Secure computing, third lecture

Fully homomorphic encryption: computing on encrypted data (part 2)

Xavier Leroy

2025-11-20

Collège de France, chair of Software sciences
xavier.leroy@college-de-france.fr

Reminders from the previous lecture

Alice's jewelry shop

(C. Gentry, CACM 53(3), 2010)

(source)

Alice the jeweler puts precious materials in a glove box, which she locks with her private key.

Bob the worker can assemble the jewel but not take it with him.

When the jewel is done, Alice unlocks the box and takes the jewel.

Alice's jewelry shop

(C. Gentry, CACM 53(3), 2010)

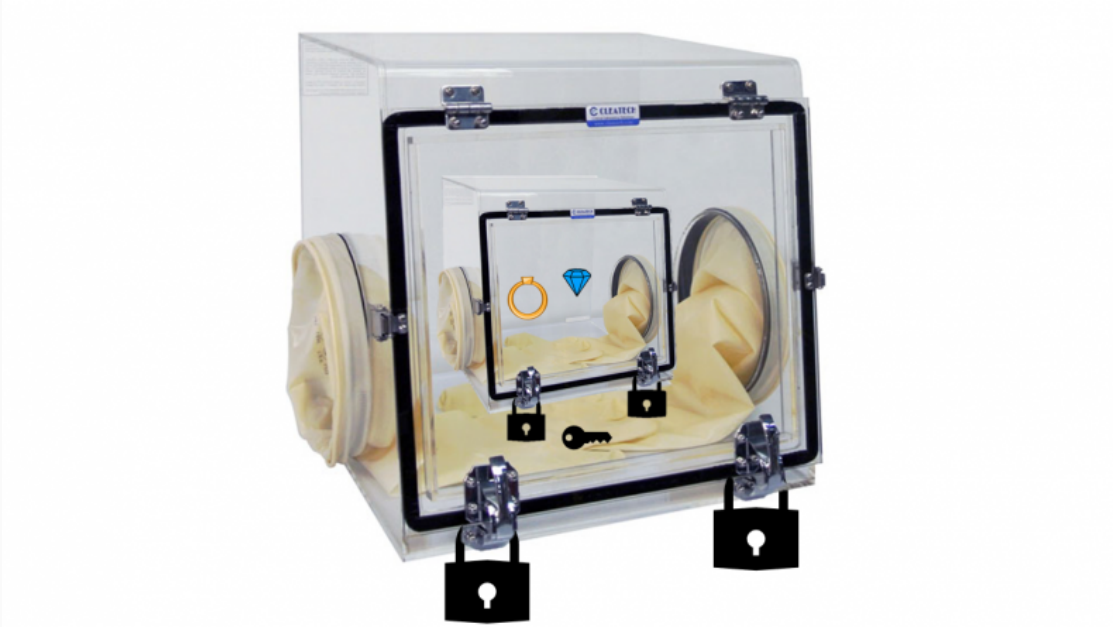
(source)

Problem: the gloves harden rapidly when used.

They become solid and unusable before the jewel is finished.

Alice's jewelry shop

(C. Gentry, CACM 53(3), 2010)



(source)

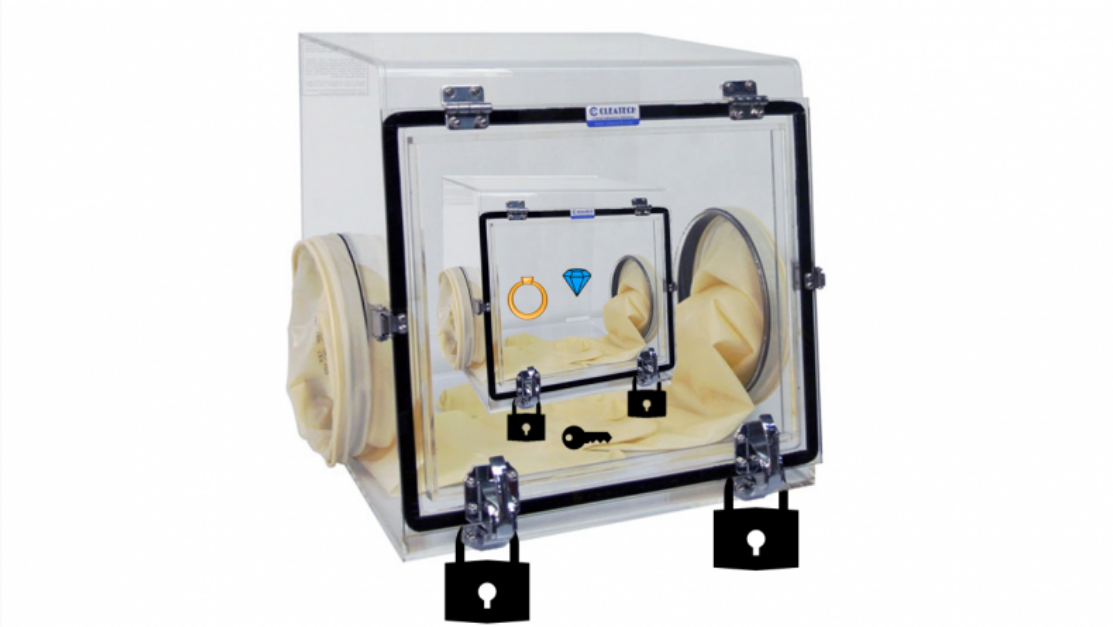
Solution: Alice puts the glove box in a bigger box, along with the key for the first box, and locks the bigger box.

Bob unlocks the first box, fetches the jewel and the materials, and continues working.

This "bootstrap" procedure is repeated until the jewel is finished.

Alice's jewelry shop

(C. Gentry, CACM 53(3), 2010)



(source)

Problem: Alice's locks are complicated and rusty. Bob is unable to open them before his gloves become solid.

Solution: Alice puts a can of oil in the box so that Bob can grease the locks and open them more easily.

Caution: the oil could weaken the security of the boxes!

Reminder: constructing a cipher based on noise

The encryption of a bit b with the private key p is

$$\mathcal{E}_p(b) = pq + 2r + b \qquad \begin{array}{l} q \text{ random nonnegative integer} \gg p \\ r \text{ random integer, } |r| < p/4 \end{array}$$

General form: $\mathit{ciphertext} = \mathit{noise}_1 \cdot \mathit{key} + \mathit{noise}_2$

The cleartext b is inserted in noise_2 and masked by noise_1.

Decryption is possible as long as $|r| < p/4$, by cancelling noise_1 :

$$\mathcal{D}_p(c) = (c - p\lfloor c/p \rceil) \bmod 2$$

A somewhat homomorphic cipher

$$\mathcal{E}_p(b_1) + \mathcal{E}_p(b_2) = \text{an encryption of } b_1 \oplus b_2$$
$$\mathcal{E}_p(b_1) \cdot \mathcal{E}_p(b_2) = \text{an encryption of } b_1 \cdot b_2$$

as long as the noise r of the result remains $< p/4$.

The noise $N(c)$ (in bits) increases

- slowly during addition $\max(N(c_1), N(c_2)) + 1$
- rapidly during multiplication $N(c_1) + N(c_2) + 1$

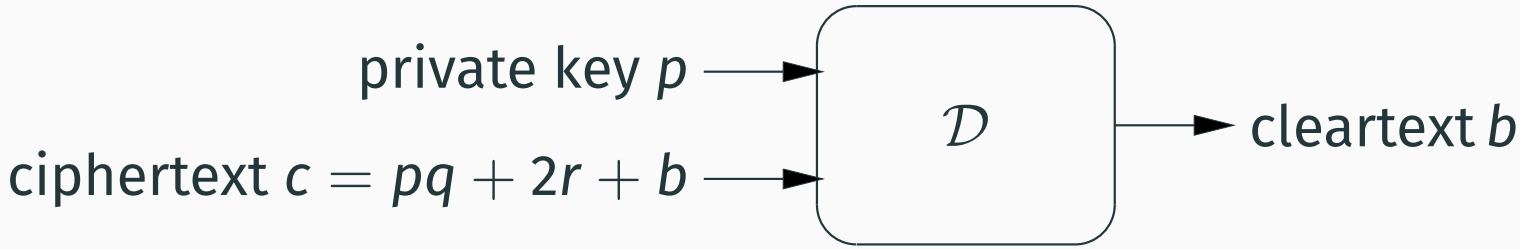
Waht can be evaluated homomorphically with this cipher?

- Multivariate polynomials of low degree ($\leq \lambda$).
  The number of monomials can be high.

- Boolean circuits of low multiplicative depth (AND, OR)
  (between $\log_2 \lambda$ and λ depending on circuit shape).
  The additive depth (XOR, NOT) can be high.

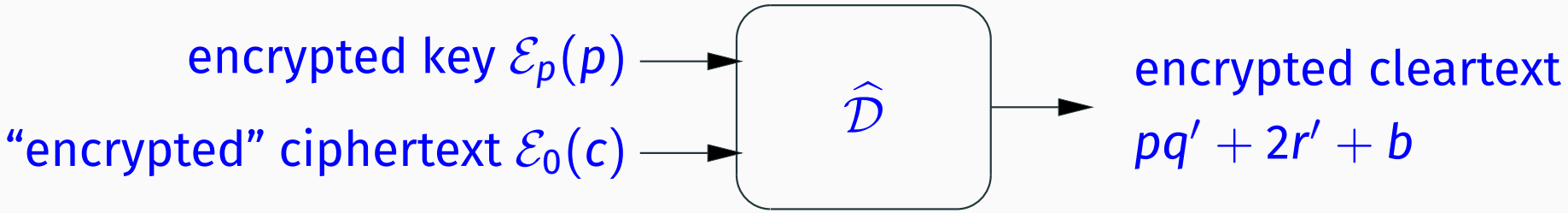
Reminder: the bootstrapping procedure (Gentry, 2009)

De-noise a ciphertext by homomorphic evaluation of the decryption circuit $\mathcal{D}$.



Reminder: the bootstrapping procedure (Gentry, 2009)

De-noise a ciphertext by homomorphic evaluation of the decryption circuit $\mathcal{D}$.



The result is a ciphertext equivalent to c, whose noise depends only on the multiplicative depth of the circuit $\mathcal{D}$.

Towards fully homomorphic encryption

Somewhat homomorphic encryption + bootstrap

$\Rightarrow$ fully homomorphic encryption
(able to evaluate any Boolean circuit)

Provided the decryption circuit $\mathcal{D}$ is of low enough multiplicative depth...

Gentry's approach: "grease encryption"
(add redundant info to ciphertexts)
so as to simplify the decryption circuit.

$$\mathcal{D}_p(c, z_1 \ldots z_N) = LSB(\lfloor \sum_i s_i \cdot z_i \rceil) \oplus LSB(c)$$

In this lecture: how to improve this approach

1. Better somewhat homomorphic ciphers

Based on the LWE (Learning With Errors) problem.

2. Multiplication that increases noise less rapidly

The BGV approach, using “leveled” somewhat homomorphic encryption.

3. Less costly bootstrapping procedures

The TFHE approach and its “programmable bootstrap” (de-noise while evaluating tabulated functions).

The LWE problem (Learning With Errors)

The LWE problem (Oded Regev, 2005)

Solve a system of "approximate" linear equations over integers modulo q.

Example:

$$
\begin{aligned}
14s_1 + 15s_2 + 5s_3 + 2s_4 &\approx 8 \pmod{17} \\
13s_1 + 14s_2 + 14s_3 + 6s_4 &\approx 16 \pmod{17} \\
6s_1 + 10s_2 + 13s_3 + 1s_4 &\approx 3 \pmod{17} \\
10s_1 + 4s_2 + 12s_3 + 16s_4 &\approx 12 \pmod{17} \\
9s_1 + 5s_2 + 9s_3 + 6s_4 &\approx 9 \pmod{17} \\
3s_1 + 6s_2 + 4s_3 + 5s_4 &\approx 16 \pmod{17} \\
6s_1 + 7s_2 + 16s_3 + 2s_4 &\approx 3 \pmod{17}
\end{aligned}
$$

where $\approx$ means "equal to within 1". (Solution: $\mathbf{s} = (0, 13, 9, 11)$.)

The LWE problem

$$14s_1 + 15s_2 + 5s_3 + 2s_4 = 8 \pmod{17} \quad (1)$$

$$13s_1 + 14s_2 + 14s_3 + 6s_4 = 16 \pmod{17} \quad (2)$$

$$6s_1 + 10s_2 + 13s_3 + 1s_4 = 3 \pmod{17} \quad (3)$$

$$10s_1 + 4s_2 + 12s_3 + 16s_4 = 12 \pmod{17} \quad (4)$$

If the error is zero, the problem is easily solved by Gaussian elimination, as soon as we have 4 independent equations.

The LWE problem

$$14s_1 + 15s_2 + 5s_3 + 2s_4 = 8 \pm 1 \pmod{17} \quad (1)$$

$$13s_1 + 14s_2 + 14s_3 + 6s_4 = 16 \pm 1 \pmod{17} \quad (2)$$

$$6s_1 + 10s_2 + 13s_3 + 1s_4 = 3 \pm 1 \pmod{17} \quad (3)$$

$$10s_1 + 4s_2 + 12s_3 + 16s_4 = 12 \pm 1 \pmod{17} \quad (4)$$

If the error is not zero, it explodes during Gaussian elimination. For instance, $3 \times (4) - 5 \times (3)$ eliminates s_1 and results in

$$13s_2 + 5s_3 + 9s_4 = 4 \pm 8 \pmod{17}$$

All values of the left-hand side are possible (modulo 17).

The LPN problem (Learning Parity with Noise)

LWE generalizes an older problem: LPN.

A parity function is a XOR of a subset of its inputs:

$$P_{3,4,7}(x_1, \ldots, x_8) = x_3 \oplus x_4 \oplus x_7$$

It's a dot product with a bit vector $\mathbf{s}$, which characterizes the function uniquely:

$$P_{\mathbf{s}}(x_1, \ldots, x_n) = \sum_{i=1}^{n} s_i \cdot x_i \pmod 2$$

The LPN problem (Learning Parity with Noise)

The LP problem (Learning Parity): given a number of samples $(\mathbf{x}, P_{\mathbf{s}}(\mathbf{x}))$ for randomly-chosen $\mathbf{x}$, find $\mathbf{s}$.

Easy by Gaussian elimination if we have n independent samples.
Difficult for learning algorithms based on gradient descent.

The LPN problem (Learning Parity with Noise)

The LP problem (Learning Parity): given a number of samples $(\mathbf{x}, P_{\mathbf{s}}(\mathbf{x}))$ for randomly-chosen $\mathbf{x}$, find $\mathbf{s}$.

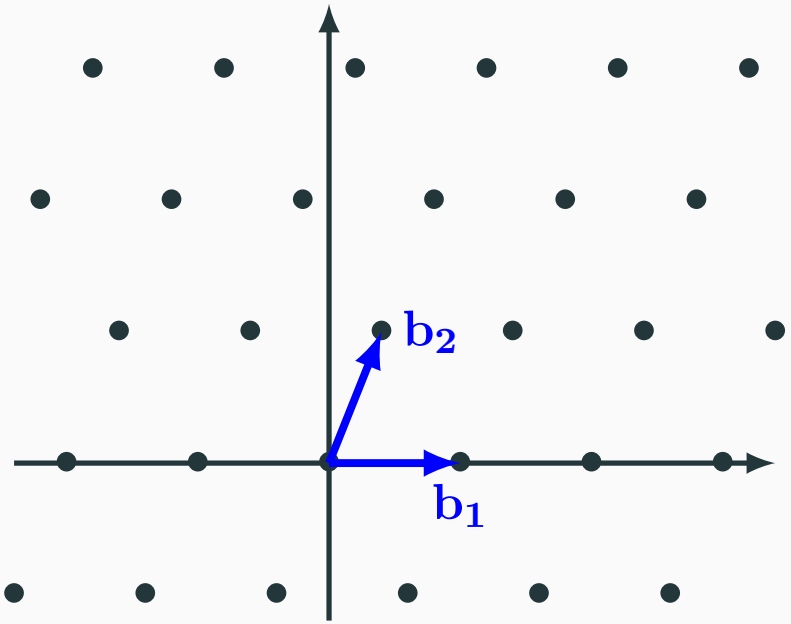
The LPN problem (Learning Parity with Noise): find $\mathbf{s}$ given a number of "noisy" samples

$$(\mathbf{x}, P_{\mathbf{s}}(\mathbf{x}) + e) \qquad \mathbf{x} \text{ random}$$
$$e = 1 \text{ with probability } \varepsilon, \quad e = 0 \text{ otherwise}$$

Problem known to be difficult, even in the average case, even for a quantum computer.

The best known algorithms are in sub-exponential time and space.

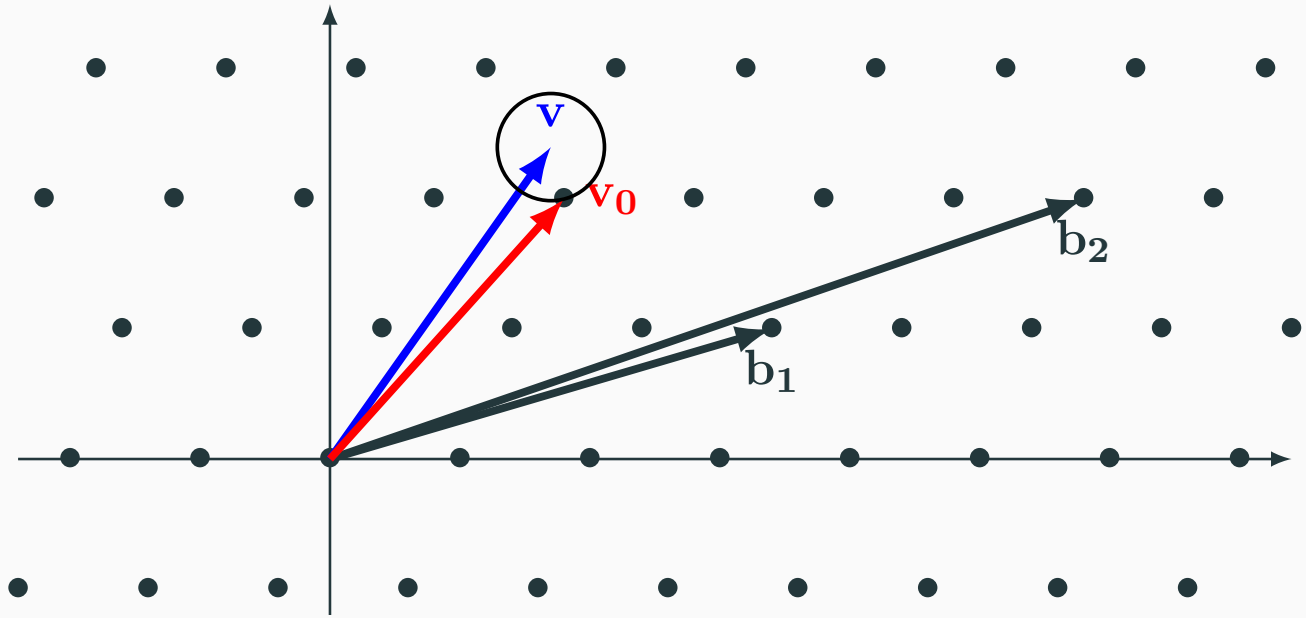
Euclidean lattices



A lattice = the set of vectors with integer coordinates in a given base $B = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$.

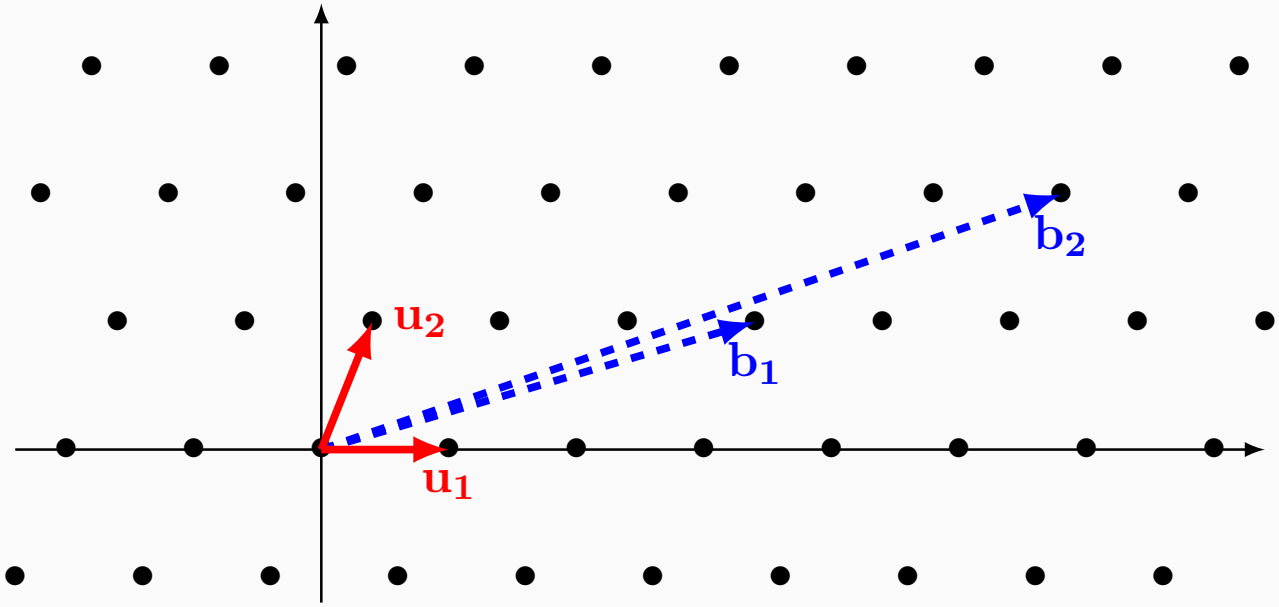
$$\left\{ \sum_{i=1}^{n} p_i \, \mathbf{b}_i \;\middle|\; p_i \in \mathbb{Z} \right\}$$

The CVP problem (Closest Vector Problem)



Given a vector **v**, find the coordinates of a vector $\mathbf{v}_0$ of the lattice that is closest to **v**.

The SIVP problem (Shortest Independent Vectors Problem)



Find n independent lattice vectors of minimal length.

Other hard problems on Euclidean lattices

SVP (Shortest Vector Problem): find the shortest non-null vector of the lattice. (Or just determine its length.)

GapSVP: is the length of the shortest non-null vector ≤ 1 or $\geq \beta$ (for a fixed $\beta > 1$) ?

All these problems are NP-hard, even when approximated.

Several reductions "worst-case hard" $\Longrightarrow$ "average-case hard" (Ajtai, 1995).

LWE and CVP

LWE can be viewed as a generalization of the CVP problem in $\mathbb{Z}_q$ (integers modulo q).

Solving

$$a_{11}x_1 + \cdots a_{1n}x_n \approx b_1$$
$$\vdots$$
$$a_{n1}x_1 + \cdots a_{nn}x_n \approx b_n$$

amounts to finding the coordinates $\mathbf{x}$ of an element of the lattice generated by the base $\mathbf{a}_1, \ldots, \mathbf{a}_n$ that is close enough to the point $\mathbf{b}$.

The LWE problem

Let $\mathbf{s} \in \mathbb{Z}_q^n$ be the secret. Consider the numbers

$\langle \mathbf{a}, \mathbf{s} \rangle + e \pmod{q}$ $\quad$ $\mathbf{a} \in \mathbb{Z}_q^n$ uniformly random vector

$e \in \mathbb{Z}$ random integer with distribution χ

LWE problem: given a set of pairs $(\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e)$, find $\mathbf{s}$.

Decisional LWE problem: distinguish between
a set of pairs $(\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e)$ and
a set of pairs $(\mathbf{a}, u)$ where $u \in \mathbb{Z}_q$ is uniformly random.

Some hardness results

(O. Regev, *On lattices, learning with errors, random linear codes, and cryptography*, J. ACM, 2006).

- If q is exponential in n, LWE is at least as hard as GapSVP approximated up to a polynomial factor.
- If q is polynomial in n, LWE is at least as hard as GapSVP or SIVP approximated up to a polynomial factor on a quantum computer.
- Decisional LWE is as hard as LWE.
- Decisional LWE is as hard in the average case as in the worst case.

Encryption based on LWE

Encrypting with LWE

Basic idea: hide the cleartext in the error term.

Symmetric encryption: (using the secret key $\mathbf{s}$)

$$\mathcal{E}_{\mathbf{s}}(m) = (\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + f(m))$$

where $\mathbf{a} \in \mathbb{Z}_q^n$ is a random vector
and $f : Msg \to \mathbb{Z}_q$ a randomized, invertible, additive function.

Decryption:

$$\mathcal{D}_{\mathbf{s}}((\mathbf{a}, b)) = f^{-1}(b - \langle \mathbf{a}, \mathbf{s} \rangle)$$

IND-CPA security follows from the decisional LWE problem:
an attacker who does not know $\mathbf{s}$ cannot distinguish $\mathcal{E}_{\mathbf{s}}(m)$ from $(\mathbf{a}, b)$
with $\mathbf{a}$ and b uniformly random.

Encoding the message in the error term

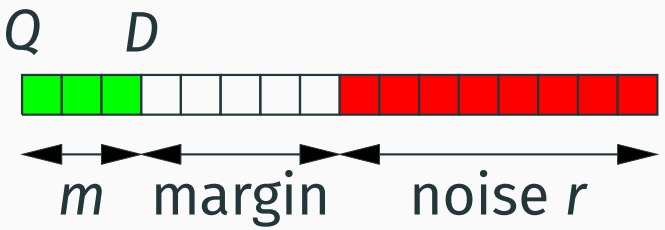
Big-endian encoding: m in the most significant bits.

$$f(m) = \Delta m + r \quad \text{with } r \text{ random, } |r| < \Delta/2$$

$$f^{-1}(c) = \lfloor c/\Delta \rceil$$

The messages m are integers in $\mathbb{Z}_t$, where $t = q/\Delta$.

Example if $q = 2^Q$ and $t = 2^T$:
the messages m are T-bit words, stored in the T most significant bits; the noise r is a signed integer of at most $D = Q - T$ bits.



Encoding the message in the error term

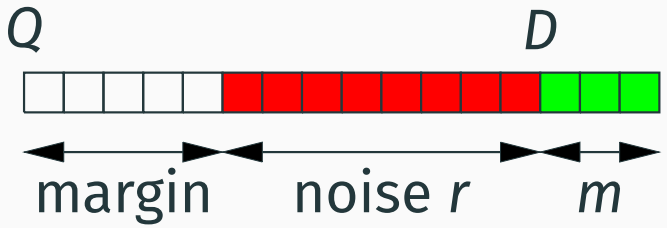
Little-endian encoding: m in the least significant bits.

$$f(m) = \Delta r + m \quad \text{with } r \text{ random, } |r| < q/2\Delta$$

$$f^{-1}(c) = c \bmod \Delta$$

The messages m are integers in $\mathbb{Z}_\Delta$.

Example if $q = 2^Q$ and $\Delta = 2^D$:
the messages are D-bit words, stored in the D least significant bits; the noise r is an integer of at most $Q - D$ bits.



Trivial encryption

If the message m reveals no confidential information
(e.g. it is a constant, or it is already encrypted),
we have a trivial encryption:

$$\mathcal{E}_0(m) = (\mathbf{0}, m) \qquad \text{(little-endian)}$$
$$\mathcal{E}_0(m) = (\mathbf{0}, \Delta m) \qquad \text{(big-endian)}$$

These ciphers decrypt to m for any secret key $\mathbf{s}$.

Public-key encryption

A public key = k encryptions of 0.

$$pk = \big((\mathbf{a}_1, \langle \mathbf{a}_1, \mathbf{s} \rangle + f(0)), \ldots (\mathbf{a}_k, \langle \mathbf{a}_k, \mathbf{s} \rangle + f(0))\big)$$

Encrypting m with the public key $pk = (\mathbf{a}_1, b_1), \ldots, (\mathbf{a}_k, b_k)$:
draw a subset $P \subseteq \{1, \ldots, k\}$ and take

$$\mathcal{E}_{pk}(m) = \big(\sum_{\mathbf{i} \in P} \mathbf{a}_i, \sum_{\mathbf{i} \in P} b_i + m\big) \qquad \text{(little-endian)}$$

$$\mathcal{E}_{pk}(m) = \big(\sum_{\mathbf{i} \in P} \mathbf{a}_i, \sum_{\mathbf{i} \in P} b_i + \Delta m\big) \qquad \text{(big-endian)}$$

Encrypting a vector of n messages

We can encrypt a vector $\mathbf{m}$ of n messages like we would encrypt n messages $m_1, \ldots, m_n$:

$$\mathcal{E}_{\mathbf{s}}(\mathbf{m}) = (\mathbf{A}, \mathbf{A} \cdot \mathbf{s} + \mathbf{f}(\mathbf{m}))$$

where $\mathbf{A}$ is a $n \times n$ random matrix
and $\mathbf{A} \cdot \mathbf{s}$ the application of $\mathbf{A}$ to vector $\mathbf{s}$.

Problem: the ciphertext has quadratic size ($n^2 + n$ integers).

(Example: $n = 1024$ and $q = 2^{64} \Rightarrow$ 8396800 bytes.)

Encrypting with RLWE (Ring Learning With Error)

Idea: instead of random matrices **A**, let's use matrices with the following shape:

$$\mathbf{A} = \begin{pmatrix} a_0 & a_1 & a_2 & \cdots & a_{n-1} \\ -a_{n-1} & a_0 & a_1 & \cdots & a_{n-2} \\ -a_{n-2} & -a_{n-1} & a_0 & \cdots & a_{n-3} \\ \vdots & \vdots & \vdots & & \vdots \\ -a_1 & -a_2 & -a_3 & \cdots & a_0 \end{pmatrix}$$

with $a_0, \ldots, a_{n-1}$ drawn randomly in $\mathbb{Z}_q$.

This matrix is not pulled out of thin air: it corresponds to the multiplication of polynomials modulo $X^n + 1$ (i.e. with $X^n = -1$).

Encrypting with RLWE (Ring Learning With Error)

The secret S, the randomness A, the noise R, the cleartext M and the ciphertext C are all viewed as polynomials in $Z_q[X]/(X^n + 1)$.

$$\mathcal{E}_S(M) = (A, AS + \Delta R + M) \qquad \text{little-endian}$$
$$\mathcal{E}_S(M) = (A, AS + \Delta M + R) \qquad \text{big-endian}$$

A coefficients: random, uniform in $\mathbb{Z}_q$.
R coefficients: random, not too big.
M coefficients: in $\mathbb{Z}_\Delta$ (little-endian) or $\mathbb{Z}_{q/\Delta}$ (big-endian).

The ciphertext has linear size ($2n$ integers).

Security proved by the difficulty of the RLWE problem (LWE problem for polynomials) (provided n is a power of 2).

GLWE (Generalized LWE)

Like RLWE, but the secret and the randomness are k-tuples of polynomials in $Z_q[X]/(X^n + 1)$.

$$\mathcal{E}_{(S_1,\ldots,S_k)}(M) = (A_1, \ldots, A_k, A_1S_1 + \cdots + A_kS_k + \Delta R + M)$$

A unified presentation of LWE (case $n = 1$) and RLWE (case $k = 1$).

Encodes n numbers using $kn + n$ numbers.

Homomorphic addition

Somewhat homomorphic encryption for addition

The pointwise sum of two LWE ciphertexts is an encryption of the sum modulo t, if the noise doesn't overflow.

Big-endian:

$$\begin{aligned}
&(\mathbf{a}_1, \langle \mathbf{a}_1, \mathbf{s} \rangle + \Delta m_1 + r_1) + (\mathbf{a}_2, \langle \mathbf{a}_2, \mathbf{s} \rangle + \Delta m_2 + r_2) \\
&\quad = \quad (\mathbf{a}_1 + \mathbf{a}_2, \langle \mathbf{a}_1 + \mathbf{a}_2, \mathbf{s} \rangle + \Delta(m_1 + m_2) + r_1 + r_2) \\
&\quad = \quad \text{an encryption of } (m_1 + m_2) \bmod t \\
&\qquad\quad\ \ \text{provided that } |r_1 + r_2| < \Delta/2
\end{aligned}$$

Somewhat homomorphic encryption for addition

Little-endian:

$$
\begin{aligned}
&(\mathbf{a}_1, \langle \mathbf{a}_1, \mathbf{s} \rangle + \Delta r_1 + m_1) + (\mathbf{a}_2, \langle \mathbf{a}_2, \mathbf{s} \rangle + \Delta r_2 + m_2) \\
&\quad = \quad (\mathbf{a}_1 + \mathbf{a}_2, \langle \mathbf{a}_1 + \mathbf{a}_2, \mathbf{s} \rangle + \Delta(r_1 + r_2) + m_1 + m_2) \\
&\quad = \quad \text{an encryption of } (m_1 + m_2) \bmod \Delta \\
&\qquad\quad\ \ \text{provided that } \Delta | r_1 + r_2 + (m_1 + m_2)/\Delta | < q/2
\end{aligned}
$$

Same properties for RLWE and GLWE.

Multiplication by a small constant

The product of a ciphertext and a constant k is an encryption of the product of the cleartext and k, provided k is small enough to avoid overflow.

Big-endian:

$$\begin{aligned} & k \cdot (\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + \Delta m + r) \\ = \quad & (k \cdot \mathbf{a}, \langle k \cdot \mathbf{a}, \mathbf{s} \rangle + \Delta(k \cdot m) + k \cdot r) \\ = \quad & \text{an encryption of } k \cdot m \bmod t \\ & \text{provided that } |k \cdot r| < \Delta/2 \end{aligned}$$

Unusable if k is large w.r.t. Δ.

Multiplication by an arbitrary constant

Assume $q = \beta^\ell$. We can write the constant k in base β:

$$k = k_0 + k_1\beta + k_2\beta^2 + \cdots + k_{\ell-1}\beta^{\ell-1}$$

Consider the leveled encryption of a message m: it's a collection of encryptions of m scaled by $1, \beta, \beta^2, \ldots$:

$$\mathcal{E}_{Lev}(m) = \left(\mathcal{E}(m), \mathcal{E}(\beta m), \mathcal{E}(\beta^2 m), \ldots, \mathcal{E}(\beta^{\ell-1} m)\right)$$

We can, then, compute the product km homomorphically:

$$\begin{aligned} & k_0 \cdot \mathcal{E}(m) + k_1 \cdot \mathcal{E}(\beta m) + \cdots + k_{\ell-1} \cdot \mathcal{E}(\beta^{\ell-1} m) \\ = \;& \text{an encryption of } k_0 m + k_1 \beta m + \cdots k_{\ell-1}\beta^{\ell-1} m \pmod q \\ = \;& \text{an encryption of } (k_0 + k_1\beta + \cdots + k_{\ell-1}\beta^{\ell-1})m = km \pmod q \end{aligned}$$

The numbers k_i are small, therefore the noise doesn't overflow.

Homomorphic multiplication and leveled circuits (the BGV approach)

Intermission: the tensor product

The tensor product $\mathbf{u} \otimes \mathbf{v}$ of a vector of dimension n by a vector of dimension m is the vector of dimension nm defined by

$$\mathbf{u} \otimes \mathbf{v} = (u_1 v_1, \ldots, u_1 v_m, \ldots, u_n v_1, \ldots, u_n v_m)$$

Tensor product commutes with dot product:

$$\langle \mathbf{u}_1, \mathbf{u}_2 \rangle \cdot \langle \mathbf{v}_1, \mathbf{v}_2 \rangle = \langle \mathbf{u}_1 \otimes \mathbf{v}_1, \mathbf{u}_2 \otimes \mathbf{v}_2 \rangle$$

Multiplication of two ciphertexts (BGV approach)

(Zvika Brakerski, Craig Gentry, Vinod Vaikuntanathan, *(Leveled) fully homomorphic encryption without bootstrapping*, ICTS 2012, TOCT 2014)

Consider two LWE encryptions of m_1 and m_2 (little-endian):

$$c_1 = (\mathbf{a}_1, b_1) \qquad \text{where } b_1 = \langle \mathbf{a}_1, \mathbf{s} \rangle + \Delta r_1 + m_1$$
$$c_2 = (\mathbf{a}_2, b_2) \qquad \text{where } b_2 = \langle \mathbf{a}_2, \mathbf{s} \rangle + \Delta r_2 + m_2$$

We have

$$\begin{aligned} b_1 \cdot b_2 &= (\langle \mathbf{a}_1, \mathbf{s} \rangle + \Delta r_1 + m_1) \cdot (\langle \mathbf{a}_2, \mathbf{s} \rangle + \Delta r_2 + m_2) \\ &= \langle \mathbf{a}_1, \mathbf{s} \rangle \cdot \langle \mathbf{a}_2, \mathbf{s} \rangle + \Delta(\cdots) + m_1 \cdot m_2 \\ &= \langle \mathbf{a}_1 \otimes \mathbf{a}_2, \mathbf{s} \otimes \mathbf{s} \rangle + \Delta(\cdots) + m_1 \cdot m_2 \end{aligned}$$

Hence, $(\mathbf{a}_1 \otimes \mathbf{a}_2, b_1 \cdot b_2)$ looks like an encryption of $m_1 \cdot m_2$ with the key $\mathbf{s} \otimes \mathbf{s}$ instead of $\mathbf{s}$, and the dimension n^2 instead of n.

Multiplication of two ciphertexts (BGV approach)

Ciphertexts $(\mathbf{a}_1, b_1)$, $(\mathbf{a}_2, b_2)$

(key $\mathbf{s}$, dimension n, modulus q, noise B)

tensor product

Ciphertext $(\mathbf{a}_1 \otimes \mathbf{a}_2, b_1 \cdot b_2)$

(key $\mathbf{s} \otimes \mathbf{s}$, dimension n^2, modulus q, noise B^2)

key switching

Ciphertext $(\mathbf{a}', b')$

(key $\mathbf{s}$ or $\mathbf{s}'$, dimension n, modulus q, noise B^2)

modulus switching

Chiffré $(\mathbf{a}'', b'')$

(key $\mathbf{s}$ or $\mathbf{s}'$, dimension n, modulus q/B, noise B)

Key switching

We have an encryption $c = (\mathbf{a}, b)$ of the message m with the key $\mathbf{s}$.

We want an encryption c' of m with another key $\mathbf{s}'$.

Idea: perform a homomorphic partial decryption of c.

We give $\mathbf{s}$ encrypted under $\mathbf{s}'$ to the computer:

$$\mathcal{E}_{\mathbf{s}'}(s_1), \ldots, \mathcal{E}_{\mathbf{s}'}(s_n) \qquad \text{(using the leveled form)}$$

We can, then, compute $b - \langle \mathbf{a}, \mathbf{s} \rangle$ homomorphically:

$$\begin{aligned} & \mathcal{E}_0(b) - a_1 \cdot \mathcal{E}_{\mathbf{s}'}(s_1) - \cdots - a_n \cdot \mathcal{E}_{\mathbf{s}'}(s_n) \\ = \quad & \text{an encryption of } b - \langle \mathbf{a}, \mathbf{s} \rangle \text{ under key } \mathbf{s}' \end{aligned}$$

(Must use the leveled form, since the a_i coefficients are large.)

Key switching

In little-endian representation, $b - \langle \mathbf{a}, \mathbf{s} \rangle = \Delta r + m$.

The result is an encryption $(\mathbf{a}', b')$ of $\Delta r + m$ under key $\mathbf{s}'$.
Moreover,

$$\begin{aligned}(\mathbf{a}', b') &= (\mathbf{a}', \langle \mathbf{a}', \mathbf{s}' \rangle + \Delta r' + (\Delta r + m)) \\ &= (\mathbf{a}', \langle \mathbf{a}', \mathbf{s}' \rangle + \Delta(r' + r) + m) \\ &= \text{an encryption of } m \text{ under } \mathbf{s}' \text{ if } |r' + r| \text{ is small enough}\end{aligned}$$

Therefore,

$$\mathcal{E}_0(b) - a_1 \cdot \mathcal{E}_{\mathbf{s}'}(s_1) - \cdots - a_n \cdot \mathcal{E}_{\mathbf{s}'}(s_n)$$

is an encryption of m under the new key $\mathbf{s}'$, if the noise doesn't overflow.

Noise analysis

The noise of a ciphertext is the magnitude of Δr.
It must remain $< q$ for decryption to succeed.

If both operands have noise B, tensor product followed by key switching produce a ciphertext with noise $\approx B^2$.

This greatly limits the multiplicative depth of circuits / the degree of polynomials that can be evaluated homomorphically.
Example with $q = B^{10}$:

	Noise / Modulus
initial ciphertext	B / B^{10}
depth 1, degree 2	B^2 / B^{10}
depth 2, degree 4	B^4 / B^{10}
depth 3, degree 8	B^8 / B^{10}
depth 4, degree 16	error! B^{16} / B^{10}

Modulus switching

Idea: reduce the modulus by a factor $\approx B$ at each multiplication.

This way, the absolute error remains $\pm$ constant.
The limiting factor becomes the size of the modulus.

Noise / Modulus	fixed modulus	decreasing modulus
initial ciphertext	B/B^{10}	B/B^{10}
depth 1, degree 2	B^2/B^{10}	$B^2/B^{10} = B/B^9$
depth 2, degree 4	B^4/B^{10}	$B^2/B^9 = B/B^8$
depth 3, degree 8	B^8/B^{10}	$B^2/B^8 = B/B^7$
depth 4, degree 16	error! B^{16}/B^{10}	$B^2/B^7 = B/B^6$

If $q_0 \approx B^n$, the maximal multiplicative depth is $n - \mathit{constant}$,
instead of $\log_2 n$ with a fixed modulus.

Modulus switching (big-endian)

Let q and q' be two moduli. Assume $\Delta' = \Delta q'/q$ is an integer.

Consider a big-endian encryption $(\mathbf{a}, b)$ modulo q of m:

$$b = \langle \mathbf{a}, \mathbf{s} \rangle + \Delta m + r \pmod{q}$$

Define

$$(\mathbf{a}', b') = \left(\left\lfloor \mathbf{a}\frac{q'}{q} \right\rceil, \left\lfloor b\frac{q'}{q} \right\rceil\right)$$

Assuming $\mathbf{s}$ has small coefficients (0, 1, -1), we can show

$$b' = \langle \mathbf{a}', \mathbf{s} \rangle + \Delta' m + \left\lfloor r\frac{q'}{q} \right\rceil + \varepsilon \pmod{q'}$$

with $|\varepsilon| = \mathcal{O}(n)$.

If Δ' remains large enough, $(\mathbf{a}', b')$ is an encryption modulo q' of m.

Leveled somewhat homomorphic encryption

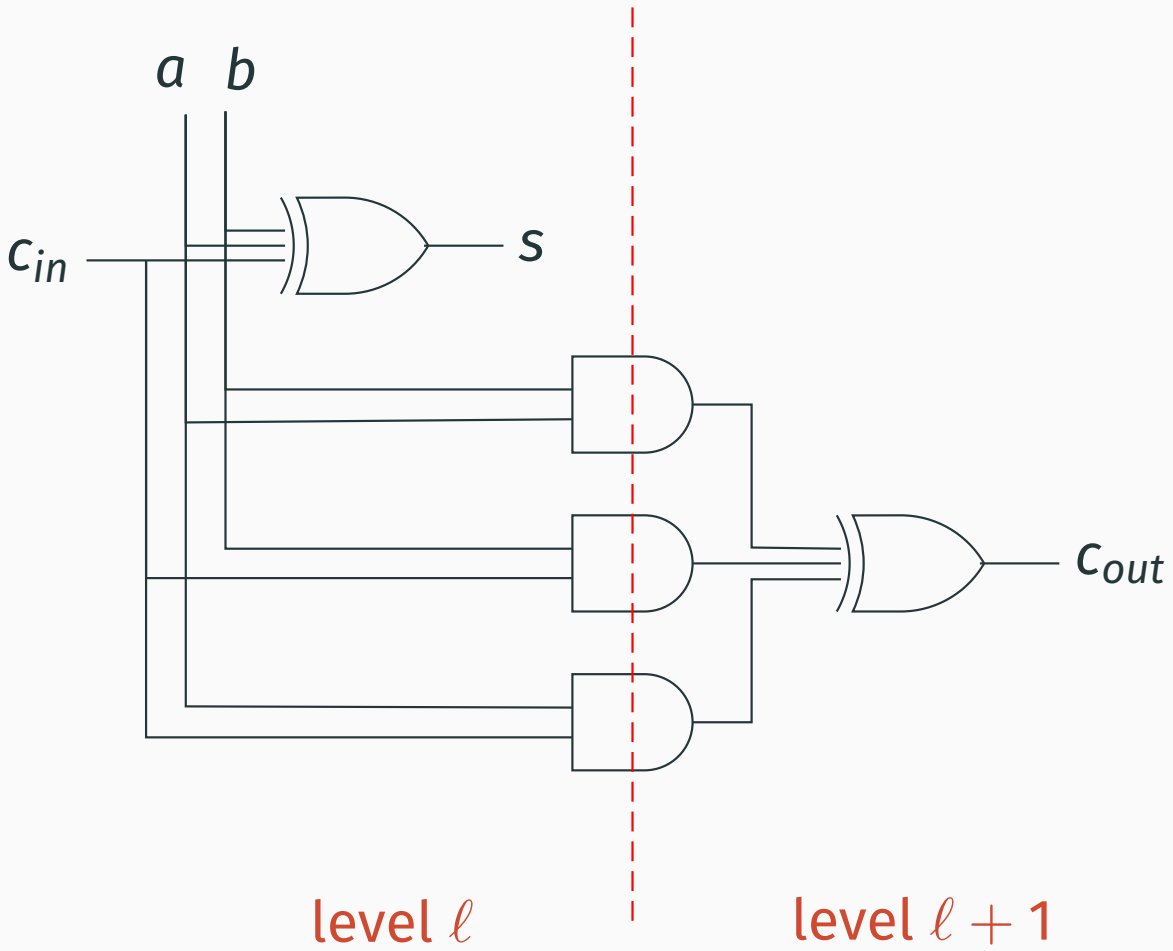
Fix a maximal multiplicative depth d.

The client chooses moduli $q_1, \ldots, q_d$ with $q_i/q_{i+1} \approx B$ (B minimal noise level) and private keys $\mathbf{s}_1, \ldots, \mathbf{s}_d$.

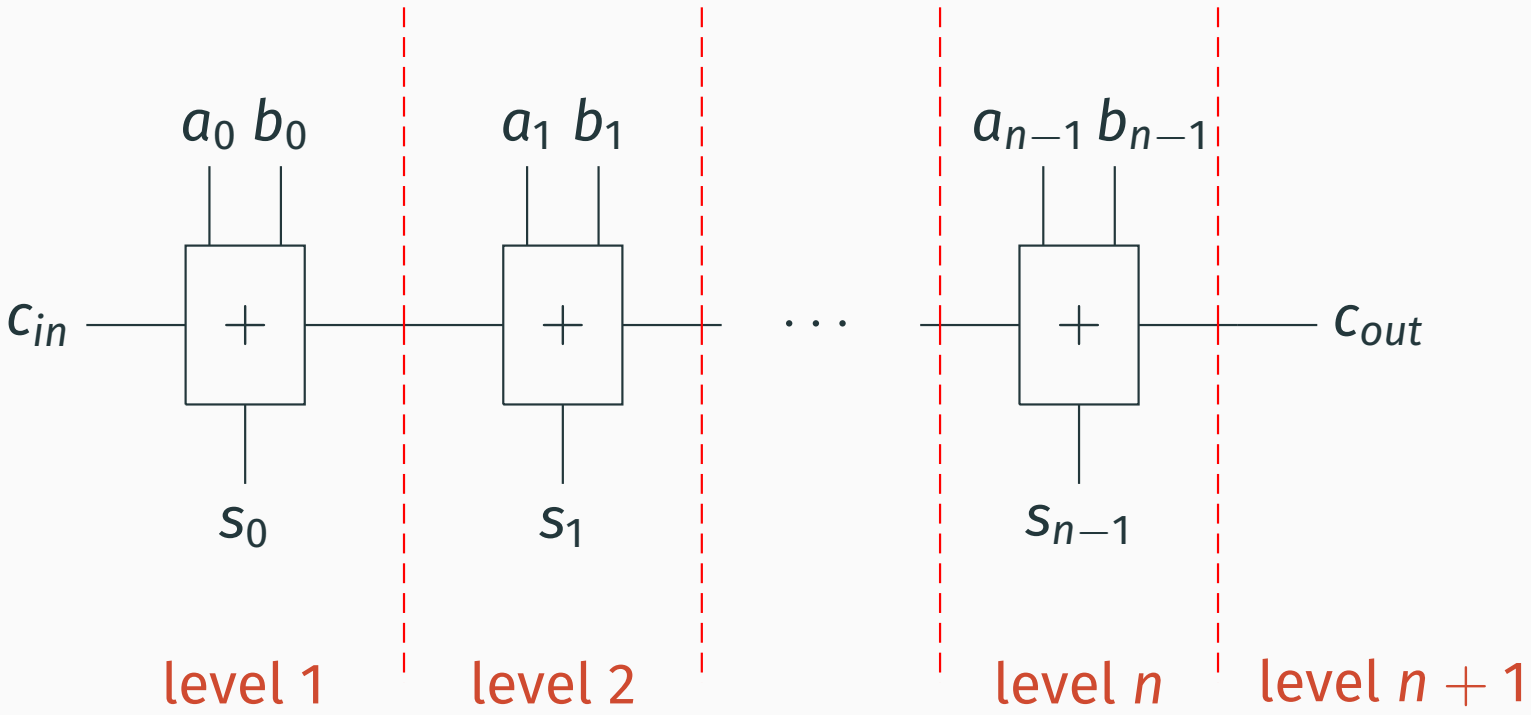
It gives the computer the q_i and the key switching matrices $\mathcal{E}_{\mathbf{s}_{i+1}}(\mathbf{s}_i \otimes \mathbf{s}_i)$.

The computer can homomorphically evaluate any circuit of multiplicative depth $\leq d$, switching key and modulus at each multiplication.

Examples of leveled circuit: the full adder



Example of leveled circuit: the n-bit adder



Programmable bootstrap: the FHEW/TFHE approach

Bootstrap vs. key change

Goal of a bootstrap: given a ciphertext c under key $\mathbf{s}$,
produce a ciphertext c' under key $\mathbf{s}'$ (possibly $= \mathbf{s}$)
such that

1. c' decrypts to the same message m as c;
2. the noise level of c' is reset to a known value, independent of that of c.

Key switching enforces (1) but not (2):
the noise level of c' is that of c + the noise of the switching.

Bootstrap = key switching + table lookup

(Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, Malika Izabachène, *Faster Fully Homomorphic Encryption: Bootstrapping in less than 0.1 seconds*, 2016)

Big-endian encryption: $b = \langle \mathbf{a}, \mathbf{s} \rangle + \Delta m + r$.

Decryption of the ciphertext $(\mathbf{a}, b)$:

1. Compute $e = b - \langle \mathbf{a}, \mathbf{s} \rangle$
2. Extract $m = \lfloor e/\Delta \rceil$

Bootstrap = key switching + table lookup

(Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, Malika Izabachène, *Faster Fully Homomorphic Encryption: Bootstrapping in less than 0.1 seconds*, 2016)

Big-endian encryption: $b = \langle \mathbf{a}, \mathbf{s} \rangle + \Delta m + r$.

Homomorphic decryption of the ciphertext $(\mathbf{a}, b)$:

1. Compute $e = b - \langle \mathbf{a}, \mathbf{s} \rangle$ encrypted under $\mathbf{s}'$
2. Extract $m = \lfloor e/\Delta \rceil$ encrypted under $\mathbf{s}'$, with minimal noise

Bootstrap = key switching + table lookup

(Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, Malika Izabachène, *Faster Fully Homomorphic Encryption: Bootstrapping in less than 0.1 seconds*, 2016)

Big-endian encryption: $b = \langle \mathbf{a}, \mathbf{s} \rangle + \Delta m + r$.

Homomorphic decryption of the ciphertext $(\mathbf{a}, b)$:

1. Compute $e = b - \langle \mathbf{a}, \mathbf{s} \rangle$ encrypted under $\mathbf{s}'$
2. Extract $m = \lfloor e/\Delta \rceil$ encrypted under $\mathbf{s}'$, with minimal noise

(1) is achieved by key switching, using $\mathcal{E}_{\mathbf{s}'}(\mathbf{s})$.

Bootstrap = key switching + table lookup

(Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, Malika Izabachène, *Faster Fully Homomorphic Encryption: Bootstrapping in less than 0.1 seconds*, 2016)

Big-endian encryption: $b = \langle \mathbf{a}, \mathbf{s} \rangle + \Delta m + r$.

Homomorphic decryption of the ciphertext $(\mathbf{a}, b)$:

1. Compute $e = b - \langle \mathbf{a}, \mathbf{s} \rangle$ encrypted under $\mathbf{s}'$
2. Extract $m = \lfloor e/\Delta \rceil$ encrypted under $\mathbf{s}'$, with minimal noise

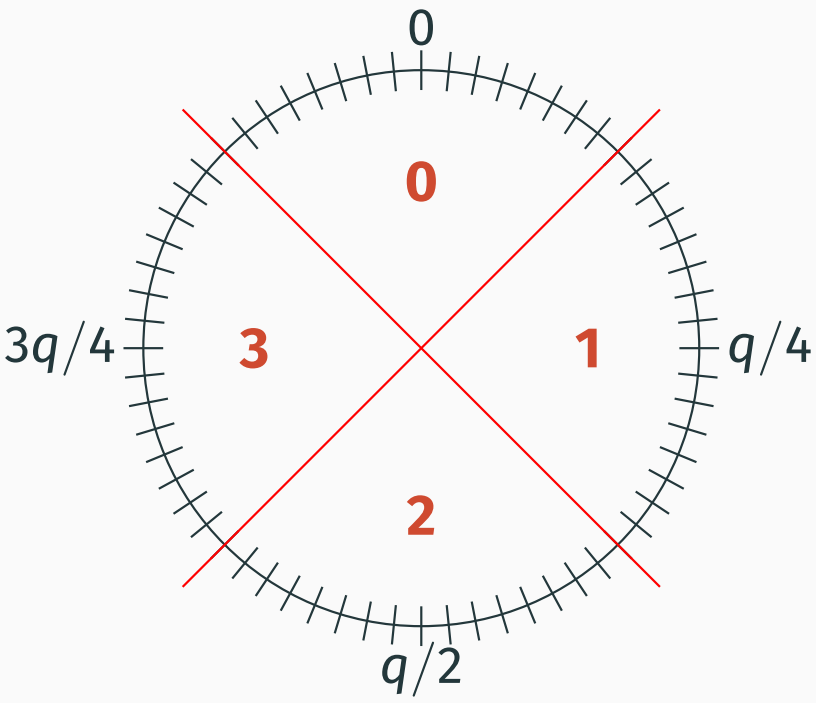
(1) is achieved by key switching, using $\mathcal{E}_{\mathbf{s}'}(\mathbf{s})$.

(2) is achieved by homomorphic lookup in a table

$i \in \mathbb{Z}_q \longmapsto$ an encryption of $\lfloor i/\Delta \rceil$ under $\mathbf{s}'$ with minimal noise

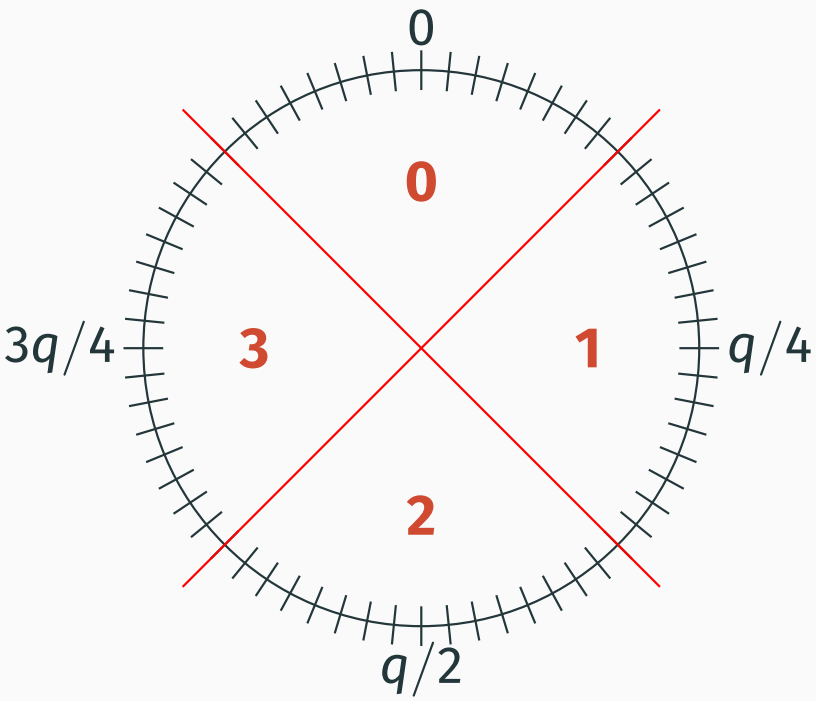
Torus and tables

Assuming $q = 64$ and $\Delta = 4$:

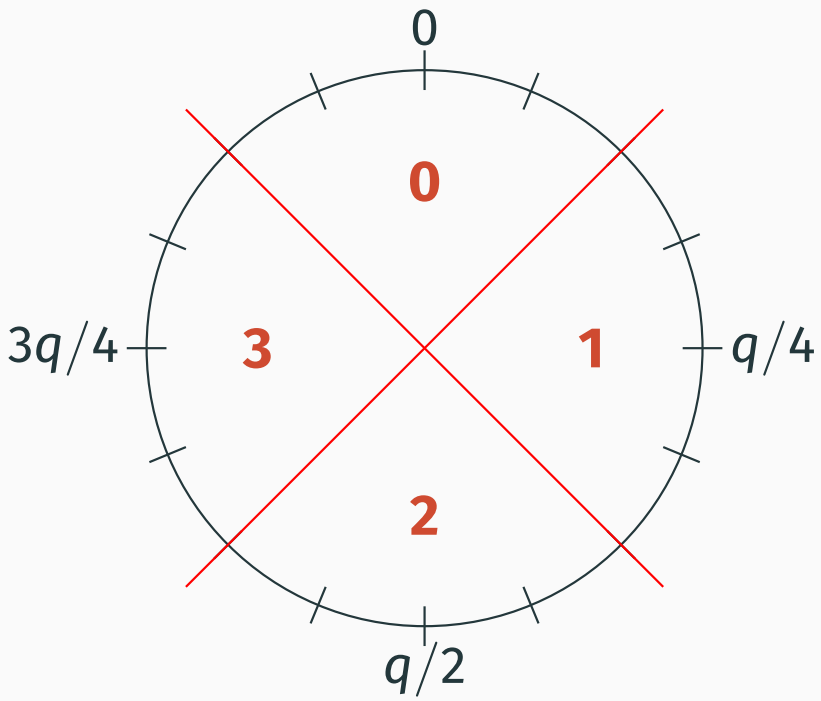


Torus and tables

Assuming $q = 64$ and $\Delta = 4$:



Reduction to $q = 16$:



We can make the table smaller by switching moduli beforehand.

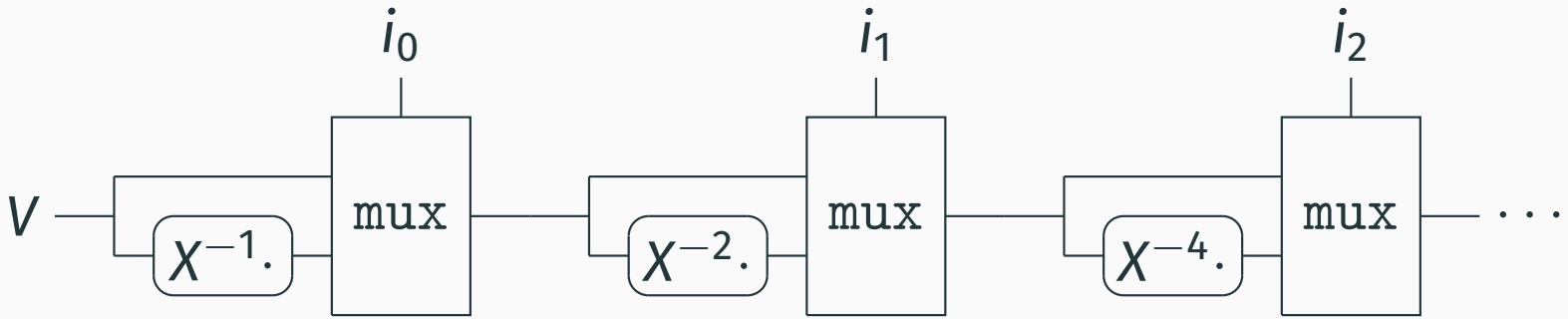
Indexing in a table

An N-entry table is represented by a polynomial V modulo $X^N + 1$.

To extract entry number $i \in [0, N)$:

- compute $X^{-i} \cdot V$ (right rotation of i slots)
- extract the constant coefficient of $X^{-i} \cdot V$.

If we have a binary decomposition of $i = i_0 + 2i_1 + \cdots + 2^n i_n$, we can use an barrel shifter:



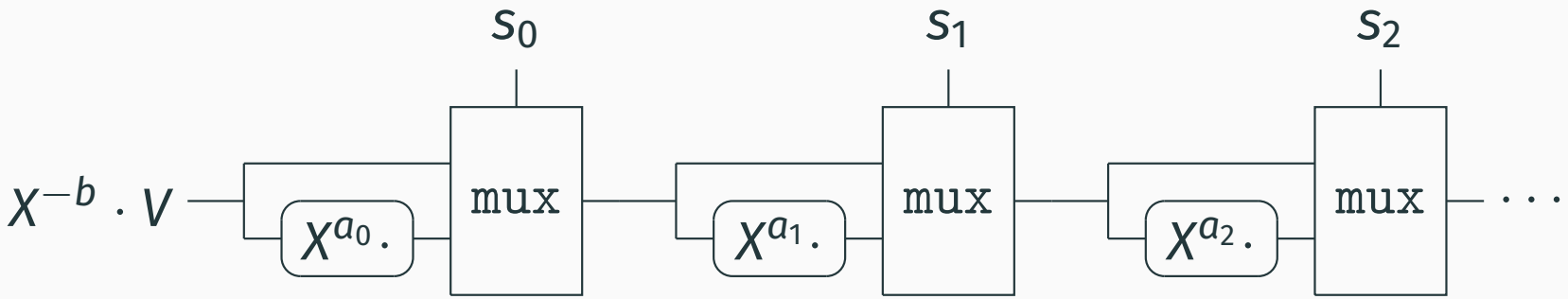
where $\texttt{mux}(i, a, b) = a$ if $i = 0$ and $= b$ if $i = 1$.

Application to homomorphic decryption

After modulus switching if necessary, the ciphertext is $(\mathbf{a}, b)$ and the index in table V is $i = b - \langle \mathbf{a}, \mathbf{s} \rangle$.

We want to multiply V by $X^{-i} = X^{-b+a_0 s_0 + \cdots + a_{n-1} s_{n-1}}$.

Assume the s_i components of the key are 0/1 bits.
We reuse the barrel shifter design as follows:



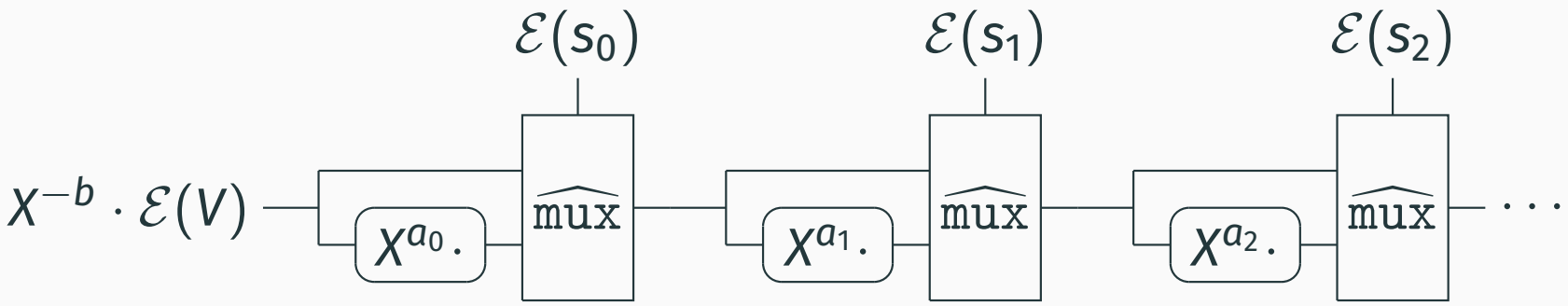
Application to homomorphic decryption

The $\texttt{mux}$ gate can be evaluated homomorphically:

$$\widehat{\texttt{mux}}(i, a, b) = i \otimes (b \ominus a) \oplus a$$

where $\oplus$, $\ominus$, $\otimes$ are addition, subtraction and multiplication of two ciphertexts.

Hence, we can evaluate the barrel shifter homomorphically:



The dot $\cdot$ is cleartext $\times$ ciphertext multiplication.
The $\mathcal{E}(s_i)$ are encryptions of the bits of the key $\mathbf{s}$.

Bootstrap accomplished !

The output of the barrel shifter is

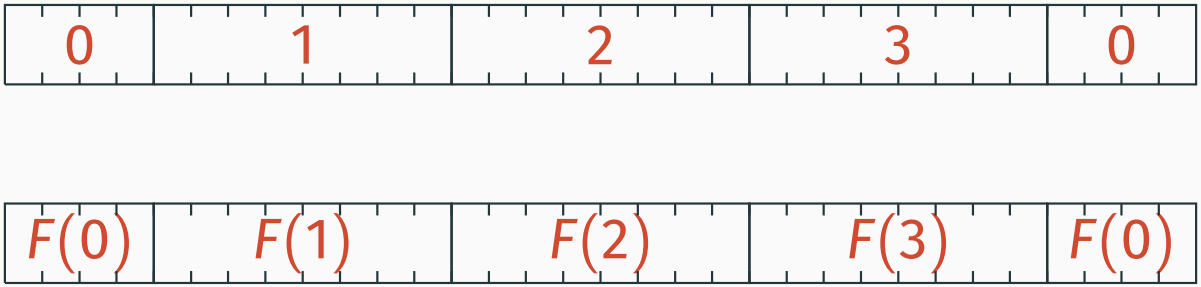
$$X^{-b+a_0\cdot\mathcal{E}(s_0)+\cdots+a_{n-1}\cdot\mathcal{E}(s_{n-1})} \cdot \mathcal{E}(V)$$

from which we can extract the constant coefficient $\mathcal{E}(v_i)$
where $i = b - \langle \mathbf{a}, \mathbf{s} \rangle = \Delta m + r$ (the noise term),
and v_i the i-th entry of the table V:

$$V = \sum_{i=0}^{N-1} v_i X^i \quad \text{where} \quad v_i = \lfloor i/\Delta \rceil$$

The end result is a low-noise encryption of $\mathcal{E}(m)$.

Programmable bootstrap



The table V used for bootstrapping is

$$V = \sum_{i=0}^{N-1} v_i X^i \quad \text{where} \quad v_i = \lfloor i/\Delta \rceil$$

But we can also take $v_i = F(\lfloor i/\Delta \rceil)$ for any function $F : \mathbb{Z}_t \to \mathbb{Z}_t$.

Then, the bootstrap procedure evaluates F homomorphically while reducing the noise.

Applications of programmable bootstrap

Many computations can be expressed as a unary function applied to a linear combination of the inputs.

Minimum and maximum:

$$\max(x, y) = x + \texttt{relu}(y - x)$$
$$\min(x, y) = y - \texttt{relu}(y - x)$$

$$\text{where } \texttt{relu}(z) = \begin{cases} z & \text{if } z \geq 0 \\ 0 & \text{if } z < 0 \end{cases}$$

Artificial neuron:

$$\varphi(w_1 x_1 + \cdots + w_n x_n) \quad \text{where } \varphi \text{ is the activation function}$$

Multiplication:

$$x \cdot y = F(x + y) - F(x - y) \quad \text{where } F(z) = z^2/4$$

Summary

Advances in homomorphic encryption

Starting with Gentry's 2009 conceptual breakthrough, major progress was made in three directions:

- Efficient somewhat homomorphic ciphers, able to homomorphically evaluate circuits of respectable multiplicative depth.

  (Brakerski-Gentry-Vaikuntanathan 2011, Brakerski-Fan-Vercauteren 2012, Gentry-Sahai-Waters 2013, etc.)

- Efficient, programmable bootstrap: FHEW, TFHE.

  (Chillotti, Gama, Georgieva, Izabachène, 2016)

  $\Rightarrow$ Ilaria Chillotti's seminar talk

- Approximate homomorphic computation over reals and complex numbers: CKKS (Cheon-Kim-Kim-Song 2016)

  $\Rightarrow$ Damien Stehlé's seminar talk

Practical implementations

> *I don't think we'll see anyone using Gentry's solution in our lifetimes.*
>
> (Butler Lampson)

Libraries: HElib, SEAL, TFHE, OpenFHE, HEEAN, ...

(see https://github.com/jonaschn/awesome-he for a list)

Reasonable but still insufficient performance on CPUs

(homomorphic evaluation of an AES decryption: 30 sec.)

Under development: GPU and hardware (FPGA, ASIC) implementations.

References

References

Advanced introduction to TFHE:

- Ilaria Chillotti, *TFHE deep dive*, 2022.
  https://www.zama.org/post/tfhe-deep-dive-part-1

For reference:

- Ronny Ko, *The Beginner's Textbook for Fully Homomorphic Encryption*, 2025.
  https://arxiv.org/abs/2503.05136 https://fhetextbook.github.io/
- Oded Regev, *The Learning with Errors Problem*, 2010.
  http://www.cims.nyu.edu/~regev/papers/lwesurvey.pdf