

Supplementary Lecture

The TFHE Scheme

SEC 595: Encrypted Computing

Roadmap

Lecture 15, Section 5 sketched TFHE (Chillotti, Gama, Georgieva and Izabachène, 2016): a gate is a public linear combination followed by one bootstrapping, and bootstrapping is a blind rotation built from GSW-type selectors. This lecture makes every step of that sketch precise and runs one bootstrapping on a toy instance. Section 1 fixes the message layout. Section 2 defines the three ciphertext types TFHE uses. Section 3 gives the external product and the CMux gate. Section 4 assembles bootstrapping in four stages. Section 5 turns it into a lookup-table evaluator. Section 6 works an instance, and Section 7 gives sizes and costs.

Notation. In TFHE, n is the LWE dimension and N the ring degree. The ring degree that Lectures 10–12 called n is called N here.

1. The Torus and the Message Layout

1.1 The Discretized Torus

TFHE is stated over the torus $\mathbb{T} = \mathbb{R}/\mathbb{Z}$, the reals modulo 1. Implementations store $x \in \mathbb{T}$ as the integer $\lfloor qx \rfloor \bmod q$ with $q = 2^{32}$ or 2^{64} , so the torus becomes $\mathbb{Z}_q$ and addition on $\mathbb{T}$ becomes addition modulo q . Two torus elements cannot be multiplied; only integer multiples kx are defined. That is enough, because every TFHE operation is a sum of integer multiples of ciphertexts. From here on we work in $\mathbb{Z}_q$, with centered representatives (Lecture 1).

1.2 Encoding

The layout is BFV's (Lecture 10, Section 4.1). Fix a message modulus p dividing q and let $\Delta = q/p$. A message $m \in \{0, \dots, p/2 - 1\}$ is placed at Δm : in the top bits, with the noise below it. The top bit of the message space is kept at zero; it is the *padding bit*, and Section 5.2 explains why bootstrapping needs it. Decryption is correct while the noise is below $\Delta/2$.

For example, with $q = 1024$ and $p = 8$, $\Delta = 128$. The messages 0, 1, 2, 3 sit at 0, 128, 256, 384, the upper half $[512, 1024)$ is left empty, and the noise must stay in $(-64, 64)$.

2. Three Ciphertext Types

Definition 1 (LWE ciphertext). With a binary key $s \in \{0, 1\}^n$, an encryption of m is $(a, b) \in \mathbb{Z}_q^{n+1}$ with a uniform and

$$b = \langle a, s \rangle + \Delta m + e.$$

The **phase** is $\varphi = b - \langle a, s \rangle = \Delta m + e$, and decryption returns $\lfloor \varphi / \Delta \rfloor \bmod p$.

The phase is linear: adding ciphertexts or multiplying by an integer does the same to the phase. This is the only ciphertext that travels between gates.

Definition 2 (GLWE ciphertext). Let $R_q = \mathbb{Z}_q[X]/(X^N + 1)$ (Lecture 3) and $S \in R_q$ have binary coefficients. An encryption of $M \in R_q$ is $(A, B) \in R_q^2$ with A uniform and $B = A \cdot S + M + E$; its phase is $B - A \cdot S = M + E$.

This is ring-LWE, as in BFV. TFHE uses it to carry a whole polynomial: the lookup table of Section 4.2. (General GLWE uses k key polynomials; $k = 1$ throughout this lecture.)

Definition 3 (GGSW ciphertext). Let $g = (1, \beta, \dots, \beta^{\ell-1})$ with $\beta^\ell = q$ be the gadget vector (Lecture 4) and $G = \begin{pmatrix} g^\top & 0 \\ 0 & g^\top \end{pmatrix}$, a $2\ell \times 2$ matrix. An encryption of $m \in \mathbb{Z}$ is

$$C = Z + mG \in R_q^{2\ell \times 2},$$

where each row of Z is a GLWE encryption of 0 under S .

This is GSW (Lecture 13) with ring entries and a GLWE row format.

Type	Encrypts	Size (elements of $\mathbb{Z}_q$)	Role
LWE	one message m	$n + 1$	data between gates
GLWE	a polynomial M	$2N$	the accumulator
GGSW	one integer, usually a key bit	$4\ell N$	the bootstrapping key

3. The External Product and CMux

3.1 The External Product

Definition 4 (External product). For a GGSW ciphertext C and a GLWE ciphertext $c = (A, B)$,

$$C \boxdot c = G^{-1}(c) \cdot C,$$

where $G^{-1}(c) \in R_q^{1 \times 2\ell}$ lists the balanced base- β digits of the coefficients of A and then of B , each digit in $[-\beta/2, \beta/2]$.

Proposition 1. *If C encrypts m with row errors E_C , then*

$$\text{phase}(C \boxdot c) = m \cdot \text{phase}(c) + G^{-1}(c) \cdot E_C, \quad \|G^{-1}(c) \cdot E_C\|_\infty \leq 2\ell \cdot N \cdot \frac{\beta}{2} \cdot \|E_C\|_\infty.$$

Proof. The phase of a row (A', B') is $(A', B') \cdot t$ with $t = (-S, 1)^\top$. Then $Zt = E_C$ and $Gt = (-gS, g)^\top$, so $G^{-1}(c)Gt = -AS + B = \text{phase}(c)$. Each of the 2ℓ products $d_i \cdot E_{C,i}$ has norm at most $N \cdot \frac{\beta}{2} \cdot \|E_C\|_\infty$ (Lecture 3). $\square$

With $m \in \{0, 1\}$ the noise of c is either passed through unchanged or erased, and the GGSW adds a term that does not depend on c . This is GSW's asymmetry (Lecture 13, Section 5.4; Lecture 15, Section 4.1), with the GLWE ciphertext in the decomposed slot.

3.2 CMux

Definition 5 (CMux). For a GGSW ciphertext C of a bit and GLWE ciphertexts d_0, d_1 ,

$$\text{CMux}(C, d_1, d_0) = C \boxplus (d_1 - d_0) + d_0.$$

Proposition 2. *If C encrypts $x \in \{0, 1\}$ with row errors bounded by E_{BK} , the output has the phase of d_x plus an error of norm at most $\max(\|e_0\|_\infty, \|e_1\|_\infty) + 2\ell N \frac{\beta}{2} E_{\text{BK}}$.*

Proof. By Proposition 1, the phase is $x(\varphi_1 - \varphi_0) + \varphi_0 + G^{-1}(d_1 - d_0)E_C$, which is φ_x plus the stated term. $\square$

CMux is an *if-then-else* on an encrypted bit: it returns one of two ciphertexts without revealing which.

4. Bootstrapping

Bootstrapping takes an LWE ciphertext under s (length n) with any noise below a threshold and returns an LWE ciphertext under s with fixed small noise. It has four stages:

$$\text{LWE}_s \xrightarrow{\text{mod switch}} \text{integers mod } 2N \xrightarrow{\text{blind rotation}} \text{GLWE}_S \xrightarrow{\text{extract}} \text{LWE}_S \xrightarrow{\text{key switch}} \text{LWE}_s.$$

4.1 Modulus Switching

Rescale every coordinate from modulus q to modulus $2N$:

$$\tilde{a}_i = \lfloor 2Na_i/q \rfloor \bmod 2N, \quad \tilde{b} = \lfloor 2Nb/q \rfloor \bmod 2N, \quad \tilde{\varphi} = \tilde{b} - \sum_i \tilde{a}_i s_i \bmod 2N.$$

Then $\tilde{\varphi} \approx 2N\varphi/q$. Each rounding is off by at most $\frac{1}{2}$, so

$$\left| \tilde{\varphi} - \frac{2N}{q}\varphi \right| \leq \frac{1}{2} \left(1 + \sum_i s_i \right) \leq \frac{n+1}{2}.$$

This is a worst case; the rounding errors are independent, so in practice the sum grows like $\sqrt{n}$, not n . The phase now has only $2N$ possible values. Each message occupies a **box** of $w = 2N/p$ consecutive values, so N must be large compared with p .

4.2 Rotation and the Test Polynomial

Proposition 3 (Negacyclic rotation). *For $v \in R_q$ and $0 \leq k < 2N$, the constant coefficient of $X^{-k}v$ is v_k if $k < N$ and $-v_{k-N}$ if $N \leq k < 2N$.*

Proof. For $k < N$, X^{-k} moves coefficient k to position 0. For $k \geq N$, $X^{-k} = X^{-N}X^{-(k-N)} = -X^{-(k-N)}$, since $X^N = -1$. $\square$

So rotating a polynomial by $X^{-\tilde{\varphi}}$ reads its coefficient number $\tilde{\varphi}$. Store a table in the coefficients. Given f on messages, set

$$v_j = \Delta' f(\lfloor j/w \rfloor), \quad 0 \leq j < N,$$

where Δ' is the output scaling. Box m is centered on mw , so add $w/2$ to $\tilde{b}$ first. A phase in box m then lands in $[mw, (m+1)w)$, and the constant coefficient of $X^{-\tilde{\varphi}}v$ is $\Delta' f(m)$ for every $m < p/2$.

4.3 Blind Rotation

Definition 6 (Bootstrapping key and blind rotation). The bootstrapping key is $\text{BK}_i = \text{GGSW}_S(s_i)$ for $i = 1, \dots, n$. Start from the trivial GLWE ciphertext $\text{ACC} = (0, X^{-\tilde{b}}v)$ and, for $i = 1, \dots, n$, set

$$\text{ACC} \leftarrow \text{CMux}(\text{BK}_i, X^{\tilde{a}_i} \text{ACC}, \text{ACC}).$$

Proposition 4. After n steps ACC is a GLWE encryption of $X^{-\tilde{\varphi}}v$, with noise at most $n \cdot 2\ell N \frac{\beta}{2} E_{\text{BK}}$.

Proof. Multiplying both parts of a GLWE ciphertext by X^k multiplies its phase by X^k and only permutes and negates the noise coefficients. Step i therefore multiplies the encrypted polynomial by $X^{\tilde{a}_i s_i}$, and after n steps the exponent is $-\tilde{b} + \sum_i \tilde{a}_i s_i = -\tilde{\varphi}$. The trivial start has no noise, and each CMux adds at most $2\ell N \frac{\beta}{2} E_{\text{BK}}$. $\square$

The rotation is *blind*: the evaluator rotates by $\tilde{\varphi}$ without ever learning $\tilde{\varphi}$. The bound does not involve the input's noise. The input's noise only decides which box $\tilde{\varphi}$ lands in.

4.4 Sample Extraction

Proposition 5 (Sample extraction). Let (A, B) be a GLWE ciphertext under S . Then

$$a' = (A_0, -A_{N-1}, -A_{N-2}, \dots, -A_1), \quad b' = B_0$$

is an LWE ciphertext under $s' = (S_0, S_1, \dots, S_{N-1})$ whose phase is the constant coefficient of the GLWE phase.

Proof. In R_q , $(AS)_0 = A_0 S_0 - \sum_{j=1}^{N-1} A_j S_{N-j}$, since $X^j X^{N-j} = -1$. Reindexing gives $\langle a', s' \rangle$. $\square$

Extraction is free: it adds no noise and needs no key material.

Example 1 (Extracting, $N = 4, q = 1024$). Take $S = 1 + X^2 + X^3$, $A = 3 + 5X + 7X^2 + 2X^3$, and $M + E = 130 - 2X + X^2$. Then $AS = -9 - 4X + 8X^2 + 10X^3$, so $B = 121 + 1018X + 9X^2 + 10X^3$. Extraction gives $a' = (3, -2, -7, -5)$, $b' = 121$, and with $s' = (1, 0, 1, 1)$,

$$b' - \langle a', s' \rangle = 121 - (3 - 7 - 5) = 130,$$

the constant coefficient of $M + E$. $\checkmark$

4.5 Key Switching

The extracted ciphertext is under s' , of length N , but the next gate needs s , of length n . With a base β_{ks} and ℓ_{ks} digits, publish $\text{KSK}_{j,t} = \text{LWE}_s(s'_j \beta_{ks}^t)$, an LWE encryption under s of the integer $s'_j \beta_{ks}^t$, for $0 \leq j < N$ and $0 \leq t < \ell_{ks}$.

Definition 7 (Key switching). Write $a'_j = \sum_t d_{j,t} \beta_{ks}^t$ with balanced digits and output

$$(0, b') - \sum_{j,t} d_{j,t} \text{KSK}_{j,t}.$$

The phase is $b' - \sum_j a'_j s'_j$ minus $\sum d_{j,t} e_{j,t}$, so key switching adds at most $N \ell_{ks} \frac{\beta_{ks}}{2} E_{ks}$. This is the gadget trick of BFV relinearization (Lecture 10, Section 7.3), used here to change the key rather than its degree.

Proposition 6 (Bootstrapping correctness). *Bootstrapping returns an encryption of $f(m)$ if*

- (i) *the scaled input noise plus the modulus-switching error is below $w/2$, so $\tilde{\varphi}$ lands in box m ;*
and
- (ii) *the blind-rotation noise plus the key-switching noise is below $\Delta'/2$.*

Condition (i) is the only one that involves the input. Whatever the input noise was, the output noise is the fixed amount in (ii). TFHE parameters are chosen so that (i) fails with negligible probability.

5. Programmable Bootstrapping

5.1 Any Lookup Table

The test polynomial v is public and arbitrary. One bootstrapping both refreshes the noise and applies f . Comparisons, thresholds, ReLU, and any other function of a few-bit message cost exactly one bootstrapping. In BFV, BGV and CKKS these functions need high-degree polynomial approximations. Gate bootstrapping (Lecture 15, Section 5.1) is the special case of a one-bit table.

5.2 The Padding Bit

By Proposition 3, a phase in the upper half of the torus ($\tilde{\varphi} \geq N$) returns $-v_{\tilde{\varphi}-N}$, not a table entry. Two remedies:

1. keep messages in $[0, p/2)$, so the top bit (the padding bit) is always 0, and f can be any function; or
2. use only *negacyclic* functions, $f(m+p/2) = -f(m)$, as the sign function of gate bootstrapping is.

Linear operations can carry into the padding bit. Adding two messages below $p/4$ is safe; adding two messages below $p/2$ may not be. The circuit designer tracks this.

5.3 Multiplying Two Ciphertexts

TFHE has no direct product of two LWE ciphertexts. Multiplication goes through a table. If $x, y < 2^k$ and the message space holds $2k$ bits, the linear combination $2^k c_x + c_y$ encrypts $2^k x + y$, and one bootstrapping with the table $2^k x + y \mapsto xy \bmod 2^k$ returns the product. The same idea evaluates any function of two small inputs. Larger integers are split into small blocks, with bootstrappings to propagate carries.

6. A Worked Instance

Example 2 (One programmable bootstrapping). *Parameters.* $q = 1024$, $p = 8$, $\Delta = \Delta' = 128$; LWE key $s = (1, 0, 1)$, so $n = 3$; $N = 16$, so $2N = 32$ and $w = 2N/p = 4$. The function is $f(m) = [m \geq 2]$, so

$$v = 128 \left(\underbrace{0, \dots, 0}_8, \underbrace{1, \dots, 1}_8 \right).$$

Input. Encrypt $m = 2$ with $a = (333, 790, 150)$ and $e = 5$: $b = 333 + 150 + 256 + 5 = 744$, with phase 261.

Modulus switch. $\tilde{a} = \lfloor a/32 \rfloor = (10, 25, 5)$ and $\tilde{b} = \lfloor 744/32 \rfloor = 23$. Then $\tilde{\varphi} = 23 - (10 + 5) = 8$, against the exact value $32 \cdot 261/1024 = 8.16$. Adding $w/2 = 2$ gives 10, which lies in box $\lfloor 10/4 \rfloor = 2 = m$. ✓

Blind rotation. The accumulator starts at $X^{-25}v$, since $\tilde{b} + w/2 = 25$. Track only the exponent of X :

$$-25 \xrightarrow{s_1=1: +10} -15 \xrightarrow{s_2=0: \text{no change}} -15 \xrightarrow{s_3=1: +5} -10.$$

The constant coefficient of $X^{-10}v$ is $v_{10} = 128 = \Delta' f(2)$.

Encrypted run. With $\beta = 4$, $\ell = 5$, $\beta_{ks} = 4$, $\ell_{ks} = 5$ and key errors in $\{-1, 0, 1\}$, the accumulator's constant coefficient has phase 100 (noise -28). Extraction keeps 100, and key switching gives 95 (noise -33). Decryption returns $\lfloor 95/128 \rfloor = 1 = f(2)$. ✓

Noise independence. With the same a and input noise $e = \pm 9$, the outputs for $m = 0, 1, 2, 3$ are 0, 0, 1, 1: the output does not depend on the input noise, only on its box.

At these toy sizes the output noise (-33) exceeds the input noise (5). Real parameters make the fixed output noise a small fraction of $\Delta'/2$, so the output is ready for the next gate.

7. Parameters and Cost

The original parameter set uses $n = 500$, $N = 1024$, $k = 1$, $q = 2^{32}$ and $\ell = 2$; the reference library now uses $n = 630$ for a higher security margin.

Example 3 (Key size and work). (a) *Bootstrapping key.* n GGSW ciphertexts, each $2\ell \times 2$ polynomials of N coefficients of 4 bytes:

$$500 \cdot (2 \cdot 2) \cdot 2 \cdot 1024 \cdot 4 = 16\,384\,000 \text{ bytes} \approx 16 \text{ MB},$$

against about 1 GB for FHEW, the scheme TFHE improved on.

(b) *Work.* Blind rotation is n external products, each 2ℓ digit polynomials times a row of 2 polynomials: $4\ell = 8$ polynomial products, done with the FFT. That is $500 \cdot 8 = 4000$ products of degree-1024 polynomials per bootstrapping. The reported time is about 13 ms on one core, against 690 ms for FHEW.

Every gate costs one bootstrapping, so TFHE trades throughput for flexibility. BFV, BGV and CKKS process thousands of slots per operation. TFHE processes one small message per bootstrapping, but with no depth limit and with any function of that message available at the same price. Lecture 15, Section 5.4 compares the schemes.

Security rests on LWE (for s and the key-switching key), ring-LWE (for S and the bootstrapping key), and circular security, since BK encrypts s under S and KSK encrypts S under s (Lecture 14, Section 4).

Summary

1. TFHE uses three ciphertexts: LWE for data, GLWE for the accumulator, GGSW for the bootstrapping key. Messages sit at Δm with a padding bit on top.

2. The external product $G^{-1}(c) \cdot C$ multiplies a phase by a GGSW bit with additive noise; CMux turns it into an encrypted if-then-else.
3. Bootstrapping is modulus switching to $2N$, blind rotation of a test polynomial by $X^{-\tilde{\varphi}}$ through n CMux steps, sample extraction, and key switching.
4. The output noise is fixed and independent of the input noise. The input only has to land in the right box of width $w = 2N/p$.
5. The test polynomial is a lookup table: one bootstrapping refreshes the noise and applies any function of a small message. The padding bit avoids the negacyclic sign flip.

Exercises

1. Repeat Example 2 for $m = 3$ with the same a and $e = 5$: compute b , $\tilde{b}$, $\tilde{\varphi}$, the box, the rotation exponents and the output.
2. With v from Example 2, give the constant coefficient of $X^{-k}v$ for $k = 3, 10, 26$.
3. In Example 2, encrypt $m = 6$, which uses the padding bit. Show that $\tilde{\varphi} + w/2$ lands in $[24, 28)$ and that bootstrapping returns -128 , which decodes to 7, not $f(6)$.
4. For $n = 630$, $N = 1024$ and $p = 8$, compare the worst-case modulus-switching error $(n+1)/2$ with $w/2$. Why do implementations rely on the $\sqrt{n}$ behaviour instead?
5. Compute the bootstrapping-key size for $n = 630$, $N = 1024$, $\ell = 3$, $q = 2^{32}$.
6. Write the test polynomial for $f(m) = m^2 \bmod 4$ with $p = 8$, $N = 16$.
7. Using Section 5.3, give a linear combination of two encrypted bits and a table that computes their AND with one bootstrapping. How many bits of message space does it need?
8. In blind rotation, the accumulator goes in the decomposed slot of the external product and the key in the other. What happens to the noise bound of Proposition 4 if the roles are swapped?