

Lecture 14

Bootstrapping

SEC 595: Encrypted Computing

Roadmap

BFV, BGV and CKKS (Lectures 10–12) are levelled: the depth they support is fixed at key generation. This lecture removes that limit with one idea: *evaluate the scheme's own decryption under encryption*. Section 1 states the limit. Section 2 gives the construction, Section 3 the condition under which it works, and Section 4 the extra assumption it needs. Section 5 carries the construction out for GSW (Lecture 13).

1. The Ceiling

Every lattice ciphertext carries noise. Addition adds noises, multiplication multiplies them, and decryption fails once the noise passes a threshold. So the multiplicative depth is fixed at KeyGen, and whether a ciphertext can still be used depends on its *history*, not on its message. A larger modulus raises the ceiling; it never removes it.

Decryption is a fixed, public, finite function of two inputs, $\text{Dec}_{\mathcal{E}}(\text{Sk}, \text{ct}) = m$. It is therefore a valid choice of the function Π in $\text{Evaluate}(\text{Pk}, \Pi, \text{ct}_1, \dots, \text{ct}_k)$ (Lecture 8). The only obstacle is that Evaluate needs its inputs encrypted, and one input is the secret key. So encrypt the secret key.

2. Recryption

Definition 1 (Bootstrapping, Gentry 2009). Publish the **bootstrapping key** $\overline{\text{Sk}} = \text{Enc}(\text{Pk}, \text{Sk})$ once. To refresh ct , encrypt it, $\overline{\text{ct}} = \text{Enc}(\text{Pk}, \text{ct})$, and compute

$$\text{ct}^{\#} = \text{Evaluate}(\text{Pk}, \text{Dec}_{\mathcal{E}}, \overline{\text{ct}}, \overline{\text{Sk}}).$$

No new algorithm is introduced: bootstrapping is Evaluate with $\Pi = \text{Dec}_{\mathcal{E}}$. A ciphertext is a string, so it can be encrypted, in practice piece by piece. $\overline{\text{ct}}$ has two layers: an outer one we compute under and an inner one that the evaluated decryption removes.

Proposition 1 (Correctness and refreshing). *If $\mathcal{E}$ evaluates $\text{Dec}_{\mathcal{E}}$ correctly and ct decrypts to m , then $\text{ct}^{\#}$ decrypts to m , and its noise depends only on the circuit $\text{Dec}_{\mathcal{E}}$, not on the history of ct .*

Proof sketch. $\text{ct}^{\#}$ encrypts $\text{Dec}_{\mathcal{E}}(\text{Sk}, \text{ct}) = m$ by correctness of Evaluate. It is the output of one run of Evaluate on a fixed circuit with fresh inputs; the noise of ct sits inside the *plaintext* of $\overline{\text{ct}}$ and is removed by the decryption being evaluated. $\square$

At no point is m in the clear, and the evaluator never holds Sk . The output is an ordinary ciphertext at a fixed, known noise level.

3. Bootstrappable Schemes

Measure the budget in levels: a fresh ciphertext supports D levels of multiplication, and evaluating $\text{Dec}_{\mathcal{E}}$ consumes d of them.

Definition 2 (Bootstrappable). $\mathcal{E}$ is **bootstrappable** if it can evaluate its own decryption circuit plus at least one more operation: $d < D$.

Then every bootstrapped ciphertext has $D - d \geq 1$ levels left, whatever came before, and alternating evaluate $\rightarrow$ bootstrap $\rightarrow$ evaluate $\rightarrow \dots$ reaches any depth: **levelled** + **bootstrapping** = **FHE**.

The budget is spent by multiplicative *depth*, not by gate count, so the decryption circuit must be shallow, however wide. Lattice decryption, an inner product and a rounding test, is many additions and little depth. The difficulty is that D and d grow together: larger parameters raise D but also enlarge the ciphertext, and so its decryption circuit.

Example 1 (Where the budget goes). Let $D = 10$ and $d = 7$. Each bootstrap returns $D - d = 3$ useful levels. A circuit of depth $L > D$ needs the least k with $D + k(D - d) \geq L$, that is $k = \lceil (L - D)/(D - d) \rceil$. For $L = 100$: $k = \lceil 90/3 \rceil = 30$, and indeed $10 + 30 \cdot 3 = 100$ while $10 + 29 \cdot 3 = 97 < 100$. If $d = D$, a refreshed ciphertext has no level left and no progress is possible.

Bootstrapping removes *noise* ceilings only. BGN (Lecture 9) allows one multiplication because of its algebra, not its noise: no choice of parameters lets it evaluate its own decryption.

4. Circular Security

$\overline{\text{Sk}} = \text{Enc}(\text{Pk}, \text{Sk})$ encrypts a key under itself. In the IND-CPA game (Lecture 8) the messages are chosen without knowledge of the key, so the game says nothing about releasing $\overline{\text{Sk}}$.

Definition 3 (Circular security). $\mathcal{E}$ is **circular secure** if it remains IND-CPA secure when the attacker is also given $\text{Enc}(\text{Pk}, \text{Sk})$.

It is assumed, not proved: no reduction from LWE is known, and no attack. The alternative is a chain of keys, publishing $\text{Enc}(\text{Pk}_{i+1}, \text{Sk}_i)$ for $i = 0, \dots, L - 1$; no key is encrypted under itself, but only L bootstraps are possible, so the scheme is levelled again.

5. Bootstrapping GSW

5.1 Decryption Is One Inner Product

A GSW ciphertext (Lecture 13) satisfies $Cs = M \cdot Gs + e$ with $\|e\|_{\infty} < q/4$. The last row of G is $(0, \dots, 0, q/2)$ and the last key entry is 1, so

$$(Cs)_{\ell} = \sum_{j=1}^{n+1} C_{\ell j} s_j = M \cdot \frac{q}{2} + e_{\ell}.$$

M is read by asking whether this value is nearer 0 or $q/2$, as in Regev decryption (Lecture 7). So the decryption circuit C_{Dec} is **one inner product and a top-bit test**; the other rows of C are not needed.

5.2 As a Boolean Circuit

Take the secret binary, $\hat{s} \in \{0, 1\}^n$ (Lecture 13, Section 6.2). The entries $C_{\ell j}$ are known to the evaluator, so they are *constants* of the circuit:

- *selection*: $C_{\ell j} \hat{s}_j$ keeps or zeroes a known number according to a key bit, which is free;
- *adder tree*: the sum of $n + 1$ numbers of $\log q$ bits, built from full adders;
- *test*: read the top bit of the sum (after adding $q/4$).

GSW supplies the gates (Lecture 13): $G^{-1}(C_1)C_2$ is AND, $G - G^{-1}(C_1)C_2$ is NAND, and NAND is universal. The circuit has about $(n + 1) \log q$ full adders and AND-depth $O(\log n + \log \log q)$: wide and shallow.

5.3 The Bootstrapping Key

The key is one GSW ciphertext per key bit, under that same key:

$$ek = (\text{Enc}(s, \hat{s}_1), \dots, \text{Enc}(s, \hat{s}_n)).$$

Refreshing C evaluates C_{Dec} with C hard-wired and ek as the encrypted inputs. The result $\tilde{C}$ decrypts to the same bit as C with new noise. Publishing ek needs circular security (Section 4).

5.4 Noise: the Accumulator

GSW multiplication is asymmetric (Lecture 13): $\|e_{\text{new}}\|_{\infty} \leq \|e_1\|_{\infty} + \ell \|e_2\|_{\infty}$ for $G^{-1}(C_1)C_2$ with bit messages; only the second operand's noise is multiplied by ℓ . So evaluate C_{Dec} *sequentially*, with the running value (the **accumulator**) always in the decomposed slot and a fresh key ciphertext in the other.

Proposition 2 (Bootstrapping condition). *If the key ciphertexts have noise at most E and C_{Dec} is evaluated as g sequential gates, the output noise is at most $E(1 + g\ell)$, and bootstrapping is correct when*

$$E(1 + g\ell) < \frac{q}{4}.$$

Each gate adds at most ℓE and nothing is multiplied. The noise of the input C does not appear. A balanced tree of the same gates would instead give $E(\ell + 1)^d$, exponential in the depth.

Example 2 (The budget: $n = 500$, $E = 2^{10}$). At $q = 2^{60}$: $\ell = (n + 1) \log q = 501 \cdot 60 = 30\,060$ and $q/4 = 2^{58}$. The condition allows g up to about 9.4×10^9 gates, against roughly 30 000 in C_{Dec} : GSW is bootstrappable here. At $q = 2^{30}$: $\ell = 15\,030$ and only $g \leq 17$ gates fit, so bootstrapping fails.

GSW bootstrapping works but is expensive: one ciphertext at $n = 500$, $q = 2^{60}$ is a $30\,060 \times 501$ matrix of about 113 MB. TFHE (Lecture 15) keeps the algebra and the accumulator, and replaces the matrices by ring elements.

Summary

1. Levelled schemes have a fixed depth: noise grows with the history of a ciphertext.
2. Bootstrapping is $\text{Evaluate}(\text{Pk}, \text{Dec}_\mathcal{E}, \text{Enc}(\text{Pk}, \text{ct}), \text{Enc}(\text{Pk}, \text{Sk}))$: the output encrypts the same m with noise that depends only on $\text{Dec}_\mathcal{E}$.
3. A scheme is bootstrappable if $d < D$; then levelled becomes fully homomorphic. Depth, not gate count, is the cost.
4. Publishing $\text{Enc}(\text{Pk}, \text{Sk})$ needs circular security.
5. GSW decryption is one inner product and a top-bit test. Evaluated sequentially with the accumulator in the decomposed slot, the noise stays below $E(1 + g\ell)$, independent of the input.

Exercises

1. Write bootstrapping in the interface of Lecture 8: name Π , the input ciphertexts and the output.
2. With $D = 10$ and $d = 7$, how many bootstraps does a circuit of depth 40 need? Why does the answer not depend on the width of the circuit?
3. Why does bootstrapping need compactness (Lecture 8)? Consider the non-compact Evaluate that returns $(\Pi, \text{ct}_1, \dots, \text{ct}_k)$.
4. For $n = 3$, $q = 8$, $\hat{s} = (1, 0, 1)$ and last row $C_\ell = (3, 6, 7, 7)$, compute $(Cs)_\ell$ with $s = (-\hat{s}, 1)$ and read the message bit.
5. In Example 2, find the smallest $\log_2 q$ for which $E(1 + g\ell) < q/4$ holds with $g = 30\,000$.
6. Explain why the accumulator must be in the decomposed slot of $G^{-1}(C_1)C_2$ rather than the other one.