

Lecture 15

The TFHE Scheme

SEC 595: Encrypted Computing

Roadmap

Lecture 14 bootstrapped GSW: correct, but rare and expensive, so the schemes of Lectures 10–12 spend a noise budget between bootstraps. TFHE (Chillotti, Gama, Georgieva and Izabachène, 2016) turns this around: it keeps ciphertexts small, makes bootstrapping fast, and bootstraps after *every* gate, so there is no depth limit. Section 1 fixes the message layout and Section 2 the ciphertexts. Section 3 gives the GSW-type selector, CMux. Section 4 builds bootstrapping and Section 5 turns it into a lookup-table evaluator. Section 6 runs one instance and Section 7 compares TFHE with the other schemes.

Notation. n is the LWE dimension and N the ring degree.

1. The Torus and the Message Layout

TFHE is stated over the torus $\mathbb{T} = \mathbb{R}/\mathbb{Z}$; implementations store it as $\mathbb{Z}_q$ with $q = 2^{32}$ or 2^{64} , and we work in $\mathbb{Z}_q$ throughout. Only additions and integer multiples are needed.

The layout is BFV's (Lecture 10). With a message modulus $p \mid q$ and $\Delta = q/p$, a message $m \in \{0, \dots, p/2 - 1\}$ sits at Δm , with the noise below it. The top half of $\mathbb{Z}_q$ is left empty: this **padding bit** is needed in Section 5. Decryption is correct while the noise is below $\Delta/2$. For example, $q = 1024$, $p = 8$ gives $\Delta = 128$: messages $0, \dots, 3$ at $0, 128, 256, 384$, noise in $(-64, 64)$.

2. Three Ciphertexts

Definition 1 (LWE ciphertext). With a binary key $s \in \{0, 1\}^n$, an encryption of m is (a, b) with a uniform in $\mathbb{Z}_q^n$ and $b = \langle a, s \rangle + \Delta m + e$. The **phase** is $\varphi = b - \langle a, s \rangle = \Delta m + e$, and decryption returns $\lfloor \varphi / \Delta \rfloor$.

The phase is linear, so adding ciphertexts adds messages and noises, for free. Two more types serve inside bootstrapping. A **GLWE** ciphertext is ring-LWE over $R_q = \mathbb{Z}_q[X]/(X^N + 1)$ (Lecture 10): (A, B) with $B = AS + M + E$, carrying a whole polynomial M . A **GGSW** ciphertext is GSW (Lecture 13) with ring entries: $C = Z + mG$, a $2\ell \times 2$ matrix whose rows of Z are GLWE encryptions of 0.

Type	Encrypts	Role
LWE	one message	data between gates
GLWE	a polynomial	the accumulator
GGSW	one key bit	the bootstrapping key

3. The External Product and CMux

Definition 2 (External product and CMux). For a GGSW ciphertext C of m and a GLWE ciphertext c , the external product is $C \boxtimes c = G^{-1}(c) \cdot C$, with G^{-1} the gadget decomposition of Lecture 4 in base β with ℓ digits. For C encrypting a bit x ,

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

Proposition 1. $C \boxtimes c$ encrypts m times the message of c , with noise at most $|m| \cdot (\text{noise of } c) + 2\ell N^{\frac{\beta}{2}} E_C$. Hence $\text{CMux}(C, d_1, d_0)$ returns an encryption of the message of d_x , adding only a bounded amount of noise.

This is the external product of Lecture 13 (Section 6.3) over the ring: GSW's asymmetry, with a GLWE ciphertext in the decomposed slot. 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 with noise below a threshold and returns an LWE ciphertext of the same message, or of a function of it, with fixed small noise:

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

Modulus switching. Round every coordinate to modulus $2N$: $\tilde{a}_i = \lfloor 2Na_i/q \rfloor$, $\tilde{b} = \lfloor 2Nb/q \rfloor$, so that $\tilde{\varphi} = \tilde{b} - \sum_i \tilde{a}_i s_i \bmod 2N$ approximates $2N\varphi/q$. Each message now owns a **box** of $w = 2N/p$ consecutive values of $\tilde{\varphi}$.

The test polynomial. For $0 \leq k < 2N$, the constant coefficient of $X^{-k}v$ is v_k if $k < N$ and $-v_{k-N}$ if $k \geq N$, since $X^N = -1$. So rotating a polynomial by $X^{-\tilde{\varphi}}$ reads its coefficient $\tilde{\varphi}$. Store a table in the coefficients:

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

and add $w/2$ to $\tilde{b}$ so that each box is centred on its message.

Definition 3 (Blind rotation). The bootstrapping key is $\text{BK}_i = \text{GGSW}_S(s_i)$, $i = 1, \dots, n$. Start from $\text{ACC} = (0, X^{-\tilde{b}}v)$ and for each i set

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

Step i multiplies the accumulator by $X^{\tilde{a}_i s_i}$, so at the end ACC encrypts $X^{-\tilde{\varphi}}v$, whose constant coefficient is $\Delta f(m)$. The rotation is *blind*: the evaluator rotates by an amount it never learns. Each CMux adds a bounded amount of noise, and the input's noise only decides which box $\tilde{\varphi}$ lands in.

Extraction and key switching. The constant coefficient of a GLWE ciphertext (A, B) is the LWE ciphertext $a' = (A_0, -A_{N-1}, \dots, -A_1)$, $b' = B_0$ under the coefficients of S ; this costs nothing. A key switch, the gadget trick of relinearization (Lecture 10), brings it back to the key s .

Proposition 2 (Bootstrapping correctness). *Bootstrapping returns an encryption of $f(m)$ if the input noise plus the rounding of the modulus switch keeps $\tilde{\varphi}$ in box m , and the noise added by blind rotation and key switching stays below $\Delta/2$.*

The output noise is fixed; it does not depend on the input noise.

5. Programmable Bootstrapping

Any table. The test polynomial is public and arbitrary, so one bootstrapping both refreshes the noise and applies any function f of a few-bit message: comparisons, thresholds, ReLU. In BFV, BGV and CKKS these need high-degree polynomial approximations.

Gates. For bits m_1, m_2 , the sum $c_1 + c_2$ encrypts $m_1 + m_2 \in \{0, 1, 2\}$, and one bootstrapping with the right table gives any gate:

$m_1 + m_2$	AND	OR	XOR	NAND
0	0	0	0	1
1	0	1	1	1
2	1	1	0	0

NAND is universal, so any Boolean circuit runs gate by gate, each gate leaving a fresh ciphertext. (The original TFHE encodes bits as $\pm \frac{1}{8}$ on the torus; the idea is the same.)

The padding bit. A phase in the upper half ($\tilde{\varphi} \geq N$) returns $-v_{\tilde{\varphi}-N}$, the negated table entry. So messages are kept below $p/2$, and the circuit designer makes sure sums do not spill into the padding half.

Multiplication. TFHE has no direct product of two LWE ciphertexts. For $x, y < 2^k$, the combination $2^k c_x + c_y$ encrypts $2^k x + y$, and one bootstrapping with the table $2^k x + y \mapsto xy$ returns the product.

6. A Worked Instance

Example 1 (One programmable bootstrapping). *Parameters.* $q = 1024$, $p = 8$, $\Delta = 128$; $s = (1, 0, 1)$, so $n = 3$; $N = 16$, so $2N = 32$ and $w = 4$. The function is $f(m) = [m \geq 2]$, so $v = 128(0, \dots, 0, 1, \dots, 1)$ with eight zeros then eight ones.

Input. $m = 2$, $a = (333, 790, 150)$, $e = 5$: $b = 333 + 150 + 256 + 5 = 744$, 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 $32 \cdot 261/1024 = 8.16$. Adding $w/2 = 2$ gives 10, in box $\lfloor 10/4 \rfloor = 2 = m$.

Blind rotation. The accumulator starts at $X^{-25}v$. Tracking the exponent: $-25 \rightarrow -15$ ($s_1 = 1$, add 10) $\rightarrow -15$ ($s_2 = 0$) $\rightarrow -10$ ($s_3 = 1$, add 5). The constant coefficient of $X^{-10}v$ is $v_{10} = 128 = \Delta f(2)$.

Output. After extraction and key switching the ciphertext decrypts to $1 = f(2)$. Repeating with input noise $e = \pm 9$ gives the same result for every m : the output depends on the box, not on the input noise.

7. Cost and Comparison

The original parameters are $n = 500$, $N = 1024$, $q = 2^{32}$: the bootstrapping key is about 16 MB and one bootstrapped gate takes about 13 ms on one core, against 1 GB and 690 ms for its predecessor FHEW. Blind rotation is n external products, done with the FFT.

	BFV / BGV / CKKS	GSW	TFHE
Plaintext	vectors in slots	a bit	a bit or a few-bit integer
Ciphertext	pair of ring elements	matrix	LWE vector
Multiplication	tensor, relinearize	$G^{-1}(C_1)C_2$	table lookup by bootstrapping
Noise	levels, spent until bootstrapping	asymmetric growth	reset after every gate
Bootstrapping	rare, expensive	expensive	every gate, fast
Suited to	arithmetic on many values	theory	comparisons, branching

The trade is throughput for flexibility: BFV, BGV and CKKS process thousands of slots per operation; TFHE processes one small message per bootstrapping but evaluates any function with no depth to plan. Security rests on LWE, ring-LWE and circular security, since the bootstrapping key encrypts s under S .

Summary

1. TFHE stores messages at Δm in $\mathbb{Z}_q$ with a padding bit on top. LWE ciphertexts carry data, GLWE the accumulator, GGSW the key.
2. CMux, built on the external product, is an encrypted if-then-else with additive noise.
3. Bootstrapping is modulus switching to $2N$, blind rotation of a test polynomial through n CMux steps, extraction, and key switching. The output noise is fixed.
4. The test polynomial is a lookup table: one bootstrapping refreshes the noise and applies any function. Gates are a sum plus a table.
5. TFHE bootstraps after every gate: no depth limit, one small message per bootstrapping.

Exercises

1. Repeat Example 1 for $m = 3$ with the same a and $e = 5$: compute $b, \tilde{b}, \tilde{\varphi}$, the box and the output.
2. With v from Example 1, give the constant coefficient of $X^{-k}v$ for $k = 3, 10, 26$.
3. Give the tables for NOR and XNOR on $m_1 + m_2$.
4. In Example 1, encrypt $m = 6$, which uses the padding half. What does bootstrapping return?
5. How many bits of message space does multiplying two 2-bit numbers by a table need? Which p allows it?