An Efficient and Probabilistic Secure Bit-Decomposition

Bharath K. Samanthula, Hu Chun, and Wei Jiang
Department of Computer Science, Missouri S&T
500 West 15th Street, Rolla, Missouri 65409
{bspq8, chwrc, wjiang}@mst.edu

ABSTRACT

Many secure data analysis tasks, such as secure clustering and classification, require efficient mechanisms to convert the intermediate encrypted integers into the corresponding encryptions of bits. The existing bit-decomposition algorithms either do not offer sufficient security or are computationally inefficient. In order to provide better security as well as to improve efficiency, we propose a novel probabilistic-based secure bit-decomposition protocol for values encrypted using public key additive homomorphic encryption schemes. The proposed protocol guarantees security as per the semi-honest security definition of secure multi-party computation (MPC) and is also very efficient compared to the existing method. Our protocol always returns the correct result, however, it is probabilistic in the sense that the correct result can be generated in the first run itself with very high probability. The computation time of the proposed protocol grows linearly with the input domain size in bits. We theoretically analyze the complexity of the proposed protocol with the existing method in detail.

Categories and Subject Descriptors

K.6.5 [**Management of Computing and Information Systems**]: Security and Protection (D.4.6)

General Terms

Algorithms, Security

Keywords

Security; Encryption; Bit-decomposition

1. INTRODUCTION

Statistical data analysis is an essential task in many data mining and business intelligence applications [10]. However, when the data come from multiple parties and where user privacy is a big concern, we need to perform the data analysis task in a privacy-preserving manner. The data analysis task becomes even more challenging when the data is in encrypted form which is quite common in outsourced databases.

Though the goal of this paper is not to investigate the existing privacy preserving data mining tasks over the encrypted data, we focus on specific underlying secure operations used in these tasks. In particular, performing secure comparison on the intermediate encrypted results (i.e., given the encryptions of two integers x and y in the cloud, we need to securely evaluate whether x is greater than or equal to y) is an essential operation in many data analysis tasks such as clustering and classification. Nevertheless, most of the existing secure comparison methods [2, 5] require encryptions of the individual bits of the corresponding encrypted value.

At first, it seems simply operating on the encrypted bits from the beginning of data analysis process is a better approach. However, this will lead to inefficient protocols considering the amount of computation involved in massive data analysis tasks. Therefore, many secure data analysis tasks require efficient mechanisms to convert the intermediate encrypted integers into the corresponding encryptions of bits. The process of securely converting an encrypted value of x into encryptions of the individual bits of x is termed as secure bit-decomposition (SBD) [1, 4, 14]. SBD acts as an important primitive in various secure multi-party computation (MPC) [6, 17, 18] protocols such as secure comparison, public modulo and private exponentiation on encrypted integers [7, 9].

Existing SBD methods are based on either secret sharing schemes [15] or threshold homomorphic cryptosystem [3] under MPC framework. We emphasize that SBD methods which are based on secret sharing schemes require at least three parties; therefore, they are not applicable to our problem domain (i.e., semi-honest two-party setting). Whereas, the SBD method proposed in [14], which is based on the (threshold) Paillier homomorphic cryptosystem, can easily be adopted to the two-party case. However, there are two main disadvantages regarding the SBD protocol proposed in [14]:

- The round and communication complexities are independent of the domain size of x (in bits), for $0 \leq x < 2^m$ and $m < K$, where K is the Paillier encryption key size [12]. Instead, they depend on the value of K. Therefore, the case where $m < K$, which is true for most of the practical applications, has no significant benefit with respect to both round and communication complexities. We emphasize that the protocol in [14], for smaller domain values (i.e., $0 \leq x < 2^m$), is not secure since the intermediate decrypted values are not uniformly random in $\mathbb{Z}_N$. On the other hand, the general solution (referred to as the BITREP gate) proposed in [14], for $x \in \mathbb{Z}_N$, is secure but it requires $O(K)$ rounds and also the communication complexity is bounded by $O(K^2)$ in bits;
- The BITREP gate [14] is not efficient. Therefore, using it as a building block in the relevant secure data analysis tasks may reduce the overall throughput.

To overcome the above disadvantages of BITREP gate, we propose an efficient (both in terms of computation and communication) SBD protocol that also requires less number of communication rounds between the two parties in comparison to BITREP. The proposed protocol is very efficient and probabilistic in the number of iterations. However, we show that, with very high probability, our protocol gives the correct result in the first run itself.

1.1 Problem Statement

In our problem setting, we consider two semi-honest (also referred to as honest-but-curious) parties Alice and Bob [6]. We assume that Alice generates a Paillier[1] public/secret key pair (pk, sk) and broadcasts the public key pk to Bob.

Let $\langle E, D\rangle$ be the encryption and decryption functions associated with the public/secret key pair $\langle pk, sk\rangle$. Without loss of generality, assume that Bob holds the Paillier encrypted value $E(x)$, where $0 \leq x < 2^m$ (here m is referred to as the domain size of x in bits). We explicitly assume that x is not known to Alice and Bob. Suppose $\langle x_0, \ldots, x_{m-1}\rangle$ denotes the binary representation of x where x_0 and x_{m-1} are the least and most significant bits respectively. The goal of this paper is to convert encryption of x into the encryptions of the individual bits of x without disclosing any information regarding x to both Alice and Bob. We refer to such a process as secure bit-decomposition (SBD). More formally, we define the SBD protocol as follows:

$$\text{SBD}(E(x)) \rightarrow \langle E(x_0), \ldots, E(x_{m-1})\rangle \quad (1)$$

At the end of the SBD protocol, the values $E(x_0), \ldots, E(x_{m-1})$ are known only to Bob and nothing is revealed to Alice. Note that since SBD protocol is used as a sub-routine in many secure applications, leaking either the value of x or any of the bit values (x_i's) to either Alice or Bob may not be allowed.

Also, we observe that the proposed protocol can be directly used for (Paillier) threshold based setting [3], where the secret key sk is shared between Alice and Bob, without affecting the complexities.

1.2 Our Contributions

This paper presents an efficient (in terms of both computation and communication costs) and probabilistic SBD protocol. The main contributions of this paper are summarized as follows.

(a). **Security -** The proposed protocol provides perfect security as per the standard semi-honest security definition of secure multi-party computation (MPC) framework [6, 17, 18].

(b). **Efficiency -** Our protocol is efficient compared to the solution in [14] both in terms of computation and communication.

(c). **Round Complexity -** The proposed SBD protocol requires exactly m number of communication rounds between Alice and Bob, where $0 \leq x < 2^m$. In general, when $m < K$ which is true in most of the practical applications, the round complexity of our protocol is improved by a factor of $\frac{K}{m}$ over [14].

(d). **Accuracy -** Since the proposed protocol is probabilistic in nature, we include a checking step to securely verify whether the result is correct or not. That is, the steps involved in the proposed protocol can be run in an iterative fashion until the result is correct. Nevertheless, for most of the practical applications, we show that the probability for our protocol to produce the correct result in the first run itself is statistically indistinguishable from "1". More details are given in Section 4.

[1]We emphasize that any additive homomorphic encryption scheme can be utilized in the proposed protocol, however, to be consistent with the previous work, in this paper we use Paillier cryptosystem.

The rest of the paper is organized as follows. In Section 2, we discuss the previous works related to our problem domain. Section 3 explains some definitions and properties as a background knowledge which are used throughout the paper. Then, we provide a detailed description of the proposed SBD protocol along with a running example in Section 4. Also, apart from the security analysis, we theoretically analyze the correctness of the proposed protocol and compare its complexities with those of existing work. We conclude the paper with future work in Section 5.

2. RELATED WORK

2.1 Security Definition

In this paper, the terms privacy and security are interchangeable. In general, privacy and security are closely related to the amount of information disclosed during the execution of a protocol. There are many ways to define information disclosure. This paper adopts the well-known security definition of semi-honest (also referred to as honest-but-curious) model from the field of secure multi-party computation (MPC) [6, 17, 18]. Informally speaking, a protocol Π privately computes function f (known to both parties) if whatever can be obtained from a party's view of a (semi-honest) execution could be essentially obtained from the input and output available to the party. We refer the reader to [6] for detailed security definitions.

2.2 Existing SBD Algorithms

As mentioned in Section 1, the existing SBD algorithms are based on either the secret sharing scheme [15] (where the secret $x \in \mathbb{Z}_p$ is shared among n parties and p is a prime) or the cryptographic setting [3] (Paillier encrypted value of x) under the MPC framework [6, 17, 18].

2.2.1 SBD algorithms under secret sharing

The SBD problem was first introduced in [1] under secret sharing setting assuming the value of x is much smaller than p. However, as mentioned in [4, 14], their approach follows a more complex procedure by involving an addition circuit for securely adding n numbers bitwise; therefore, leading to inefficient solution. Plus, their approach is only passively secure and round complexity is also bounded by $O(\log_2 n + \log_2 m)$ (non-constant-rounds); where n (≥ 3) is the number of parties and m is the (upper-bound) number of bits required to represent x.

Damgård et al. [4] proposed the first constant round solution to the SBD problem using linear secret sharing scheme such as Shamir's scheme [15]. However, as mentioned in [13, 16], their method is still expensive as it requires $O(l \log_2 l)$ secure multiplications, where $l = \lceil \log_2 p\rceil$. Also, their method requires 38 communication rounds. Following the work from [4], Nishide and Ohta [11] improved the round and communication complexity by a constant factor. Their method requires 25 rounds and computation complexity is still bounded by $O(l \log_2 l)$ secure multiplications. Toft [16] has proposed an almost linear and constant rounds solution to the SBD problem, under secret sharing, by reducing the bit-decomposition problem to a postfix comparison.

2.2.2 SBD algorithms under Paillier cryptosystem

As an independent work, most related to ours, Schoenmakers and Tuyls [14] considered multi-party threshold (Paillier) homomorphic cryptosystem [3] and solved the problem of converting a given Paillier encrypted integer x (under $\mathbb{Z}_N$) into Paillier encryptions of the individual bits of x (referred to as BITREP gate). We emphasize that the BITREP gate can directly be applied to a two-party case. However, though their method is secure, the compu-

Algorithm 1 Binary$(x) \rightarrow \langle x_0, \ldots, x_{m-1} \rangle$

```
Require: A positive decimal integer $x$, where $0 \leq x < 2^m$
 1: $i \leftarrow 0$
 2: while $i \neq m$ do
 3:     $x_i \leftarrow x \bmod 2$
 4:     $x \leftarrow \lfloor \frac{x}{2} \rfloor$ {observe that $x$ is updated to current quotient $q_i$}
 5:     $i \leftarrow i + 1$
 6: end while
```

tation and communication costs of BITREP gate are still high and also the round complexity is bounded by $O(K)$.

In a recent work, Reisted and Toft [13] proposed the first constant-round linear solution for the SBD problem under arbitrary setting. However, we observe that their protocol is still inefficient (requiring $O(K)$ secure multiplications). In addition, which is also the main disadvantage, the security of their protocol is not perfect because it uses an LSB gate of [14] which is not secure as a sub-routine. Therefore, for the rest of this paper, we compare our proposed solution with the BITREP (since it is the only existing secure solution, to the best of our knowledge, under Paillier cryptosystem).

3. PRELIMINARIES

3.1 Standard Binary Conversion Method

Our protocol uses standard binary conversion algorithm as a baseline. Let x be an integer such that $0 \leq x < 2^m$. The overall steps involved in the standard binary conversion method are highlighted in Algorithm 1. Briefly, we first divide x by 2. The remainder 0 or 1 (i.e., $x \mod 2$) will be the bit in question and then x is replaced by the quotient (denoted by q_0, where $q_0 = \lfloor \frac{x}{2} \rfloor$). This process is repeated until m iterations.

3.2 Paillier Cryptosystem

The Paillier cryptosystem is an additive homomorphic and probabilistic asymmetric encryption scheme whose security is based on the Decisional Composite Residuosity Assumption [12]. Let E be the encryption function with public key pk given by (N, g) and D be the decryption function with secret key sk given by a trapdoor function λ (that is, the knowledge of the factors of N). Here, $N = p*q$ of bit length K (security parameter usually of 1024 bits); where p and q are large primes of similar bit length, and generator $g \in \mathbb{Z}^*_{N^2}$. For any given $y, z \in \mathbb{Z}_N$, the Paillier cryptosystem exhibits the following properties:

a. **Homomorphic Addition -** $E(y+z) \leftarrow E(y)*E(z) \bmod N^2$;

b. **Homomorphic Multiplication -** $E(z*y) \leftarrow E(y)^z \bmod N^2$;

c. **Semantic Security -** The encryption scheme is semantically secure [6]. Briefly, given a set of ciphertexts, an adversary cannot deduce any additional information about the plaintext.

4. THE PROPOSED PROTOCOL

In this section, we present our new probabilistic SBD protocol (denoted as SBD$_\text{p}$) based on the standard binary conversion algorithm. In addition, we analyze its security and correctness, and also theoretically compare the complexity of SBD$_\text{p}$ with the existing secure bit-decomposition protocol (i.e, BITREP gate [14]).

As mentioned earlier, we assume that Bob holds a Paillier encryption of integer x i.e., $E(x)$ such that x is not known to both Alice and Bob, where $0 \leq x < 2^m$. The goal of SBD$_\text{p}$ is to compute $E(x_0), \ldots, E(x_{m-1})$ without revealing either the value of x or any of x_i's to Alice and Bob. At the end of the SBD$_\text{p}$ protocol, only Bob knows the encryptions of x_i's, for $0 \leq i \leq m-1$. The proposed SBD$_\text{p}$ protocol mainly involves the following two stages:

Algorithm 2 SBD$_\text{p}(E(x)) \rightarrow \langle E(x_0), \ldots, E(x_{m-1}) \rangle$

```
Require: Bob has Paillier encrypted value $E(x)$, where $x$ is not
    known to both parties and $0 \leq x < 2^m$; (Note: The public key
    $(g, N)$ is known to both Alice and Bob whereas the secret key
    $sk$ is known only to Alice)
 1: $l \leftarrow 2^{-1} \mod N$
 2: $T \leftarrow E(x)$
 3: for $i = 0 \rightarrow m-1$ do
 4:     $E(x_i) \leftarrow$ Encrypted_LSB$(T, i)$
 5:     $Z \leftarrow T * E(x_i)^{N-1} \bmod N^2$
        {update $T$ with the encrypted value of $q_i$}
 6:     $T \leftarrow Z^l \bmod N^2$
 7: end for
 8: $\gamma \leftarrow$ SVR$(E(x), \langle E(x_0), \ldots, E(x_{m-1}) \rangle)$
 9: if $\gamma = 1$ then
10:     return
11: else
12:     go to Step 2
13: end if
```

- **Stage 1** - Computing Encrypted Bits of x:
  In this Stage, Alice and Bob (with input $E(x)$) jointly compute encryptions of the individual bits of x in an iterative fashion. At the end of this stage, only Bob knows $E(x_i)$'s, for $0 \leq i \leq m-1$. Other than the output, during this process, nothing is revealed to Alice and Bob.
- **Stage 2** - Secure Verification of Result:
  Following from Stage 1, Alice and Bob securely verify whether the result from Stage 1 is correct or not. Only the output of the verification stage is revealed to Alice and Bob. If the result from Stage 1 is correct, then both parties terminate the protocol. Otherwise, the SBD$_\text{p}$ protocol is repeated.

The overall steps involved in SBD$_\text{p}$ are highlighted in Algorithm 2. Now, we explain the steps involved in each stage in detail.

Stage 1 - Computing Encrypted Bits of x: Initially, Bob computes l as the multiplicative inverse of 2 modulo N and assigns $E(x)$ to T. Then, in an iterative fashion, Bob and Alice jointly compute the encrypted value $E(x_i)$, for $0 \leq i \leq m-1$, by updating T based on the standard binary conversion given in Algorithm 1. Briefly, in the first iteration, Bob computes $E(x_0)$ (the encryption of least-significant bit of x) using T. Then he updates T with the encryption of quotient q_0 (which is equivalent to $\lfloor \frac{x}{2} \rfloor$). This process is continued until m iterations. Note that in iteration i, encryption of bit x_i (i.e., $E(x_i)$) is revealed only to Bob, for $0 \leq i \leq m-1$. Before explaining the steps of Stage 1, we first discuss its basic idea which follows from Observations 1 and 2.

OBSERVATION 1. *For any given x, let $y = x + r \mod N$, where r is a random number from Z_N. Here the relation between y and r depends on whether $x + r \mod N$ leads to an overflow (i.e., $x + r$ is greater than N) or not. We observe that y is always greater than r if there is no overflow. Similarly, in case of overflow, y is always less than r.*

The above observation is clear from the fact that $y = x + r$ if there is no overflow. On the other hand, if there is an overflow, then $y = x + r - N$.

Algorithm 3 Encrypted_LSB$(T, i) \rightarrow E(x_i)$

```
Require: Bob has $T$ from current iteration $i$
1: Bob:
    (a). $Y \leftarrow T * E(r) \bmod N^2$, where $r$ is random in $\mathbb{Z}_N$
    (b). Send $Y$ to Alice
2: Alice:
    (a). Receive $Y$ from Bob
    (b). $y \leftarrow D(Y)$
    (c). if $y$ is even then $\alpha \leftarrow E(0)$
         else $\alpha \leftarrow E(1)$
    (d). Send $\alpha$ to Bob
3: Bob:
    (a). Receive $\alpha$ from Alice
    (b). if $r$ is even then $E(x_i) \leftarrow \alpha$
         else $E(x_i) \leftarrow E(1) * \alpha^{N-1} \mod N^2$
    (c). return $E(x_i)$
```

OBSERVATION 2. *For any given $y = x + r \mod N$, where N is odd, the following property regarding the least significant bit of x (i.e., x_0) always hold:*

$$x_0 = \begin{cases} \lambda_1 \oplus \lambda_2 & \text{if } r \text{ is even} \\ 1 - (\lambda_1 \oplus \lambda_2) & \text{otherwise} \end{cases}$$

Here λ_1 denotes whether an overflow occurs or not, and λ_2 denotes whether y is odd or not. That is, $\lambda_1 = 1$ if $r > y$ (i.e., overflow), and 0 otherwise. Similarly, $\lambda_2 = 1$ if y is odd, and 0 otherwise. Observe that $1 - (\lambda_1 \oplus \lambda_2)$ is the negation of bit $\lambda_1 \oplus \lambda_2$. Also note that N is always odd in the Paillier cryptosystem.

EXAMPLE 1. *Consider an even integer x, i.e., $x_0 = 0$. Without loss of generality, suppose r be an odd integer such that $x + r \bmod N$ causes an overflow. Then, $y = x + r \mod N$ is always an even integer. That is $\lambda_2 = 0$. Also, $\lambda_1 = 1$ since there is an overflow. As r is odd, following from Observation 2, we have* $1 - (\lambda_1 \oplus \lambda_2) = 1 - (1 \oplus 0) = 0 = x_0$. □

By using the properties of Observation 2, Stage 1 computes the encryption of least significant bit of T by iteratively updating T to the encryption of corresponding quotient. The main steps involved in Stage 1 are shown as steps 2 to 7 in Algorithm 2. Stage 1 utilizes a sub-protocol (denoted by Encrypted_LSB) to compute the encryption of least significant bit from current T. The overall steps involved in the Encrypted_LSB protocol are shown in Algorithm 3.

To start with, in the first iteration, Bob randomizes the value of T (note that initially $T = E(x)$) by computing $Y = T * E(r) \bmod N^2$, and sends Y to Alice[2]. Upon receiving, Alice decrypts it to get $y = D(Y)$, where $y = x + r \bmod N$. If y is odd, Alice computes $\alpha = E(1)$, else she computes $\alpha = E(0)$, and sends it to Bob. Observe that $\alpha = E(\lambda_2)$. Also it is possible to securely compute $E(\lambda_1)$ using existing secure integer comparison protocol [8] on y (known only to Alice) and r (known only to Bob). However, though this construction is better than BITREP gate, we observe that the efficiency gains are not that significant. Therefore, in our

[2]In Paillier's cryptosystem, ciphertext multiplications are followed by modulo N^2 operation so that the resulting ciphertext is in $\mathbb{Z}_{N^2}$.

Algorithm 4 SVR$(E(x), \langle E(x_0), \dots, E(x_{m-1})\rangle) \rightarrow \gamma$

```
Require: Bob has $E(x)$ and $\langle E(x_0), \dots, E(x_{m-1})\rangle$
1: Bob:
    (a). $U \leftarrow \prod_{i=0}^{m-1} (E(x_i))^{2^i} \bmod N^2$
    (b). $V \leftarrow U * E(x)^{N-1} \bmod N^2$
    (c). $W \leftarrow V^{r'} \bmod N^2$, where $r'$ is random in $\mathbb{Z}_N$
    (d). Send $W$ to Alice
2: Alice:
    (a). Receive $W$ from Bob
    (b). if $D(W) = 0$ then $\gamma \leftarrow 1$
         else $\gamma \leftarrow 0$
    (c). Send $\gamma$ to Bob
```

SBD$_\text{p}$ protocol we decided to use a probabilistic assumption. That is, Bob simply assumes $\lambda_1 = 0$ (no overflow). We emphasize that though Bob assumes no overflow, $y = x + r \mod N$ can still have overflow which depends on the actual values of x and r. In the later parts of this section, we show that for many practical applications, the above probabilistic assumption is reasonable.

Once Bob receives α from Alice, he computes $E(x_i)$ depending on whether r is even or odd as below.

- If r is even, then $E(x_0) = \alpha = E(\lambda_2)$.
- Else $E(x_0) = E(1) * \alpha^{N-1} \mod N^2 = E(1 - \lambda_2)$

Since λ_1 is assumed to be 0, following from Observation 2, we have $\lambda_1 \oplus \lambda_2 = \lambda_2$. Also, note that $N - 1$ is equivalent to -1 under $\mathbb{Z}_N$. After this, Bob updates T to $E(q_0) = E(\lfloor \frac{x}{2} \rfloor)$ by performing following homomorphic additions:

- $Z = T * E(x_0)^{N-1} \bmod N^2 = E(x - x_0)$
- $T = Z^l \bmod N^2 = E((x - x_0) * 2^{-1}) = E(q_0)$

where $l = 2^{-1} \mod N$. The main observation is that $x - x_0$ is always a multiple of 2; therefore, $(x - x_0) * 2^{-1}$ always gives the correct quotient under $\mathbb{Z}_N$. The above process is continued iteratively such that in iteration i, Bob computes $E(x_i)$ and updates T to $E(q_i)$, for $0 \leq i \leq m - 1$.

Stage 2 - Secure Verification of Result: Since Bob assumes $\lambda_1 = 0$ during each iteration of Stage 1, there is a possibility for the wrong outcome. Though this probability is very low for many practical applications, as will be shown later, it is better to provide a mechanism to mitigate this effect. Along this direction, Stage 2 securely verifies whether the result of Stage 1 is correct and takes appropriate steps to compute the correct result (only in the case of wrong result from Stage 1).

The overall steps involved in Stage 2 of SBD$_\text{p}$ are highlighted as steps 8 to 13 in Algorithm 2. As shown in step 8 of Algorithm 2, Stage 2 utilizes a sub-protocol that performs the secure verification of the result (denoted by SVR). The main steps involved in SVR are given in Algorithm 4. Initially, Bob computes the encrypted integer corresponding to the encrypted bits computed in Stage 1 and performs the following homomorphic operations.

- $U = E(x_0 + \ldots + 2^{m-1} * x_{m-1})$
- $V = E(x_0 + \ldots + 2^{m-1} * x_{m-1} - x)$
- $W = V^{r'} = E(r' * (x_0 + \ldots + 2^{m-1} * x_{m-1} - x))$

Table 1: Complexity comparison of existing Paillier based SBD algorithm with our work

Method	**Rounds**	**Computations**	**Communications**
BITREP gate [14]	$O(K)$	$21K + 2$ encryptions and $179K$ exponentiations	$O(K^2)$ bits
This paper	$O(m)$	$3m + 1$ encryptions and $4m + 2$ exponentiations	$O(m * K)$ bits

After this, Bob sends W to Alice. Upon receiving W, Alice decrypts it and sets γ to 1 (denoting the result from Stage 1 is correct) if $D(W) = 0$, else γ is set to 0 (denoting the result is incorrect), and sends γ to Bob. Observe that $\gamma = 0$ iff $x = x_0 + \ldots + 2^{m-1} * x_{m-1}$, i.e., result of Stage 1 is correct. Note that $D(W)$ is either 0 or random; therefore, the above process does not reveal any information about x and x_i's to Alice. Finally, if $\gamma = 1$, Bob terminates the SBD$_\text{p}$ protocol implying the result of Stage 1 is correct. Else, Bob initiates Stage 1 of SBD$_\text{p}$ (i.e., go to Step 2 of Algorithm 2) and the above process is repeated until the result is correct.

EXAMPLE 2. *Suppose $x = 5$ and $m = 3$. We show various intermediate results during the execution of Stage 1 in the proposed protocol. Initially T is set to $E(5)$. We assume that r is even and there is no overflow. Note that r is different in each iteration.*

Iteration 0:

$$\begin{aligned} y &= 5 + r \mod N = \textit{an odd integer} \\ E(x_0) &= \alpha = E(1) \\ T &= E((5-1) * 2^{-1}) = E(2) \end{aligned}$$

Iteration 1:

$$\begin{aligned} y &= 2 + r \mod N = \textit{an even integer} \\ E(x_1) &= \alpha = E(0) \\ T &= E((2-0) * 2^{-1}) = E(1) \end{aligned}$$

Iteration 2:

$$\begin{aligned} y &= 1 + r \mod N = \textit{an odd integer} \\ E(x_2) &= \alpha = E(1) \\ T &= E((1-1) * 2^{-1}) = E(0) \end{aligned}$$

The output of Stage 1 is $\langle E(1), E(0), E(1) \rangle$. □

4.1 Correctness

We emphasize that the final encrypted bits computed from the SBD$_\text{p}$ protocol are always correct. This is because of the secure verification step (i.e., Stage 2) in SBD$_\text{p}$which makes sure that the protocol terminates only if the output of Stage 1 is correct. It is possible for the SBD$_\text{p}$ protocol to generate the correct output after many runs of Stage 1 & 2 (which depends on x and random r chosen in each iteration of Stage 1). Therefore, a natural question to ask is whether there is an upper bound on the number of runs in SBD$_\text{p}$ to produce the correct output. However, the proposed protocol produces the correct output in the first run of Stage 1 and 2 for many practical applications. We theoretically prove this below.

In general, many secure data analysis tasks such as secure clustering and classification has m much smaller than K. During Stage 1, in each iteration, r can take any value in $\mathbb{Z}_N$. We observe that if $r \in [N - 2^m, N)$, only then the corresponding computed (encrypted) bit can be wrong (due to overflow). That is, the number of possible values of r that can give rise to error are 2^m. Since we have N number of possible values for r, the probability for producing wrong bit is $\frac{2^m}{N} \approx \frac{1}{2^{K-m}}$. Therefore, the probability for computing the encryption of x_i correctly is approximately $1 - \frac{1}{2^{K-m}}$. This probability remains same for all the bits since r is chosen independently in each iteration. Hence, the probability for Stage 1 to compute the correct output is given by:

$$\left(1 - \frac{1}{2^{K-m}}\right)^m \approx e^{-\frac{m}{2^{K-m}}}$$

In general, for many real-world applications, m can be at most 100. Therefore, for 1024-bit key size (i.e., $K = 1024$), the probability for SBD$_\text{p}$ to produce correct output in the first run is approximately $e^{-\frac{100}{2^{924}}} \approx 1$. Hence, for practical domain values of x, with a probability of almost 1, the SBD$_\text{p}$ protocol gives the correct output in the first run itself. Based on the above analysis, it is clear that Stage 2 is not required in practice. We emphasize that even in the extreme case such as $m = 950$, the probability for SBD$_\text{p}$ to produce correct output in the first run is $e^{-\frac{950}{2^{74}}} \approx 1$.

4.2 Security Analysis

In the SBD$_\text{p}$ protocol, the communication between Alice and Bob occurs both in Stage 1 and 2. During Stage 1, in each i^{th} iteration, Bob randomizes current T (from step 1(a) of Algorithm 3) and sends it to Alice. Therefore, the decryption of Y by Alice (from step 2(b) of Algorithm 3) yields a random value y which is uniformly distributed in $\mathbb{Z}_N$. Thus, no information about x is revealed to Alice. Also, depending on whether y is even or odd, Alice sends $\alpha = E(0)$ or $E(1)$ to Bob. Here, the ciphertext α is uniformly random in $\mathbb{Z}_{N^2}$ and does not provide any information regarding the corresponding plaintext to Bob due to the semantic security of Paillier cryptosystem [12]. Note that, after computing $E(x_{i-1})$ in the i^{th} iteration, the process of updating T is done locally by Bob, for $1 \leq i \leq m$. Hence, it is clear that no information related to x is revealed to Alice and Bob during Stage 1 of SBD$_\text{p}$.

During Stage 2, Bob sends $W = V^{r'} \mod N^2$ to Alice (step 1(d) of Algorithm 4), where r' is a random number in $\mathbb{Z}_N$ known only to Bob. Since the decryption of W by Alice gives either 0 or a uniformly random value in $\mathbb{Z}_N$, no information is revealed to Alice and Bob except whether the output of Stage 1 is correct or not. Therefore, Stage 2 is secure under the assumption (which is also practical) that the output γ can be revealed to both parties. We emphasize that revealing γ does not leak any information regarding x to Alice and Bob. In addition, the intermediate results which are passed as inputs from Stage 1 to Stage 2 are pseudo-random. Hence, by composition theorem [6], the sequential composition of the two stages lead to a secure protocol.

4.3 Complexity Analysis

The comparison results of our approach with the BITREP gate are as shown in Table 1. As mentioned in [14], the BITREP gate utilizes an addition and subtraction circuit whose round complexities are bounded by $O(K)$, where K is the encryption key size. Also, the overall number of computations involved in the BITREP gate are roughly $21K + 2$ encryptions and $179K$ exponentiations under the assumption that Paillier's encryption and decryption operations take similar amount of time. The communication complexities of addition and subtraction circuits, as mentioned in [14], are bounded by $O(K^2)$ bits. Therefore, the overall communication complexity of BITREP gate is bounded by $O(K^2)$ bits.

On the other hand, for the SBD_p protocol, Stage 1 involves m number of iterations, where each iteration requires one round of communication between Alice and Bob. Whereas, Stage 2 requires one round of communication between Alice and Bob. As shown earlier, for many practical applications, the probability of getting the correct result in the first instantiation of the SBD_p protocol is very high. Therefore, we assume that Stage 1 and 2 are run only once. Thus, the round complexity of SBD_p is bounded by $O(m)$.

During Stage 1 of SBD_p, in each iteration, Bob has to randomize current T which requires one encryption and one homomorphic addition operations. Alice further performs one decryption and one encryption operations (at step 2(b) and 2(c) of Algorithm 3). Also, if r is odd, Bob has to perform one exponentiation and one homomorphic addition operations (at step 3(b) of Algorithm 3). In addition, Bob has to perform two exponentiation and one homomorphic addition operations while updating T in each iteration. We emphasize that the values of $l = 2^{-1} \mod N$ and $E(1)$ (step 3(b) of Algorithm 3) are pre-computed and can be re-used in each iteration. Therefore, Stage 1 requires $3m$ encryptions (assuming that encryption and decryption under Paillier take almost similar time), $3m$ exponentiation, and $3m$ homomorphic addition operations. In general, the cost of $3m$ homomorphic additions is negligible compared to the cost of $3m$ exponentiations. Therefore, the computation cost of Stage 1 is bounded by $3m$ encryptions and $3m$ exponentiations.

Furthermore, Stage 2 of SBD_p requires one decryption, $m + 2$ exponentiation, and m homomorphic addition operations. Hence, the total computation cost of SBD_p (one instantiation) is roughly $3m + 1$ encryptions and $4m + 2$ exponentiations.

The communication complexity of our protocol entirely depends on Stage 1. During Stage 1, in each iteration, Bob sends encrypted value Y to Alice. Also, Alice sends an encrypted value α to Bob. Since we have m iterations, the total communication complexity of SBD_p is bounded by $O(m * K)$ bits. Based on the above analysis, when $m < K$, it is clear that the complexities of SBD_p are significantly much better than those of BITREP. However, when $m = K$ (i.e., $x \in \mathbb{Z}_N$), the communication and round complexities of our protocol are the same as in BITREP. Nevertheless, in either case, SBD_p is more efficient than BITREP as shown in Table 1.

5. CONCLUSION AND FUTURE WORK

Secure bit-decomposition (SBD) acts as an important primitive in various secure computation operations such as comparison and public modulo. The existing SBD protocol is inefficient and also requires a number of communication rounds bounded by $O(K)$, where K is the encryption key size. Along this direction, this paper presented a new efficient and probabilistic SBD protocol, referred to as SBD_p, to securely convert a Paillier encrypted value of x, where $0 \leq x < 2^m$, into encryptions of the individual bits of x.

The proposed protocol always generates the correct result. However, we theoretically showed that the probability for SBD_p to generate the correct output in the first run itself is almost 1 for many practical applications. In addition, we analyzed the complexities of the proposed protocol with the existing secure approach (i.e., BITREP gate [14]). We emphasize that, apart from the efficiency gain, the round and communication complexities of our protocol are improved by a factor of $\frac{K}{m}$ compared to BITREP. As a future work, we will focus on extending our protocol to multi-party case.

Acknowledgment

This material is based upon work supported by the Office of Naval Research under Award No. N000141110256 and NSF under award No. CNS-1011984.

6. REFERENCES

[1] J. Algesheimer, J. Camenisch, and V. Shoup. Efficient computation modulo a shared secret with application to the generation of shared safe-prime products. In *CRYPTO*, pages 417–432. Springer-Verlag, 2002.

[2] I. F. Blake and V. Kolesnikov. One-round secure comparison of integers. *Journal of Mathematical Cryptology*, 3(1):37–68, May 2009.

[3] R. Cramer, I. Damgård, and J. B. Nielsen. Multiparty computation from threshold homomorphic encryption. In *EUROCRYPT*, pages 280–299. Springer-Verlag, 2001.

[4] I. Damgård, M. Fitzi, E. Kiltz, J. B. Nielsen, and T. Toft. Unconditionally secure constant-rounds multi-party computation for equality, comparison, bits and exponentiation. In *Theory of Cryptography Conference (TCC)*, volume 3876, pages 285–304. Springer, 2006.

[5] M. Fischlin. A cost-effective pay-per-multiplication comparison method for millionaires. In *Proceedings of the Conference on Topics in Cryptology: The Cryptographer's Track at RSA*, pages 457–472. Springer-Verlag, 2001.

[6] O. Goldreich. *The Foundations of Cryptography*, volume 2. Cambridge University Press, 2004.

[7] J. Guajardo, B. Mennink, and B. Schoenmakers. Modulo reduction for paillier encryptions and application to secure statistical analysis. In *Financial Cryptography and Data Security*, volume 6052, pages 375–382. Springer, 2010.

[8] A. E. Nergiz, M. E. Nergiz, T. Pedersen, and C. Clifton. Practical and secure integer comparison and interval check. In *Proceedings of Second International Conference on Social Computing*, pages 791–799. IEEE Computer Society, 2010.

[9] C. Ning and Q. Xu. Constant-rounds, linear multi-party computation for exponentiation and modulo reduction with perfect security. In *ASIACRYPT*, volume 7073 of *LNCS*, pages 572–589. Springer, 2011.

[10] R. Nisbet, J. Elder, and G. Miner. *Handbook of Statistical Analysis and Data Mining Applications*. Academic Press, 2009.

[11] T. Nishide and K. Ohta. Multiparty computation for interval, equality, and comparison without bit-decomposition protocol. In *PKC*, pages 343–360. Springer-Verlag, 2007.

[12] P. Paillier. Public key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology - Eurocrypt '99*, pages 223–238. Springer-Verlag, 1999.

[13] T. Reistad and T. Toft. Linear, constant-rounds bit-decomposition. In *ICISC*, pages 245–257. Springer-Verlag, 2010.

[14] B. Schoenmakers and P. Tuyls. Efficient binary conversion for paillier encrypted values. In *EUROCRYPT*, pages 522–537. Springer-Verlag, 2006.

[15] A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, nov 1979.

[16] T. Toft. Constant-rounds, almost-linear bit-decomposition of secret shared values. In *Proceedings of the The Cryptographers' Track at the RSA Conference on Topics in Cryptology*, pages 357–371. Springer-Verlag, 2009.

[17] A. C. Yao. Protocols for secure computations. In *Symposium on Foundations of Computer Science*, pages 160–164, Washington, DC, USA, 1982. IEEE Computer Society.

[18] A. C. Yao. How to generate and exchange secrets. In *Symposium on Foundations of Computer Science*, pages 162–167. IEEE Computer Society, 1986.