactly divided by k. Thus, a more accurate requirement would be $M = k(s-1) + k_s$. However, in both cases, the optimization procedure yields the same results. Thus, we choose here to present the simplest case, i.e., $M = ks$.

Optimizing k and s: To determine the optimal values $\{s^*, k^*\}$ of $\{s, k\}$ we consider the following optimization problem. Assuming the event $\mathcal{E}$ that x and y are in the same block. Then, we want to minimize the probability $\mathcal{P}(\mathcal{E})$ for this event to occur. To calculate the probability we introduce X_1 and X_2 to be random variables with $X_1, X_2 \in \{1, \ldots, s\}$ describing the bock in which x and y lie, respectively. Then, the probability is given by

$$\mathcal{P}(\mathcal{E}) = \mathcal{P}\left(\bigcup_{i=1}^{s}\left[\{X_1 = i\} \bigcap \{X_2 = i\}\right]\right)$$
$$= \sum_{i=1}^{s} \mathcal{P}\left(\{X_1 = i\} \bigcap \{X_2 = i\}\right)$$
$$= \sum_{i=1}^{s} \mathcal{P}(X_1 = i)\mathcal{P}(X_2 = i | X_1 = i)$$
$$= \sum_{i=1}^{s} \left(\frac{k}{M}\right)^2 = \frac{k}{M}. \quad (5)$$

If $k < s$, then $\min \mathcal{P}$ leads to $\{k, s\} = \{1, M\}$. This is precisely the case when the original protocol is applied on U without any modifications. Therefore, we are looking for a solution for $k \ge s$. Formulating the former constraint as $k - s = L \ge 0$ and combining it with $ks = M$ we can express k and s in terms of L as

$$k = \frac{2M}{\sqrt{L^2 + 4M} - L}, \quad s = \frac{1}{2}\left(\sqrt{L^2 + 4M} - L\right). \quad (6)$$

Substituting k into Eq. (5) we express the probability $\mathcal{P}$ as a function of L,

$$\mathcal{P}(L) = \frac{2}{\sqrt{L^2 + 4M} - L}. \quad (7)$$

Due to $\mathcal{P}(L) \in (0, 1]$ we read $L \in \{0, 1, 2, \ldots, M-1\}$. Since $\mathcal{P}(L)$ exhibits a monotone increasing behavior with respect to L, it attains its minimum value for $L = 0$. Substituting it into Eq. (6), we determine the optimal values $\{k^*, s^*\}$ of $\{k, s\}$ as

$$k^* = \sqrt{M} = s^*. \quad (8)$$

In this solution, however, we have $k^*, s^* \in \mathbb{R}^+$ while $k, s \in \mathbb{N}$. In order to overcome this difficulty, we slightly

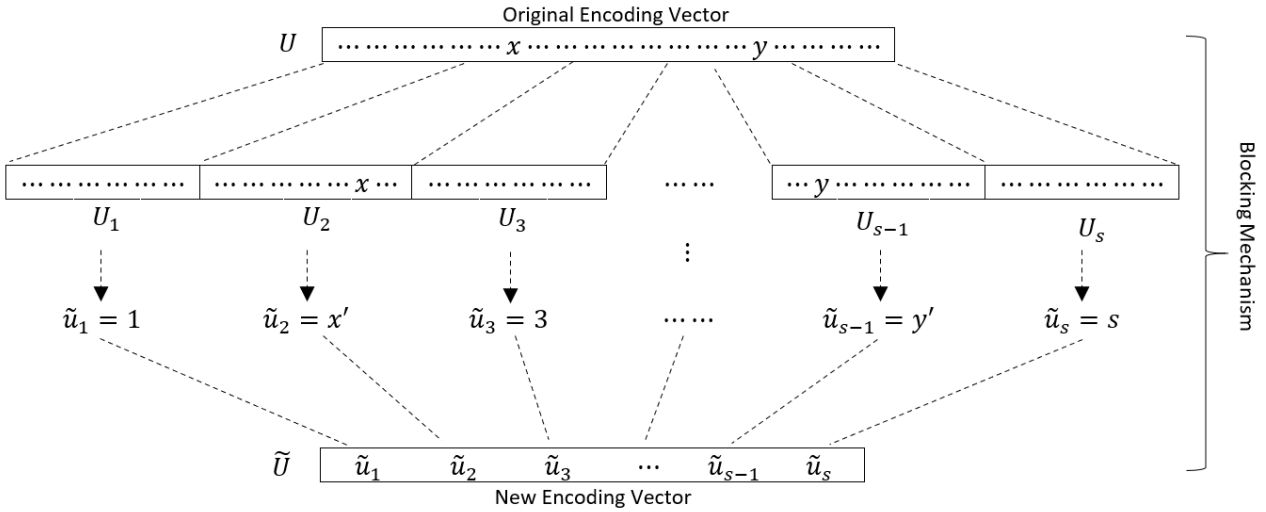

FIGURE 1. Proposed blocking mechanism.

modify s^* and k^* so that we are still as very close to their optimal values. Then, taking into account $k \geq s$, we define

$$s^* := \left\lfloor \sqrt{M} \right\rfloor, \qquad k^* := \left\lfloor M/s^* \right\rfloor, \tag{9}$$

where $\lfloor \cdots \rfloor$ is the floor function. Accordingly then, if $M = k^* s^*$, the set U of length M will be partitioned into s^* blocks of length $|U_1| = \cdots = |U_{s^*}| = k^*$. On the other hand, if $M > k^* s^*$, then the set U will be partitioned into $s^* + 1$ blocks of length $|U_1| = \cdots = |U_{s^*}| = k^*$ and $|U_{s^*+1}| = M - k^* s^* \in [0, k^*)$. Hereafter, to keep our notation as simple as possible, we will denote the optimum values in Eq. (9) as s and k.

3) NEW ENCODING VECTOR

Having determined the values of the parameters k and s, in the next step we assign a unique number to each block as $U_i \rightarrow \tilde{u}_i$, with $\tilde{u}_i < \tilde{u}_{i+1}$, e.g. $U_i \rightarrow \tilde{u}_i = i$, creating the new encoding vector $\tilde{U} = \{\tilde{u}_1, \ldots, \tilde{u}_i, \ldots, \tilde{u}_j, \ldots, \tilde{u}_{s+1}\}$. In this manner, the primal values $x, y \in U$ have been transformed into $x', y' \in \tilde{U}$ as $x \in U_i \rightarrow u_i = x'$ and $y \in U_j \rightarrow u_j = y'$, and therefore (as illustrated in Figure 1), the comparison between x and y is now transformed to the comparison between x' and y' with $|\tilde{U}| < |U|$.

4) STOPPING CONDITION

We impose a stopping condition to our protocol to avoid the worst-case scenario described as follows. Assuming that x and y are equal, and thus, after each iteration, they are still in the same block. Then, after many iterations, we will obtain a block of length 1, comprised of the elements x and y. This is an undesired situation due to privacy concerns. A way to prevent this from occurring is to switch to the original protocol when the length of the new encoding vector is less than or equal to a threshold value δ, $M \leq \delta$.

To do so, we use the probability in Eq. (5). The value of δ generally depends on the application area of our protocol. In Figure 3 we demonstrate how different values of δ may affect the execution time of our protocol.

B. OUR PROTOCOL

In our protocol design, we assume Alice and Bob hold a secret value x and y, respectively. Both parties has knowledge about $U = \{U_1, U_2, \ldots, U_M\}$, partition of each block $U_i = \{u_i^{(1)}, \ldots, u_i^{(k_i)}\}$, size of each block k_i, the number of blocks s, and the sequence $\tilde{U} = \{\tilde{u}_1, \tilde{u}_2, \ldots, \tilde{u}_s\}$. Note that the default size of U can be pre-determined for the integer comparison problem in different applications. Unlike the protocol in [8], we transform the integer comparison into a block comparison problem rather than the vector element-selection problem. Furthermore, our approach requires block partitions before the vectorization takes place. Thus, the vector's size to encode the private inputs will be reduced significantly for each execution round by employing our block vectorization mechanism.

Our protocol consists of three phases (as shown in Protocol 1): setup, encoding, and result determination. Both parties must complete the setup phase for cryptography keys generation and block preparation to participate in our protocol. Next, the encoding phase is used to encode the private inputs into blocks. Finally, in the result determination phase, Alice will evaluate the comparison result after decryption.

V. ANALYSIS

A. SECURITY ANALYSIS

Recall that the security of our protocol is defined in the semi-honest setting by requiring the existence of simulators $\mathcal{S}_\mathcal{A}$ and $\mathcal{S}_\mathcal{B}$ which can generate the view of Alice and Bob, respectively, given their input (i.e., Alice with x and Bob with y) and output $P(x, y)$). If such simulation is indistinguishable from real-world execution, it implies that the protocol does not reveal any extra information under the semi-honest model. In this section, we explain how to simulate the view of each

Protocol 1 Efficient Two-Party Integer Comparison

Inputs. Alice's input ($x \in U$) and Bob's input ($y \in U$).

Goal. Both parties jointly compute $P(x, y)$ of their inputs.

$$P(x,y) = \begin{cases} -1, & x > y \\ 0, & x = y \\ 1, & x < y \end{cases}$$

The protocol:

1) **Setup Phase.**
   a) Alice first generates the public key $\{g, N\}$ and private key $\{\lambda\}$. Next, she sends the public key to Bob and keeps the private key secretly.
   b) Alice retrieves the pre-determined U and divides it into subsets $U_i = \{U_i^{(1)}, \ldots, U_i^{(k_i)}\}$ of length $|U_i| = k_i$.
   c) Next, Alice generates an encoding vector $\widetilde{U}$ and broadcasts block information $(k_i, s, \widetilde{U})$ to Bob.
2) **Encoding Phase.**
   a) Alice identifies x', the position of $\widetilde{u_i}$ in sequence $\widetilde{U}$ that contains x. Alice then encodes x' into a block vector:
   $$X' = (m_1, m_2, \ldots, m_i, \ldots, m_{s-1}, m_s)$$
   where
   $$m_i = \begin{cases} \alpha & \text{if } 1 \le \widetilde{u_i} < x' \\ \gamma & \text{if } \widetilde{u_i} = x' \\ \beta & \text{if } \widetilde{u_i} > x' \end{cases} \quad (\alpha \neq \beta \neq \gamma)$$
   b) Next, Alice selects s random numbers $r_1, r_2, \ldots, r_s$ to encrypt the vector X' using the Paillier encryption scheme to produce $E(X') = \{E(m_i, r_i) \mid i = 1, 2, \ldots s\}$, where $E(m_i, r_i) = g^{m_i}.r_i^N \mod N^2$ for $i = 1, 2, \ldots s$. Alice sends $E(X')$ to Bob.
   c) Upon receiving $E(X')$ from Alice, Bob identifies y', the position of $\widetilde{u_j}$ in sequence $\widetilde{U}$ that contains y. Next, he selects a random number r_b to compute:
   $$E(m_i, r_i) \times E(0, r_b) = g^{m_i}.r_i^N \; mod \; N^2 = E(\mu)$$
   d) Bob sends $E(\mu)$ to Alice for result determination.
3) **Result Determination Phase.**
   a) By using her private key, Alice decrypts $E(\mu)$ to obtain μ, i.e., $D(E(\mu)) = \mu$. Alice evaluates the comparison result $P(x, y)$ as follows:
      - If $\mu = \alpha$, $P(x, y) = -1$ and $x > y$
      - If $\mu = \beta$, $P(x, y) = 1$ and $x < y$
   b) When $\mu = \gamma$, Alice divides U_i, the block that contains x and y into subsets $\widetilde{U}_i = \{\widetilde{u}_i^{(1)}, \ldots, \widetilde{u}_i^{(k_i)}\}$ of length $|\widetilde{U}_i| = \sqrt{M}$.
   c) Alice and Bob repeat the same operations in the encoding phase.
   d) When the block size $M \le \delta$ (a threshold value), switch to protocol [8].

player using their respective inputs and outputs as follows:

$$\begin{aligned} VIEW_{\mathcal{A}}^{\Pi}(x,y) &= \{x, E(X'), E(\mu), \mu, P(x,y)\}, \\ f_1(x,y) &= f_2(x,y) = OUTPUT_{\mathcal{A}}^{\Pi}(x,y) \\ &= OUTPUT_{\mathcal{B}}^{\Pi}(x,y) = P(x,y) \end{aligned} \quad (10)$$

where x, y are inputs, $E(X')$ is Alice's encryption result, μ is Alice's decryption result, $E(\mu)$ is Bob's computation result, and $P(x, y)$ is the protocol output.

Let us assume that $\mathcal{S}_{\mathcal{A}}$ simulates all internal coin flips of $\mathcal{S}_{\mathcal{B}}$ as described in our protocol. $\mathcal{S}_{\mathcal{A}}$ proceeds as follows:

1. By $f_1(x, y)$, $\mathcal{S}_{\mathcal{A}}$ randomly selects a number $y' \in U$ such that $f_1(x, y) = f_1(x, y')$. Then $\mathcal{S}_{\mathcal{A}}$ constructs vector X'.
2. $\mathcal{S}_{\mathcal{A}}$ encrypts X' by using Paillier encryption scheme to produce $E(X') = \{E(m_i, r_i) \mid i = 1, 2, \ldots s\}$, where r_i are random numbers and $E(m_i, r_i) = g^{m_i}.r_i^N \mod N^2$ for $i = 1, 2, \ldots s$.
3. $\mathcal{S}_{\mathcal{A}}$ selects a random number to compute:
   $$E(m_i, r_i) \times E(0, r') = g^{m_i}.r_i^N \; mod \; N^2 = E(\mu')$$
4. $\mathcal{S}_{\mathcal{A}}$ decrypts $E(\mu')$ to obtain μ', i.e., $D(E(\mu')) = \mu'$.
5. $\mathcal{S}_{\mathcal{A}}$ compares μ and μ' to determine the comparison result $P(x, y')$

$$\begin{aligned} \{\mathcal{S}_{\mathcal{A}}(x, f(x,y))\} &= \{x, E(X'), E(\mu), \mu, P(x,y)\} \quad (11) \\ &\quad \{\mathcal{S}_{\mathcal{A}}((x, f_1(x,y)), f_2(x,y)\}_{x,y} \\ &\overset{c}{\equiv} \{VIEW_{\mathcal{A}}^{\Pi}(x,y), OUTPUT_{\mathcal{B}}^{\Pi}(x,y)\}_{x,y} \quad (12) \end{aligned}$$

Similarly, we can construct a simulator $\mathcal{S}_{\mathcal{B}}$ which simulates the protocol for Bob. Based on the simulation for both parties, the computational indistinguishability for our protocol appears to hold on the first inspection.

B. PRIVACY ANALYSIS

Alice computes $D(E(\mu)) = \mu$; that is, $\mu = m_i$ such that $i = y$. Since $m_i \in \{m_1, \ldots, m_{x-1}, m_x, m_{x+1}, \ldots, m_s\} = \{\alpha, \ldots, \alpha, \gamma, \beta, \ldots, \beta\}$, if $m_i = \alpha$ which implies $m_i \in \{m_1, \ldots, m_{x-1}\}$, so $y < x$. If $m_i = \gamma$ which implies $m_i = m_x$, then $y = x$. If $m_i = \beta$ which implies $m_i \in \{m_{x+1}, \ldots, m_s\}$, then $y > x$.

When Alice receives $E(\mu)$ from Bob, she does not know how to compute $E(\mu)$ because she does not know r_b, so $E(m_i, R_i)$ is private for Bob. Therefore, y is private for Bob. When Bob obtains the result $x > y$ or $x < y$, he cannot know which m_i equals α or β, so x is private for Alice.

C. COMPLEXITY

We use the computational complexity measure to evaluate the efficiency of our protocol.

For comparison, we present in Table 2 the computational complexity of existing works for the two-party GT problem in descending order. The computational cost of the protocol in [1] scales exponentially with the input size, which makes it impractical for large x and y numbers. The modular multiplications in [19] and [20] are $(5c \lg N + 4c - 6)$ and

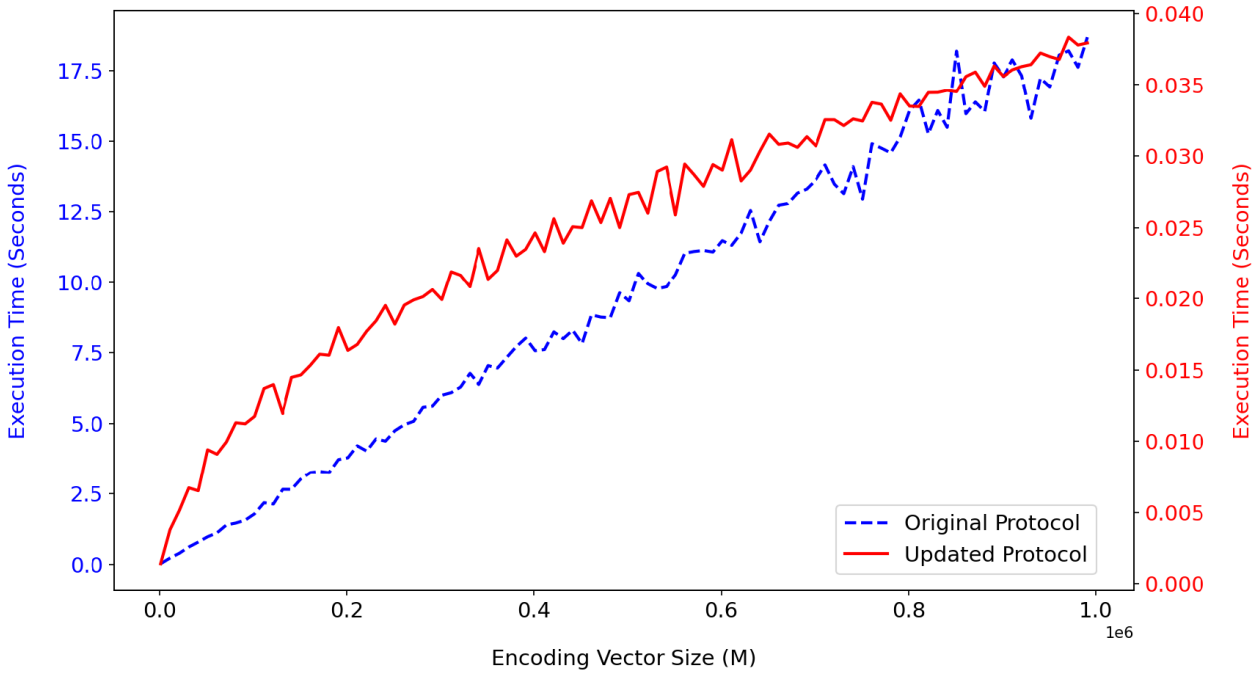

FIGURE 2. Execution time of the original protocol (blue dashed curve) and the updated protocol (red solid curve) as a function of the encoding vector size M, for the threshold value $\delta = 10^3$.

TABLE 2. Computational Complexities.

Schemes	Result	Modular multiplications
Yao [1]	$>, \leq$	Exponential
Lin and Tzeng [19]	$>, \leq$	$5c \lg N + 4c - 6$
Blake and Kolesnikov [20]	$>, \leq$	$(4c+1)\lg N + 6c$
Liu et al. [8]	$>, <, =$	$2(M+2)\lg N$
Protocol 1	$>, <, =$	$2(\sqrt{M}+2)\lg N$

$((4c+1)\lg N + 6c)$, respectively, where c is the bit number of the private inputs and N is the modulus of the encryption scheme. An improved two-party GT protocol reducing the computational cost is presented in [8], i.e., $2(M+2)\lg N$, where M is the size of the encoding vector being much small than c. However, due to its linearity with respect to M, the protocol is associated with a high computational cost for very large M. A significant improvement offers the current blocking mechanism protocol, where the encoding vector size is $\sqrt[2^\ell]{M}$, with the number of iterations $\ell = 1, 2, \ldots, K$ and $\sqrt[2^K]{M} \leq \delta$. Therefore, the computational complexity of our protocol is $\sum_{\ell=1}^{K} 2(\sqrt[2^\ell]{M}+2)\lg N \approx 2(\sqrt{M}+2)\lg N$ modular multiplications. From the former inequality, we can also determine the maximum number of iterations to be $K = \lceil \lg_2(\lg_\delta(M)) \rceil$, where $\lceil \cdots \rceil$ is the ceiling function. For example, for the very large encoding vector size $M = 10^{24}$ and a typical encoding vector size $\delta = 10^3$, we need to run our protocol $K = 3$ times.

D. PERFORMANCE ANALYSIS

In order to analyze the efficiency improvement of our updated protocol compared to the original one, we record in Figure 2 the execution time of both protocols as a function of the encoding vector size M for the worst-case scenario, $x = y$, with the randomly chosen threshold value $\delta = 10^3$. The simulations were conducted with an Intel Xeon W-1250 (VGA 2GB Nvidia Quadro P620) with 16GB of Ram. The curves in Figure 2 represent the ensemble average of 200 experiments for each value of M. The position of $x = y$ in each experiment was randomly determined. We observe two main features. First, the execution time of the updated protocol (solid red line) is at least two orders of magnitude faster than the original protocol (blue dashed line), and second, the update protocol's increase rate is lower than the original one. These features combined unveil that the proposed updated protocol reduces the execution time a great deal compared to the original one for large encoding vectors. This is because the blocking mechanism reduces the size of the encoding vector in each iteration, from M to $\sqrt{M}$. In particular, instead of comparing x and y in a set of M numbers with computational cost of the order $\sim M$, the updated protocol compares x' and y' in a set of $\sqrt{M}$ numbers with computational cost of the order $\sim \sqrt{M}$. This explains why the execution time of the updated protocol increases at a slower rate compared to the original one, which is also numerically verified in Figure 2. Therefore, the updated protocol is proved to be more efficient than the original one in terms of execution time, especially when M is large.

In Figure 3 we plot the ensemble average execution time of 200 experiments with respect to the encoding vector M for different values of the threshold size $\delta = \{10^2, 10^3, 10^4, 10^5\}$. Here, again, we observe two main patterns: the execution time itself and its behavior as a function

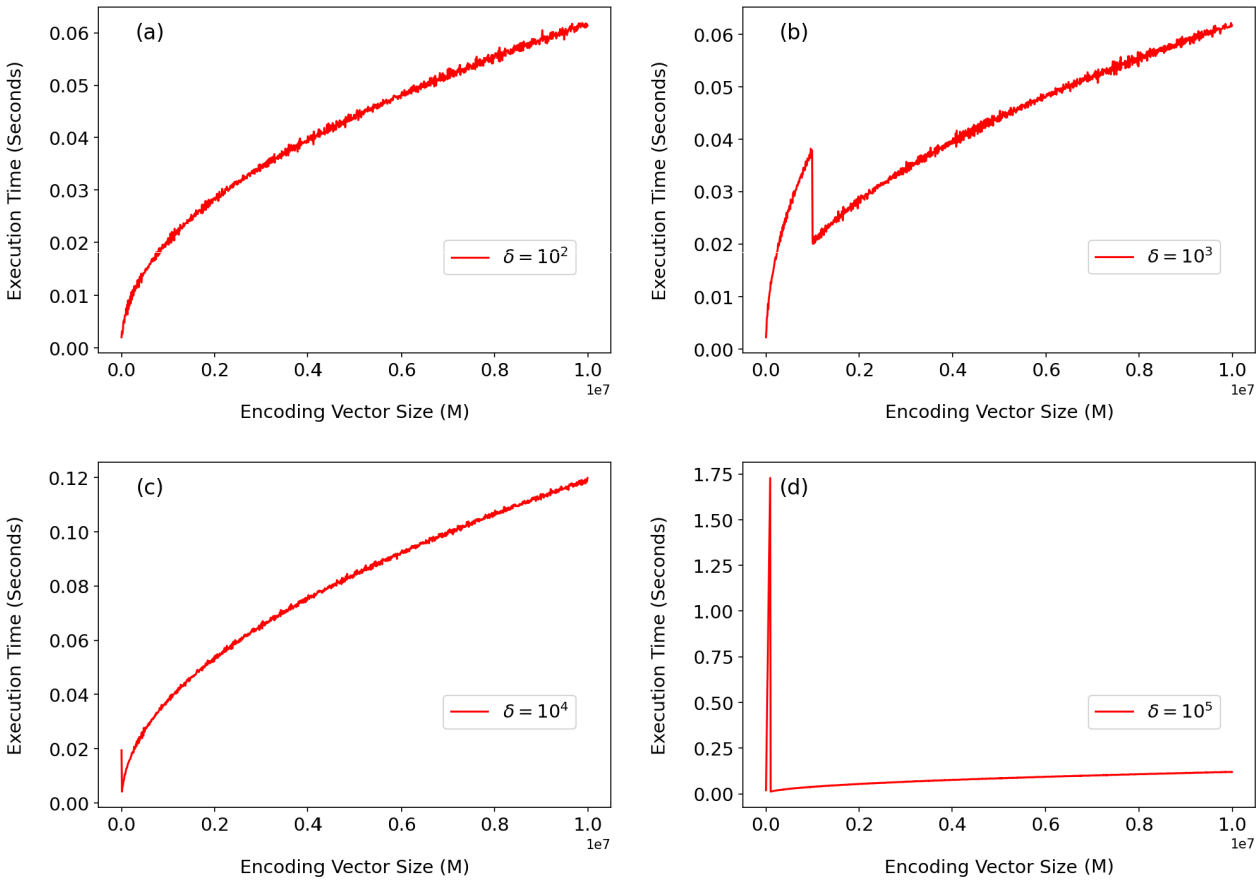

FIGURE 3. Execution time of the updated protocol as a function of the encoding vector size M for various values of δ.

of M strongly depend on the value of δ. It becomes apparent that the execution time increases as δ increases. Therefore, it is recommended to keep δ relatively small. Then, depending on δ, we may observe sudden drops of an infinite slope. This phenomenon depends on the number of iterations of our protocol, or equivalently it depends on how many times the vector U is divided into $\sqrt{|U|}$ blocks. The execution time drops when for each M the number of divisions increases. When $\delta \geq M$, we switch to the original protocol for one last iteration. For the threshold $\delta = 10^2$ in Figure 3(a), the execution time drops 2 times when $M = 10^2$ and $M = 10^4$, corresponding to the first time and the second time of division, respectively. However, since the scale of M is large (up to 10^7), it is difficult to see the execution time drops. Similarly, for the threshold $\delta = 10^3$ in Figure 3(b), the execution time drops twice when $M = 10^3$ and $M = 10^6$. For the last two thresholds $\delta = 10^4$ and $\delta = 10^5$ in Figures 3(c) and 3(d), the execution time drops only once when $M = \delta$.

VI. DISCUSSIONS AND CONCLUSION

In our protocol design, the computational complexity will not be restricted by the size of the encoding vector. Instead, the encoding vector size will be reduced to the number of block partitions for each consecutive iteration. Hence, our protocol can achieve high efficiency even with a large vector size. This property made our solution valuable for applications that required large integer comparisons.

For instance, to check if an IP address is within a certain range, we can convert the IP address into a decimal format and then apply our protocol to determine the result. The IP address with the format $D.C.B.A$ can be converted to $A + (B \times 256) + (C \times 256 \times 256) + (D \times 256 \times 256 \times 256)$ where 265 is just a weight chosen for each part of the IP address. After the conversion, the decimal format of an IP address can be a large integer. For example, given 192.168.55.1, the decimal conversion will be $1 + (55 \times 256) + (168 \times 256 \times 256) + (192 \times 256 \times 256 \times 256) = 3232249600$. Then the server and client can jointly compare two IP addresses to find out if they are within a specific range in a privacy-preserving way.

This work introduced a block vectorization mechanism to deal with the private integer comparison problem under a two-party semi-honest setting. Our protocol achieves high efficiency compared to the original protocol. Furthermore, our protocol can support large integer comparison with fast execution time for both two-party and multi-party settings. Specifically, the same blocking mechanism can be applied before the execution of the multi-party GT problem (secure sorting problem) presented in [8].

We plan to extend our protocol for multi-party settings in malicious and rational adversary models. This future work will require an additional secure mechanism to prevent two or more parties from colluding.

REFERENCES

[1] A. C. Yao, "Protocols for secure computations," in *Proc. 23rd Annu. Symp. Found. Comput. Sci. (SFCS)*, Nov. 982, pp. 160–164.
[2] R. Agrawal and R. Srikant, "Privacy-preserving data mining," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2000, pp. 439–450.
[3] Y. Lindell and B. Pinkas, "Privacy preserving data mining," *J. Cryptol.*, vol. 15, no. 3, pp. 177–206, 2002.
[4] D. A. Gritzalis, "Principles and requirements for a secure E-voting system," *Comput. Secur.*, vol. 21, no. 6, pp. 539–556, Oct. 2002.
[5] P. S. Naidu, R. Kharat, R. Tekade, P. Mendhe, and V. Magade, "E-voting system using visual cryptography & secure multi-party computation," in *Proc. Int. Conf. Comput. Commun. Control Automat. (ICCUBEA)*, Aug. 2016, pp. 1–4.
[6] S. Damle, B. Faltings, and S. Gujar, "A practical solution to Yao's millionaires' problem and its application in designing secure combinatorial auction," 2019, *arXiv:1906.06567*. [Online]. Available: http://arxiv.org/abs/1906.06567
[7] K.-S. Wong and M. H. Kim, "Toward a fair indictment for sealed-bid auction with self-enforcing privacy," *J. Supercomput.*, vol. 74, no. 8, pp. 3801–3819, Aug. 2018.
[8] X. Liu, S. Li, X. B. Chen, G. Xu, X. Zhang, and Y. Zhou, "Efficient solutions to two-party and multiparty millionaires' problem," *Secur. Commun. Netw.*, vol. 2017, May 2017, Art. no. 5207386.
[9] V. Kolesnikov, A. R. Sadeghi, and T. Schneider, "Improved garbled circuit building blocks and applications to auctions and computing minima," in *Proc. Int. Conf. Cryptol. Netw. Secur.*, 2009, pp. 1–20.
[10] T. Chou and C. Orlandi, "The simplest protocol for oblivious transfer," in *Proc. Int. Conf. Cryptol. Inf. Secur. Latin Amer.*, 2015, pp. 40–58.
[11] M. Naor and B. Pinkas, "Efficient oblivious transfer protocols," in *Proc. SODA*, vol. 1, 2001, pp. 448–457.
[12] F. Bourse, O. Sanders, and J. Traoré, "Improved secure integer comparison via homomorphic encryption," in *Proc. Cryptographers' Track RSA Conf.*, 2020, pp. 391–416.
[13] O. Goldreich, S. Micali, and A. Wigderson, "How to play ANY mental game," in *Proc. 19th Annu. ACM Conf. Theory Comput. (STOC)*, 1987, pp. 218–229.
[14] R. Carlton, A. Essex, and K. Kapulkin, "Threshold properties of prime power subgroups with application to secure integer comparisons," in *Proc. Cryptographers' Track RSA Conf.* Cham, Switzerland: Springer, 2018, pp. 137–156.
[15] P. Pascal, "Public-key cryptosystems based on composite degree residuosity classes," in *Proc. Int. Conf. Theory Appl. Cryptograph. Techn.* Berlin, Germany: Springer, pp. 223–238, 1999.
[16] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Trans. Inf. Theory*, vol. 31, no. 4, pp. 469–472, Jul. 1985.
[17] F. Marc, "A cost-effective pay-per-multiplication comparison method for millionaires," in *Proc. Cryptographers' Track RSA Conf.* Berlin, Germany: Springer, pp. 457–471, 2001.
[18] S. Goldwasser and S. Micali, "Probabilistic encryption & how to play mental poker keeping secret all partial information," in *Proc. 14th ACM STOC*, vol. 365, 1982, pp. 173–201.
[19] H.-Y. Lin and W.-G. Tzeng, "An efficient solution to the millionaires' problem based on homomorphic encryption," in *Proc. Int. Conf. Appl. Cryptogr. Netw. Secur.*, vol. 6, New York, NY, USA, 2005, pp. 456–466.
[20] I. F. Blake and V. Kolesnikov, "Strong conditional oblivious transfer and computing on intervals," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur.* Berlin, Germany: Springer, 2004, pp. 515–529.
[21] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proc. 41st Annu. ACM Symp. Symp. Theory Comput. (STOC)*, 2009, pp. 169–178.
[22] J. Fan and F. Vercauteren, "Somewhat practical fully homomorphic encryption," IACR, Cryptol. ePrint Arch., Tech. Rep. 2012/144, 2012, p. 144.
[23] B. Schoenmakers and P. Tuyls, "Practical two-party computation based on the conditional gate," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur.* Berlin, Germany: Springer, 2004, pp. 119–136.
[24] L. Wang, T. K. Saha, Y. Aono, T. Koshiba, and S. Moriai, "Enhanced secure comparison schemes using homomorphic encryption," in *Proc. Int. Conf. Netw.-Based Inf. Syst.* Cham, Switzerland: Springer, 2020, pp. 211–224.
[25] F. Brandt, "Efficient cryptographic protocol design based on distributed El Gamal encryption," in *Proc. Int. Conf. Inf. Secur. Cryptol.* Berlin, Germany: Springer, 2005, pp. 32–47.
[26] O. Goldreich, *Foundations of Cryptography: A Primer*, vol. 1. Boston, MA, USA: Now, 2005.
[27] C. Hazay and Y. Lindell, "Efficient protocols for set intersection and pattern matching with security against malicious and covert adversaries," in *Proc. Theory Cryptogr. Conf.*, 2008, pp. 155–175.
[28] O. Golderich, *Foundations of Cryptography: Basic Applications*, vol. 2. Cambridge, U.K.: Cambridge Univ. Press, 2009.
[29] O. Goldreich, "A note on computational indistinguishability," *Inf. Process. Lett.*, vol. 34, no. 6, pp. 277–281, May 1990.

THAI-HUNG NGUYEN graduated from HUS High School for Gifted Students, Vietnam, mathematics–specialized class, in 2020. He is currently pursuing the degree in computer science with VinUniversity, Vietnam.

He did his Internship at VinBrain from July 2020 to September 2020, working with Computer Vision. He is currently a Research Assistant with the College of Engineering and Computer Science, VinUniversity. He is especially interested in mathematics, computer software, data privacy and security. His research interest includes cryptography for data privacy.

KOK-SENG WONG (Member, IEEE) received the degree in computer science (software engineering) from the University of Malaya, Malaysia, in 2002, the M.Sc. degree in information technology from Malaysia University of Science and Technology (in collaboration with MIT), in 2004, and the Ph.D. degree from Soongsil University, South Korea, in 2012.

He is currently an Associate Professor with the College of Engineering and Computer Science, VinUniversity. Before joining VinUniversity, he has taught computer science subjects (undergraduate and postgraduate) in Kazakhstan, Malaysian, and South Korean universities for the past 16 years. He aims to bring principles and techniques from cryptography to the design and implementation of secure and privacy-protected systems. To this end, he conducts research that spans the areas of security, data privacy, authentication, cloud and edge computing while maintaining a strong relevance to the privacy-preserving framework. His ultimate research goal is to help users to protect their data privacy in this technology era.

THOMAS OIKONOMOU received the degree (Pre-diploma) in physics from RWTH Aachen, Germany, in 2001, the M.Sc. degree (diploma) in general relativity from Friedrich-Schiller University, Jena, Germany, in 2003, and the Ph.D. degree in statistical mechanics and application on biological data from the National and Kapodistrian University of Athens in collaboration with the National Centre of Scientific Research "Demokritos," Greece, in 2008.

He is currently an Assistant Professor with the College of Engineering and Computer Science, VinUniversity. Before joining VinUniversity, he has taught theoretical physics topics (undergraduate and postgraduate) in Kazakhstan for the past five years. His research interests include quantum and out-of-equilibrium statistical thermodynamics, hamiltonian lattices, symbolic dynamics analysis of time series, and bioinformatics.