A homomorphic encryption algorithm over integers

If a secure and efficient Fully Homomorphic Encryption (FHE) algorithm exists, it should be the ultimate solution for securing data privacy in cloud computing, where any operation can be directly applied to ciphertexts without garbling them. Though some theoretic FHE algorithms do exist, none of them is practical because of their prohibitively expensive computing cost. This research designs and analyses a homomorphic encryption algorithm over integers. Two variants of this algorithm are now under our investigation. Summary of these two variants are as follows:

- **Non-deterministic algorithm:** Both versions are non-deterministic. That is, the encryptions for the same plaintext may result in different ciphers. A nice feature to hide the equality relationship among ciphers from adversaries.
- **Functionality:**
  - Variant 1: Allows an unlimited number of arithmetic additions and multiplications on ciphers until the integer (application data) exceeds the encryption/decryption key size (a large prime). An equality testing algorithm was developed and can be used to test whether two ciphers (of a same integer) are equal.
  - Variant 2: Allows unlimited number of arithmetic additions and multiplications on ciphers until the integer (application data) exceeds a pre-defined size or the corresponding padded integer exceeds the encryption/decryption key (a large prime). Unlike Variant 1 ciphers, Variant 2 ciphers cannot be tested for equality.
- **Key size:** If both variants would like to support applications with the same maximum data value, Variant 2 algorithm's key size will be bigger than that of Variant 1 algorithm. However, both algorithms' key sizes will be in the range of several thousand bits for most applications, which is much better than that of the best implementation of the Gentry-like theoretic FHE algorithms (several megabits).
- **Ciphertext size:** The encryptions and homomorphic additions/multiplications in both variants are all modular operations. Thus, the ciphertext will be less than the modulus used.
- **Computational complexity:** Both variants are very efficient on encryption/decryption (with just a few modular multiplications) and are especially efficient on the homomorphic addition/multiplication (with a single modular addition/multiplication, respectively).
- **Security:** If an adversary knows two ciphers of a same integer, the adversary can break Variant 1, but not Variant 2. We still research on the security of these two variants to other attacks and also is trying to provide security proofs for Variant 2.

The following pages briefly describe both variants.

Homomorphic encryption algorithm: Variant 1

Parameter setup:

1. A data owner D_o randomly picks two large primes P_1 and P_2 repeatedly until $Q = 2P_1 + 1$ is also a prime. Compute the product $N = P_1 \times P_2$. Note that D_o needs to choose P_1 large enough so that all his application data $m < p_1$. If $m \geq p_1$, the cipher of m cannot be decrypted back to m.
2. The encryption key of the algorithm is the pair (P_1, N), which will be used by D_o to perform data encryption and decryption. D_o should keep it in secret.
3. D_o randomly picks another large prime P_3 and then computes the product $T = Q \times P_3$. The data owner D_o also randomly picks a set of numbers $h_1, h_2 \ldots h_k \in Z_T^*$, and then computes

$$g_i = h_i^{2(P_3-1)} \bmod T, \quad \forall i = 1, 2 \ldots k$$

The set of g_i's is the equality test key. The provided equality test algorithm is a probabilistic algorithm. Thus, if more g_i are generated by D_o, the accuracy of the test is higher. The test algorithm will be described later.

4. The set $(N, g_1, g_2 \ldots g_k, T)$ is the operational key of the algorithm, where authorized agents (e.g., cloud servers) or the public can use N to perform arithmetic operations and use $(g_1, g_2 \ldots g_k, T)$ to perform the equality testing.

Encryption: To encrypt an integer $m \in Z_{p_1}$, D_o picks a random number r and use the encryption key (P_1, N) to computes

$$E(m) = rP_1 + m \bmod N$$

Decryption: To decrypt a cipher $E(m)$, D_o can use the key (P_1, N) to compute

$$D(E(m)) = (rP_1 + m \bmod N) \bmod P_1 = m$$

Homomorphic addition: The authorized agents, with the operational key $(N, g_1, g_2 \ldots g_k, T)$, can perform homomorphic addition directly to ciphers. That is,

$$E(m_1) + E(m_2) \bmod N$$

Since

$$\begin{aligned} E(m_1) + E(m_2) \bmod N &= (r_1P_1 + m_1 \bmod N) + (r_2P_1 + m_2 \bmod N) \bmod N \\ &= (r_1 + r_2)P_1 + (m_1 + m_2) \bmod N \\ &= E(m_1 + m_2) \end{aligned}$$

Homomorphic multiplication: The authorized agents, with the operational key $(N, g_1, g_2 \ldots g_k, T)$, can perform homomorphic multiplications directly to ciphers. That is,

$$E(m_1) \times E(m_2) \mod N$$

Since

$$\begin{aligned} E(m_1) \times E(m_2) \mod N &= (r_1 P_1 + m_1 \mod N) \times (r_2 P_1 + m_2 \mod N) \mod N \\ &= r' P_1 + (m_1 + m_2) \mod N \\ &= E(m_1 \times m_2) \end{aligned}$$

Homomorphic equality testing: The authorized agents, with the operational key $(N, g_1, g_2 \ldots g_k, T)$, can perform equality testing to two ciphers. For each g_i, test whether

$$g_i^{|E(m_1) - E(m_2)|} = 1 \mod T$$

If any one of the g_i's fails the test, then $m_1 \neq m_2$. On the other hand, if all g_i's pass the test, then m_1 is most likely equal to m_2 with a neglect chance of being not equal, since

$$\begin{aligned} g_i^{|E(m_1) - E(m_2)|} &= (h_i^{2(P_3 - 1)})^{rP_1 + |m_1 - m_2|} \mod T \\ &= h_i^{r 2P_1 (P_3 - 1)} \mod T \quad if\ m_1 = m_2 \\ &= h_i^{r 2P_1 (P_3 - 1) \bmod \phi(T)} \mod T \\ &= h_i^{r 2P_1 (P_3 - 1) \bmod 2P_1 (P_3 - 1)} \mod T \\ &= 1 \end{aligned}$$

Security weakness: If an adversary knows a pair of ciphers of a same integer, i.e., $(E_1(m), E_2(m))$, he can compute the encryption key P_1 since $Diff = |E_1(m) - E_2(m)| = rP_1 \mod N$ and then the $\gcd(Diff, N) = P_1$. A special case of this attack is actually a known plaintext attack. That is, if the adversary knows a plain and a cipher pair $(m, E(m))$, he can perform the same gcd attack.

Applications may be suitable for this algorithm: Applications with highly discrete data with no repetition, e.g., high-precision scientific data.

Homomorphic encryption algorithm: Variant 2

Parameter setup: The Variant 2 algorithm has exactly the same system parameter setup as Variant except two additional parameters *w* and *z*. The selection of *w* and *z* also has impact to the selection of P_1.

1. *w* should be selected based on the application requirement. If the maximum integer required in an application is *Imax*, then *w* must be selected big enough so that $Imax < 2^w$. That is, *w* is the maximum bit size of all data in an application.
2. *z* is the number of random bits to be left padded to the data so that two encryptions of the same integer *m* are most likely to encrypt two different padded integers. Typically, *z* = 64 would be big enough for real world applications.
3. The padded integer *m'* is of the size *w*+*z* bits. Recall that $m < P_1$ in the Variant 1 algorithm. Thus, in Variant 2, we have $m' < P_1$. This implies that if the data owner would like to have *k* consecutive homomorphic multiplications on ciphertexts, then choosing P_1 with size at least (*k* + *1*)(*w* + *z*) bits.

Padding: We use an example to show how Variant 2 algorithm works. Let $m = 13$ and then the padded integer $m' = r_z \ldots r_2 r_1 \,|\, 0 \ldots 0 \,|\, 1101$, where | means concatenation and $m = 1001_2$ with leading 0's to the *w*-th bit, followed by *z* padded random bits. Thus, let an integer $R = r_z \ldots r_2 r_1$, we have

$$m' = R \times 2^w + m$$

Encryption: The encryption works he same as the Variant 1 algorithm, except it encrypts the padded integer *m'*, i.e.,

$$E(m') = rP_1 + m' \mod N$$

Decryption: Two mod operations are required.

$$D(E(m')) = (E(m') \bmod P_1) \bmod 2^w = m' \bmod 2^w = m$$

Homomorphic additions & multiplications: Same as those in the Variant 1 algorithm.

Equality testing: The equality testing algorithm for the Variant 1 ciphers does not work for the Variant 2 ciphers. No known equality testing for Variant 2 ciphers.

Security analysis: Currently no known attacks. However, the security of this Variant 2 homomorphic encryption requires formal proofs.

Applications may be suitable for this algorithm: Applications with no need to perform equality matching operations.