Fundamentals of Cryptography: Homomorphic Encryption

Instructed by *Wenfei Wu*

Due on Jan 10, 2020

Shucheng Chi, Xiaoqi Duan
YaoClass 70 2017012507, 2017011322

1 Introduction

Homomorphic encryption is a kind of encryption that allows direct computation on ciphertexts, and when decrypted these operations act as if they had been performed on the plaintext. As a result, it can be widely used for privacy-preserving computations. For example, in cloud computing users can upload the encrypted ciphertext and the server can perform computations without knowing the exact plaintext. The formal definition is as follows.

Definition 1 (Homomorphic Encryption.) An encryption scheme is said to be homomorphic for some operation $\oplus$, if there exists an operation $\otimes$ such that for any plaintext m_1, m_2, we have

$$m_1 \oplus m_2 = D(s, E(p, m_1) \otimes E(p, m_2))$$

where E is the encoding function, D is the decoding function, s is the secret key and p is the public key.

Usually we will try to implement homomorphic $+$ and $\times$ operations. If a scheme implements either homomorphic $+$ or $\times$, it is called Partially Homomorphic Encryption (PHE). If a scheme implements both $+$ and $\times$, then it's called Fully Homomorphic Encryption (FHE). The were numbers of PHE schemes before, but only in the recent 10 years had the FHE schemes come out. Briefly speaking, there are three generations of FHE divided by their methods, and we will introduce the third generation presented by Gentry, Sahai and Waters in 2013 [GSW13].

The structure of this report is as follows. In section 2, we introduce a PHE scheme that supports the integer comparision operation. In section 3, we introduce the learning with error problem, which plays an important role in the GSW scheme. In section 4 and 5, we briefly introduce the GSW scheme. In section 6, we summarize this report.

2 Secure Integer Comparison based on Partial Homomorphic Encryption

Consider the following task: two parties respectively holds some secret integers m_1 and m_2, and they want to compare them without leaking any information beyond the result of whether $m_1 \leq m_2$. In Yao's 1982 paper, this is called Millionaires' problem, in which two Millionaires want to compare their wealth without announcing the true number. [Yao82] gives the first protocol on this problem based on garbled circuits, a by-now standard protocol.

Here we want to solve Millionaires' problem within the scope of homomorphic encryption: a trusted third party holds the ciphertext of m_1, m_2, and it executes the task to calculate $[m_1 > m_2]$. (This value is 1 when

$m_1 > m_2$, and 0 otherwise.) In this process, the third party can only operate on the ciphertext. The solution provided here is from Fischlins work [Fis01].

2.1 Goldwasser-Micali cryptosystem (Goldwasser & Micali, 1982)

This protocol is based on Goldwasser-Micali cryptosystem, a well-known partial homomorphic encryption. We state the protocol here.

- **Keygen algorithm:** Find two large primes p, q, and $n = pq$. Select a quadratic nonresidue z. This is characterized by calculating the Legendre symbol $\left(\frac{z}{p}\right) = z^{\frac{p-1}{2}} \pmod p$ and $\left(\frac{z}{q}\right) = z^{\frac{q-1}{2}} \pmod q$. z is a quadratic nonresidue of n if and only if these two symbols are both zero. **The public key is** (n, z). **The secret key is** (p, q).
- **Encryption algorithm:** For a bit $m \in \{0, 1\}$, choose a random integer r, and $E(m) = z^m r^2$.
- **Decryption algorithm:** Determine whether c is quadratic residue of n. If so, $c = 1$, otherwise $c = 0$.

It has the partial homomorphic property:

$$m_1 + m_2 = D(k_s, E(k_p, m_1) \times E(k_p, m_2)). \tag{1}$$

2.2 Outline of the Secure Integer Comparison Protocol

The core solution to calculate $[x > y]$ is based on the following fact: consider what a machine does when comparing two integers x, y, given their binary presentation. Suppose $x = x_1 x_2 \cdots x_n$, $y = y_1 y_2 \cdots y_n$. The machines gives the result $x > y$, only when it finds that for some i, $x_1 = y_1, \cdots, x_i = y_i$, but $x_i = 1, y_i = 0$. As a result, we can rewrite $[x > y]$ as a logical conjunction of their bits as

$$[x > y] = \bigvee_{i=1}^{n} \left(x_i \wedge \neg y_i \wedge \bigwedge_{j=i+1}^{n} (x_j = y_j) \right) \tag{2}$$

If given the encryption $E(x), E(y)$, we can calculate $E(\neg x)$, $E([x = y])$, and $E(x \wedge y)$, then $E([x > y])$ can be calculated as a composition of them. In the following subsections, we provide methods to calculate $E(\neg x)$, $E([x = y])$, and $E(x \wedge y)$, then the problem is solved.

2.3 XOR and NOT property of Goldwasser-Micali cryptosystem

We can find that the Goldwasser-Micali cryptosystem satisfies:

- xor-property: $E(x) \times E(y) = E(x \oplus y)$.
- not-property: $E(x) \times z = E(\neg x)$.

Note that $[x = y]$ can be represented $\neg(x \oplus y)$, so $E([x = y])$ is also solved by a composition of xor and not.

2.4 AND encryption of Goldwasser-Micali cryptosystem

Getting the result of $E(x \wedge y)$ requires more efforts. In the first step, we modify the protocol as follows:

- **Keygen algorithm:** The same. **Public key** (n, z). **Secret key** (p, q).
- **Encryption algorithm:** When $m = 1$, the encryption of m is a series of $E(m, r)$. In other words, choose a large N (we will explain the choose of M later). choose N random integers $r_1, r_2, \cdots, r_N$, and $E(m) = (E(m, r_1), E(m, r_2), \cdots, E(m, r_N)) = (z{r_1}^2, z{r_2}^2, \cdots, z{r_N}^2)$. When $m = 0$, just output a series of N random numbers: $E(m) = (r_1, r_2, \cdots, r_N)$.

- **Decryption algorithm:** The ciphertext is $(c_1, c_2, \cdots, c_N)$. Determine whether c_i is quadratic residue of n. If all the c_i's are quadratic residue, $c = 1$, otherwise $c = 0$.
- **AND-property:** Given two ciphertexts $E(x) = (c_1, c_2, \cdots, c_N)$, $E(y) = (d_1, d_2, \cdots, d_N)$, the ciphertext of $x \wedge y$ can be calculated as

$$E(x \wedge y) = (c_1 d_1, c_2 d_2, \cdots, c_N d_N). \tag{3}$$

We know that for a random number, it has half probability to be a quadratic residue of n, and the other half not. Thus when decrypting the ciphertext of some $m = 1$, with probability 1 the decrypted plaintext is $m = 1$. When decrypting the ciphertext of some $m = 0$, there is a probability of 2^{-N} that the chosen random numbers are all quadratic residue, and for the other $1 - 2^{-N}$ probability, the decrypted plaintext is $m = 0$. So the error rate is 2^{-N}. Choose N to be large as 80 to 90, then the error rate can be negligible.

For the correctness of AND-property, we examine every possible case of the value of x and y. When x and y are both 1, which means $(c_1, c_2, \cdots, c_N), (d_1, d_2, \cdots, d_N)$ are all quadratic residue, their product $(c_1 d_1, c_2 d_2, \cdots, c_N d_N)$ must also be quadratic residues, so $D(c_1 d_1, c_2 d_2, \cdots, c_N d_N) = 1$.

When either x or y is 0, the decryption result should be 0 with probability $1 - 2^{-N}$. For this part, we need the knowledge from number theory that the Legendre symbol is multiplicative:

$$\left(\frac{xy}{p}\right) = \left(\frac{x}{p}\right)\left(\frac{y}{p}\right). \tag{4}$$

Consider the Legendre symbol of $(c_1 d_1, c_2 d_2, \cdots, c_N d_N)$. When a series of numbers is multiplied by a another series of random numbers, the result is actually a series of random numbers. Thus the error rate is $1 - 2^{-N}$.

2.5 Secure Integer Comparison Protocols

As a summarize of the above discussions, the protocol is given below.

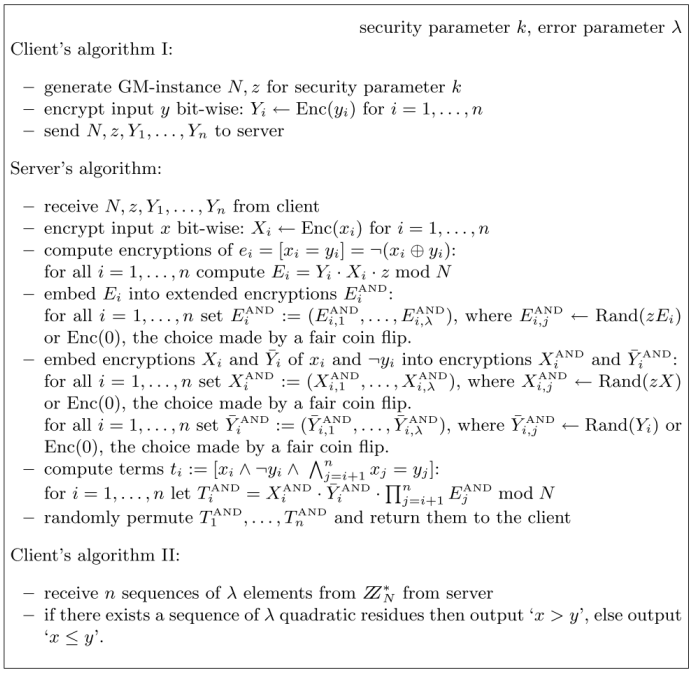

Figure 1: A Secure Integer Comparison Protocol

We should notice that some calculations in the protocol of figure 1 can be reduced. For the calculation of $\bigvee_{j=i+1}^{n}(x_j = y_j)$, if we store the values of $\bigvee_{j=i+2}^{n}(x_j = y_j)$, in each previous step, the only work needed is to calculate $E([x_{i+1} = y_{i+1}])$ and calculate $[x_{i+1} = y_{i+1}] \wedge \left(\bigvee_{j=i+2}^{n}(x_j = y_j)\right)$. There can also be some other similar optimizations. The optimized protocol is provided in figure 2.

security parameter k, error parameter λ

Optimized server algorithm:

- receive $N, z, Y_1, \ldots, Y_n$ from client
- embed input x into extended encryptions X_i^{AND}:
  for all $i = 1, \ldots, n$ let $X_i^{\text{AND}} := (X_{i,1}^{\text{AND}}, \ldots, X_{i,\lambda}^{\text{AND}})$, where $X_{i,j}^{\text{AND}} \leftarrow \text{Enc}(z^{1-x_i})$ or $\text{Enc}(0)$, the choice made by a fair coin flip.
- embed $[x_i = y_i]$ into extended encryptions E_i^{AND}:
  for all $i = 1, \ldots, n$ set $E_i^{\text{AND}} := (E_{i,1}^{\text{AND}}, \ldots, E_{i,\lambda}^{\text{AND}})$, where $E_{i,j}^{\text{AND}} := Y_j \cdot z^{x_i} \bmod N$ or $1 \in \text{QR}_N$, the choice made by a fair coin flip.
- compute extended encryptions P_i^{AND} of $p_i = \bigwedge_{j=i+1}^{n}[x_j = y_j]$:
  for $i = n-1, \ldots, 1$ let $P_i^{\text{AND}} := P_{i+1}^{\text{AND}} \cdot E_{i+1}^{\text{AND}} \bmod N$ where $P_n^{\text{AND}} := (1, \ldots, 1)$.
- embed encryptions $\bar{Y}_i$ of $\neg y_i$ into encryptions $\bar{Y}_i^{\text{AND}}$:
  for all $i = 1, \ldots, n$ set $\bar{Y}_i^{\text{AND}} := (\bar{Y}_{i,1}^{\text{AND}}, \ldots, \bar{Y}_{i,\lambda}^{\text{AND}})$, where $\bar{Y}_{i,j}^{\text{AND}} = Y_i$ or $1 \in \text{QR}_N$, the choice made by a fair coin flip.
- compute terms $t_i := [x_i \wedge \neg y_i \wedge \bigwedge_{j=i+1}^{n}[x_j = y_j]$:
  for $i = 1, \ldots, n$ let $T_i^{\text{AND}} := X_i^{\text{AND}} \cdot \bar{Y}_i^{\text{AND}} \cdot P_i^{\text{AND}} \bmod N$
- randomly permute $T_1^{\text{AND}}, \ldots, T_n^{\text{AND}}$ and return them to the client

Figure 2: Optimized Secure Integer Comparison Protocol (Fischlin 2001)

3 Learning with Errors

The learning-with-errors problem (LWE) is an important problem in lattice-based cryptography. It has two different versions:

Definition 2 (Search-LWE). Given a matrix A and a row vector b, satisfying $b = sA + \eta$ where η is some unknown noise vector. Search-LWE aims at finding s efficiently given A and b.

Definition 3 (Decision-LWE). Given a matrix A and a row vector b, it either satisfies $b = sA + \eta$, or that b is completely random. Decision-LWE aims at distinguishing this two cases.

In practice, denote A' be the matrix where b is appended as the last row of A, and b' be the vector where a -1 is appended at the end of s. This gives us a simplified formula $\eta = s'A'$. Both Search-LWE and Decision-LWE can be reduced from the GapSVP problem, which is believed to be a hard problem in lattice. This assumption is formally defined as follows:

Definition 4 (DLWE assumption). For parameters n, m, q, α, there is efficiently samplable ensemble ψ_n over pairs $(s \in \mathbb{Z}_q^n, A \in \mathbb{Z}_q^{n*m})$, such that

- A is pseudorandom over $\mathbb{Z}_q^{n*m}$.
- The l_∞ norm of $\eta = [sA]_q$ is bounded by αq with high probability, and $s_n = -1$.

4 Regev Scheme

Regev scheme is an additively homomorphic public-key encryption. Its security is based on the hardness of DLWE.

- **KeyGen(1^λ).** With a DLWE instance, draw a pair (s, A), output secret key $s \in \mathbb{Z}_q^n$ and public key $A \in \mathbb{Z}_q^{n*m}$.
- **Encrypt(A, b).** For public key A and plain bit $b \in \{0, 1\}$, random sample a vector $r \in \{0, 1\}^m$, output cipher text $u = [b[\frac{q}{2}](0, 0, \ldots, -1) + Ar]_q$.

- **Decrypt(s, u).** Compute $z = \langle s, u \rangle$. Output 0 if $|z| < \frac{q}{4}$ and 1 otherwise.

The correctness comes from the following fact:

$$\begin{aligned} \langle s, u \rangle &= b[\frac{q}{2}] + sAr \\ &= b[\frac{q}{2}] + \langle \eta, r \rangle \end{aligned}$$

Since $\langle \eta, r \rangle$ is small, with high probability we can recover the correct answer.

5 GSW scheme

5.1 Intuition

Consider two matrices C_1, C_2 that shares a same (left) eigenvector s with corresponding eigenvalues μ_1, μ_2. Then we have

$$\begin{aligned} sC_1 &= \mu_1 s \\ sC_2 &= \mu_2 s \\ s(C_1 + C_2) &= (\mu_1 + \mu_2)s \\ sC_1C_2 &= \mu_1\mu_2 s \end{aligned}$$

If we denote μ as the plaintext and C as the ciphertext, and the homomorphic addition and multiplication by

- $ADD(C_1, C_2) = C_1 + C_2$
- $MUL(C_1, C_2) = C_1C_2$,

then this mechanism satisfies fully homomorphism. However, this scheme is not safe since we can compute eigenvectors directly from C.

A natural idea is to add noise. Namely, we have

$$\begin{aligned} sC_1 &= \mu_1 s + \eta_1 \\ sC_2 &= \mu_2 s + \eta_2 \\ s(C_1 + C_2) &= (\mu_1 + \mu_2)s + (\eta_1 + \eta_2) \\ sC_1C_2 &= (\mu_1 s + \eta_1)C_2 \\ &= \mu_1\mu_2 s + (\eta_1 C_2 + \mu_1\eta_2) \end{aligned}$$

The idea for encryption and decryption with noise is very similar to Regev scheme. Here, the safety is not a problem, since it's literally an instance of LWE, which means one can't retrieve s given C and $\mu s + \eta$. However, the problem here is that the error term $\eta_1 C_2$ will no longer be small, which may lead to an error in decryption. To avoid this, we want C_2 to be "small', e.g. have only $0, 1$ elements.

To introduce the final mechanism, first introduce two operations. The function $G^{-1}(C)$ will expand an $n * m$ matrix to an $(nd) * m$ matrix, where $d = [\log q] + 1$ is the length of binary representation of an integer in C. The matrix G satisfies $GG^{-1}(C) = C$, namely it shrinks back the previous matrix. Normally the matrix G looks as follows:

$$G = \begin{bmatrix} 2^0 & 2^1 & \dots & 2^{d-1} & 0 & 0 & \dots & 0 & \dots & 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 & 2^0 & 2^1 & \dots & 2^{d-1} & \dots & 0 & 0 & \dots & 0 \\ \dots & & & & & & & & & & & & \\ 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 & \dots & 2^0 & 2^1 & \dots & 2^{d-1} \end{bmatrix}.$$

It's worth noticing that $G^{-1}(C)$ has smaller norm than original C. Hence, what we want is to use $G^{-1}(C_2)$ in the homomorphic multiplication instead of C_2. To maintain the property of homomorphism, we slightly change our definition of secret key. Instead of choosing a eigen vector of C, here we choose s such that

$$sC = \mu sG + \eta$$

The only difference is we append the matrix G on the right hand side. Now we define homomorphic addition and multiplication correspondingly:

- $ADD(C_1, C_2) = C_1 + C_2$ (i.e. remains the same). We have $s(C_1 + C_2) = (\mu_1 + \mu_2)sG + (\eta_1 + \eta_2)$ so the error is small.
- $MUL(C_1, C_2) = C_1 G^{-1}(C_2)$. We have

$$\begin{aligned} s(C_1 G^{-1}(C_2)) &= (\mu_1 sG + \eta_1)G^{-1}(C_2) \\ &= \mu_1 sC_2 + \eta_1 G^{-1}(C_2) \\ &= \mu_1\mu_2 sG + (\mu_1\eta_2 + \eta_1 G^{-1}(C_2)) \end{aligned}$$

Still, we can decrypt similar to Regev scheme. Now let's look at the error. Since $G^{-1}(C_2)$ consists of only 0 and 1, the error $\eta_1 G^{-1}(C_2)$ can be controlled.

Besides addition and multiplication, we can also define the homomorphic NAND operation as follows:

- $NAND(C_1, C_2) = [G - C_1 G^{-1}(C_2)]_q$.

The correctness comes from

$$\begin{aligned} sNAND(C_1, C_2) &= sG - sC_1 G^{-1}(C_2) \\ &= (1 - \mu_1\mu_2)sG - (\mu_1\eta_2 + \eta_1 G^{-1}(C_2)) \end{aligned}$$

where $1 - \mu_1\mu_2$ is exactly $NAND(\mu_1, \mu_2)$. This version of GSW offers a homomorphic NAND gate which can be used to construct any kind of circuit, since NAND is a universal gate. Here, we regard the computation process as running a circuit with depth i. We can also analyze the error:

$$||\eta'||_\infty \le ||\eta_2||_\infty + nd||\eta_1||_\infty \le (nd + 1)\max\{||\eta_1||_\infty, ||\eta_2||_\infty\}$$

Formally, after i levels of circuit, the error $||\eta||_\infty$ is bounded by $nm(nd + 1)^i$.

5.2 Mechanism

The above are the intuitions of GSW scheme. Now we represent the full construction:

- **KeyGen**$(1^\lambda, 1^\tau)$**.** First find a DLWE instance with $\alpha q = n$, and $nm(nd + 1)^{\tau+1} < \frac{q}{4}$. Draw a pair (s, A) from it, output secret key $s \in \mathbb{Z}_q^n$ (notice $s_n = -1$) and public key $A \in \mathbb{Z}_q^{n*m}$.
- **Encrypt**(A, b)**.** For public key A and plain bit $b \in \{0, 1\}$, random sample a matrix $R \in 0, 1^{m*nd}$, output cipher text $C = [bG + AR]_q$.

- **Evaluate(Π, C).** This is the homomorphic computation of circuit Π on ciphertext C. Just treat it as several combination of NAND gates and for each gate $NAND(C_1, C_2)$ output $[G - C_1 G^{-1}(C_2)]_q$.
- **Decrypt(s, u).** Let $w = -[\frac{q}{2}](0, \ldots, 0, 1) \in Z_1^n$. Note $\langle s, w \rangle = \frac{q}{2}$. Compute $z = [sCG^{-1}(w)]_q$. Output 0 if $|z| < \frac{q}{4}$ and 1 otherwise.

The correctness comes from the following:

$$\begin{aligned} z &= sCG^{-1}(w) \\ &= (bsG + \eta)G^{-1}(w) \\ &= b\langle s, w \rangle + \langle \eta, G^{-1}(w) \rangle \\ &= b\frac{q}{2} + \eta' \end{aligned}$$

The choice of safety parameters guarantees $||\eta'||$ won't exceed $\frac{q}{4}$ after running on the circuit, hence we will get the correct answer.

It's worth noticing that no FHE schemes can guarantee CI or CCA security, since we can always construct a valid ciphertext via the homomorphic operation. On the other hand, the following theorem states that GSW scheme is semantic secure, hence CPA secure.

Theorem 1 (Semantic Secure.) The GSW scheme is semantically secure under DLWE assumption with $\alpha = \frac{n}{q}$ and $m > 1 + 2n(2 + d)$.

Proof Idea. We already know A is pseudorandom by DLWE assumption. We first prove (A, RA) is indistinguishable with (U_1, U_2) where U_1, U_2 are uniformly drawn random matrices. The main idea is that we can treat RA as a universal hash function on A so only except small probability the outcome RA will be close to the uniform distribution. Since (A, RA) is indistinguishable, so will $(A, RA + b \cdot G)$ indistinguishable from truly random matrices. Hence we don't have any advantage on b. □

5.3 Bootstrapping

The above mechanism only works for a circuit of a maximum depth, so it's only a "somewhat" fully homomorphic encryption. However, we can use the idea of bootstrapping to reduce the noise during the process.

The general idea is when we get a ciphertext c with a relative high noise, we first decrypt it (secretly) to get the plaintext, then encrypt it again to get a new ciphertext c'. Since c' is directly encrypted from the plaintext, it contains less noise than the original ciphertext. Seeing from outside, it looks as if we have "reduced" the noise in the ciphertext.

More formally, define $D_c(sk)$ be the circuit that decrypts cipher text c taken secret key sk as an input (here c is hardwired into the circuit). If for each possible cihper text pair (c_1, c_2), we can design a circuit $D^*_{c_1,c_2}(sk) = NAND(D_{c_1}(sk), D_{c_2}(sk))$, such that this circuit has depth no more than the depth limit required by the encryption mechanism, then we call this mechanism "bootstrappable". Here $D^*_{c_1,c_2}$ acts as a black box to reduce the noise.

Theorem 2. The GSW scheme is bootstrappable.

The proof can be found in [Hal17].

6 Conclusion

In this report, we introduced the concept of homomorphic encryption. We also introduced a PHE scheme for comparing integers and a (somewhat) FHE scheme that can be bootstrapped to get a truly FHE scheme. However, one major problem for all FHE schemes is that the cost of bootstrapping is relatively high, making it not efficient enough for common usage. Therefore, there are a number of papers focusing on improving

the bootstrapping process, like [AP13], [DD15] and [CGGI17]. Anyway, homomorphic encryption is an interesting and promising area that remains to be explored.

References

Yao, Andrew. C. Protocols for secure computations. 23rd annual symposium on foundations of computer science (sfcs 1982). IEEE, 1982.

Fischlin, Marc. A cost-effective pay-per-multiplication comparison method for millionaires. Cryptographers Track at the RSA Conference. Springer, Berlin, Heidelberg, 2001.

Bourse, Florian, Olivier Sanders, and Jacques Traor. Improved Secure Integer Comparison via Homomorphic Encryption. IACR Cryptology ePrint Archive 2019 (2019): 427.

C. Gentry, A. Sahai, and B. Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In R. Canetti and J. A. Garay, editors, Advances in Cryptology - CRYPTO 2013, Part I, pages 75-92. Springer, 2013.

Shai Halevi. Homomorphic Encryption. Tutorials on the Foundations of Cryptography. Information Security and Cryptography. Springer International Publishing, 2017.

Jacob Alperin-Sheriff and Chris Peikerty. Practical Bootstrapping in Quasilinear Time. CRYPTO 2013.

Lo Ducas and Daniele Micciancio. FHEW: Bootstrapping Homomorphic Encryption in less than a second. EUROCRYPT 2015.

Ilaria Chillotti et al. TFHE: Fast Fully Homomorphic Encryption over the Torus. AsiaCrypt 2017.