An Efficient and Probabilistic Secure Bit-Decomposition

Reporter: Ximeng Liu
Supervisor: Rongxing Lu

School of EEE, NTU
http://www.ntu.edu.sg/home/rxlu/seminars.htm

May 22, 2014
Statistical data analysis is an essential task in many data mining and business intelligence applications. 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.
secure bit-decomposition (SBD)

SBD acts as an important primitive in various secure multi-party computation (MPC) protocols such as secure comparison, public modulo and private exponentiation on encrypted integers.
We consider two semi-honest (also referred to as honest-but-curious) parties Alice and Bob. We assume that Alice generates a Paillier public/secret key pair \((pk; sk)\) and broadcasts the public key \(pk\) to Bob.
k-Nearest Neighbor algorithm

Let $< E; D >$ be the encryption and decryption functions associated with the public/secret key pair $(pk, sk)$. 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).
Problem Statement

We explicitly assume that $x$ is not known to Alice and Bob. Suppose $(x_0, \cdots, x_{m-1})$ 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.
More formally, we define the SBD protocol as follows:

$$SBD(E(x)) = \langle E(x_0), \cdots, E(x_{m-1}) \rangle$$

At the end of the SBD protocol, the values $E(x_0), \cdots, E(x_{m-1})$ 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.
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.
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: \quad $x_i \leftarrow x \mod 2$
4: \quad $x \leftarrow \left\lfloor \frac{x}{2} \right\rfloor$ \{observe that $x$ is updated to current quotient $q_i$\}
5: \quad $i \leftarrow i + 1$
6: end while
Paillier cryptosystem exhibits the following properties:

a. Homomorphic Addition: $E(y + z) = E(y) \cdot E(z) \mod N^2$;

b. Homomorphic Multiplication: $E(z \cdot y) = E(y)^z \mod N^2$;
Algorithm 2 SBD\(_p\)\(E(x)\) → \(\langle E(x_0), \ldots, E(x_{m-1})\rangle\)

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: \(E(x_i) \leftarrow \text{Encrypted_LSB}(T, i)\)
5: \(Z \leftarrow T * E(x_i)^{N-1} \mod N^2\)
   \{update \(T\) with the encrypted value of \(q_i\)\}
6: \(T \leftarrow Z^l \mod N^2\)
7: end for
8: \(\gamma \leftarrow \text{SVR}(E(x), \langle E(x_0), \ldots, E(x_{m-1})\rangle)\)
9: if \(\gamma = 1\) then
10: return
11: else
12: go to Step 2
13: end if
**Encrypted\_LSB protocol**

**Algorithm 3** Encrypted\_LSB\((T, i) \rightarrow E(x_i)\)

Require: Bob has \(T\) from current iteration \(i\)

1: Bob:
   (a). \(Y \leftarrow T \ast E(r) \mod N^2\), 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 \(E(x_i) \leftarrow E(1) \ast \alpha^{N-1} \mod N^2\)
   (c). return \(E(x_i)\)
Secure Verification of Result (SVR)

Algorithm 4 SVR\( (E(x), (E(x_0), \ldots, E(x_{m-1}))) \rightarrow \gamma \)

**Require:** Bob has \( E(x) \) and \( (E(x_0), \ldots, E(x_{m-1})) \)

1: Bob:
   
   (a). \( U \leftarrow \prod_{i=0}^{m-1} (E(x_i))^{2^i} \mod N^2 \)
   (b). \( V \leftarrow U \ast E(x)^{N-1} \mod N^2 \)
   (c). \( W \leftarrow V^{r'} \mod N^2 \), 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
Thank you

Rongxing’s Homepage:
http://www.ntu.edu.sg/home/rxlu/index.htm

PPT available @: http://www.ntu.edu.sg/home/rxlu/seminars.htm

Ximeng’s Homepage:
http://www.nbnix.com/