Secure computing, second lecture

Fully homomorphic encryption: computing on encrypted data (part 1)

Xavier Leroy

2025-11-13

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

Reminders about encryption

The interface of an asymmetric cipher

Key generation:

$$\mathit{Keygen} : \text{public key} \times \text{private key} \qquad \text{(randomized)}$$

Encryption:

$$\mathcal{E} : \text{public key} \times \text{plaintext} \to \text{ciphertext} \qquad \text{(randomized)}$$

Decryption:

$$\mathcal{D} : \text{private key} \times \text{ciphertext} \to \text{plaintext} \qquad \text{(deterministic)}$$

Functional correctness

$$\begin{aligned} \mathit{Keygen} &: \text{public key} \times \text{private key} && \text{(randomized)} \\ \mathcal{E} &: \text{public key} \times \text{plaintext} \to \text{ciphertext} && \text{(randomized)} \\ \mathcal{D} &: \text{private key} \times \text{ciphertext} \to \text{plaintext} && \text{(deterministic)} \end{aligned}$$

Intuitively: $\mathcal{D}_{sk}(\mathcal{E}_{pk}(m)) = m.$

More precisely:

$$\mathcal{D}_{sk}(c) = m \qquad \text{for all } (pk, sk) \leftarrow \mathit{Keygen},\ c \leftarrow \mathcal{E}_{pk}(m)$$

Using probabilities:

$$\Pr\left[(pk, sk) \leftarrow \mathit{Keygen}; \mathcal{D}_{sk}(\mathcal{E}_{pk}(m)) = m\right] = 1$$

With explicit randomness r, r':

$$\mathcal{D}_{sk}(\mathcal{E}_{pk}(m, r)) = m \qquad \text{if } (pk, sk) = \mathit{Keygen}(r')$$

Security of a cipher

Unconditional, information-theoretic security: (informally)

An attacker who knows (cleartext, ciphertext) pairs and a ciphertext cannot deduce any information on the corresponding cleartext.

Security of a cipher

Unconditional, information-theoretic security: (informally)

An attacker who knows (cleartext, ciphertext) pairs and a ciphertext cannot deduce any information on the corresponding cleartext.

Computational security: (informally)

An attacker in time polynomial in the size n of keys
who knows (cleartext, ciphertext) pairs and a ciphertext
has a probability $\leq \varepsilon(n)$ to deduce some information on the corresponding cleartext, where ε is a negligible function.

(Negligible = decreasing rapidly with n, e.g. 2^{-n})

Attack models

- **Known plaintext attacks (KPA)**: The attacker obtained (cleartext, ciphertext) pairs but did not choose them.
- **Chosen plaintext attacks (CPA)**: The attacker can encrypt any cleartext of their choice. (E.g. public key.)
- **Chosen ciphertext attacks (CCA)**: CPA + the attacker can ask for decryption of any ciphertext of their choice except the one that is under attack.

Security objectives

- **Indistinguishability (IND)**: the attacker is unable to distinguish a ciphertext from noise, or a ciphertext from another ciphertext.

- **Non-malleability (NM)**: given a ciphertext c, the attacker cannot produce a ciphertext c' that modifies the plaintext in a controlled manner, e.g. such that $\mathcal{D}(c') = \mathcal{D}(c) \oplus 1$.

The IND-CPA criterion (indistinguishability with chosen plaintexts)

Presented as a two-player game, the attacker A and the challenger C.

> C: draws $(pk, sk) \leftarrow Keygen$, sends pk to A.
>
> A: chooses two plaintexts m_1, m_2, sends them to C.
>
> C: chooses $i \in \{1, 2\}$, sends $c = \mathcal{E}(pk, m_i)$ to A.
>
> A: tries to guess whether c is the encryption of m_1 or m_2.

The cipher is IND-CCA if the probability that the attacker guesses correctly is no more than $1/2 + \varepsilon(n)$.

(Note: a cipher with deterministic encryption is not IND-CPA.)

Provable security

By reduction to a mathematical problem that is believed to be algorithmically hard:

Attack that succeeds
with non-negligible probability

reduction

Probabilistic polynomial time algorithm
that solves a hard problem

Examples of algorithmically-hard problems

For a classical computer:

- Factoring: given N, find $p, q < N$ such that $N = pq$.
- The RSA problem: given N and e coprime with $\varphi(N)$ and $c < N$, find m such that $m^e = c \bmod N$.
- Discrete logarithm: given g a generator of a group G and $a \in G$, find $x \in \mathbb{N}$ such that $g^x = a$.
- The Diffie-Hellman problem: given g^x and g^y (for unknown x, y), compute g^{xy}.

Examples of algorithmically-hard problems

For a classical computer:

- Factoring: given N, find $p, q < N$ such that $N = pq$.
- The RSA problem: given N and e coprime with $\varphi(N)$ and $c < N$, find m such that $m^e = c \bmod N$.
- Discrete logarithm: given g a generator of a group G and $a \in G$, find $x \in \mathbb{N}$ such that $g^x = a$.
- The Diffie-Hellman problem: given g^x and g^y (for unknown x, y), compute g^{xy}.

For a quantum computer:

- The MQ problem: solve a system of degree-2 polynomial equations with multiple unknowns.

Quantifying security

Choose ciphers and key sizes such that the best known attack takes time $\approx 2^{\lambda}$.

λ is the security parameter.

Some recommendations by NIST:

Security parameter λ	112	128	192	256
Symmetric encryption	3DES	AES-128	AES-192	AES-256
Hashing	SHA-224	SHA-256	SHA-384	SHA-512
RSA (factoring)	2048	3072	7680	15360
Elliptic curves (discr. log.)	224+	256+	384+	512+
ML-DSA (post-quantum)	—	1312 / 2560	1952 / 4032	2592 / 4896

Weakly homomorphic encryption

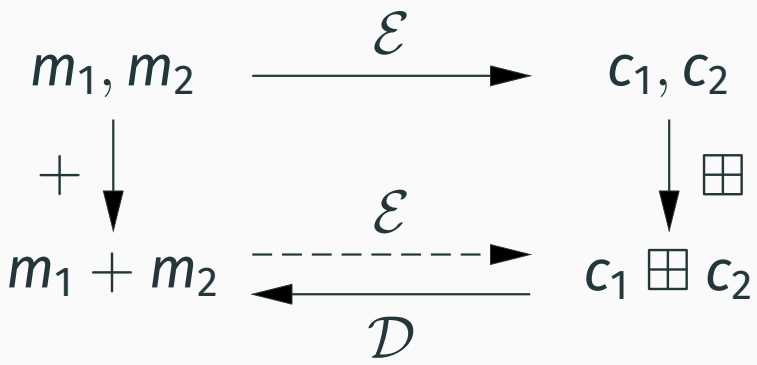
Weakly homomorphic encryption

A cipher that is homomorphic for one operation only, typically addition or multiplication of plaintexts but not both.

In the additive case, we have a $\boxplus$ operation on ciphertexts such that "the encryption of a sum is the $\boxplus$ of the ciphertexts":

$$c_1 \boxplus c_2 \in \mathcal{E}(m_1 + m_2) \quad \text{for all } c_1 \in \mathcal{E}(m_1),\ c_2 \in \mathcal{E}(m_2)$$

It follows that $m_1 + m_2 = \mathcal{D}(\mathcal{E}(m_1) \boxplus \mathcal{E}(m_2))$.



Unpadded RSA is homomorphic for multiplication

The RSA cipher: let p, q be large prime numbers;
— public key: $N = pq$ and e coprime with $(p-1)(q-1)$;
— private key: $d = e^{-1} \pmod{(p-1)(q-1)}$.

$$\mathcal{E}(m) \stackrel{def}{=} m^e \bmod N \qquad \mathcal{D}(c) \stackrel{def}{=} c^d \bmod N$$

If m_1, m_2 are two plaintext messages,

$$\mathcal{E}(m_1) \cdot \mathcal{E}(m_2) = m_1^e \cdot m_2^e = (m_1 \cdot m_2)^e = \mathcal{E}(m_1 \cdot m_2) \pmod{N}$$

Multiplication modulo N has multiplication modulo N as its homomorphic operation.

Unusable: without padding, RSA is not IND-CPA; but all known random padding scheme break the homomorphism.

The ElGamal cipher (EG)

A finite group $(G, \cdot)$ of order q generated by g.

Private key: $x \in \{1, \ldots, q-1\}$. Public key: $h \stackrel{def}{=} g^x$.

Randomized encryption (IND-CCA):

$$\mathcal{E}(m) = (g^r, h^r \cdot m) \quad \text{with } r \in \{1, \ldots, q-1\} \text{ random}$$

Note: cleartext messages are elements of G.

Decryption: $\mathcal{D}((a, b)) = (a^x)^{-1} \cdot b$

Homomorphic for the group operation:

$$\begin{aligned} \mathcal{E}(m_1) \cdot \mathcal{E}(m_2) &= (g^{r_1} \cdot g^{r_2}, (h^{r_1} \cdot m_1) \cdot (h^{r_2} \cdot m_2)) \\ &= (g^r, h^r \cdot (m_1 \cdot m_2)) \qquad (\text{with } r = r_1 + r_2 \bmod q) \\ &= \text{an encryption of } m_1 \cdot m_2 \end{aligned}$$

The exponential ElGamal cipher (EEG)

Cleartext messages are integers $m \in \{0, \ldots, q-1\}$,
which we map to elements of G by exponentiation g^m.

Encryption:

$$\mathcal{E}(m) = EG.\mathcal{E}(g^m) = (g^r, h^r \cdot g^m) \quad \text{with } r \text{ random}$$

Decryption:

$$\mathcal{D}(c) = \log_g(EG.\mathcal{D}(c)) \quad \text{(base-}g\text{ discrete logarithm)}$$

Practical for small messages m (votes, monetary amounts).

Homomorphic for the sum of messages (modulo q):

$$\begin{aligned} \mathcal{E}(m_1) \cdot \mathcal{E}(m_2) &= EG.\mathcal{E}(g^{m_1}) \cdot EG.\mathcal{E}(g^{m_2}) \\ &= \text{an EG encryption of } g^{m_1} \cdot g^{m_2} = g^{m_1+m_2} \\ &= \text{an EEG encryption of } (m_1 + m_2) \bmod q \end{aligned}$$

Paillier's cipher

Exploits the fact that some discrete logarithms modulo n^2 are easily computed:

$$(1+n)^x = 1 + xn + \binom{x}{2} n^2 + \ldots = 1 + xn \pmod{n^2}$$

hence $L((1+n)^x \bmod n^2) = x$ where $L(u) \stackrel{def}{=} (u-1)/n$.

Key generation: p, q prime numbers of the same length.

Public key: (n, g) where $n = pq$ and $g = 1 + n$.

Private key: (α, β) where $\alpha = \varphi(n) = (p-1)(q-1)$
and $\beta = L(g^\alpha \bmod n^2)^{-1} \bmod n$.

Paillier's cipher

Encryption: $\mathcal{E}(m) = g^m \cdot r^n \bmod n^2$
with $r \in \{1, \ldots, n-1\}$ random and coprime with n.

Decryption: $\mathcal{D}(c) = L(c^\alpha \bmod n^2) \cdot \beta \bmod n$

This cipher is homomorphic for addition modulo n and for multiplication by a constant k:

$$\begin{aligned}
\mathcal{E}(m_1) \cdot \mathcal{E}(m_2) \bmod n^2 &= g^{m_1+m_2}(r_1 \cdot r_2)^n \bmod n^2 \\
&= \text{an encryption of } (m_1 + m_2) \bmod n \\
\mathcal{E}(m)^k \bmod n^2 &= g^{km} r^{kn} \bmod n^2 \\
&= \text{an encryption of } km \bmod n
\end{aligned}$$

No relation between $\mathcal{E}(m_1 m_2 \bmod n)$ and $\mathcal{E}(m_1), \mathcal{E}(m_2)$.

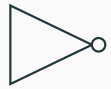
Somewhat homomorphic encryption

Fully homomorphic encryption (FHE) and Boolean circuits

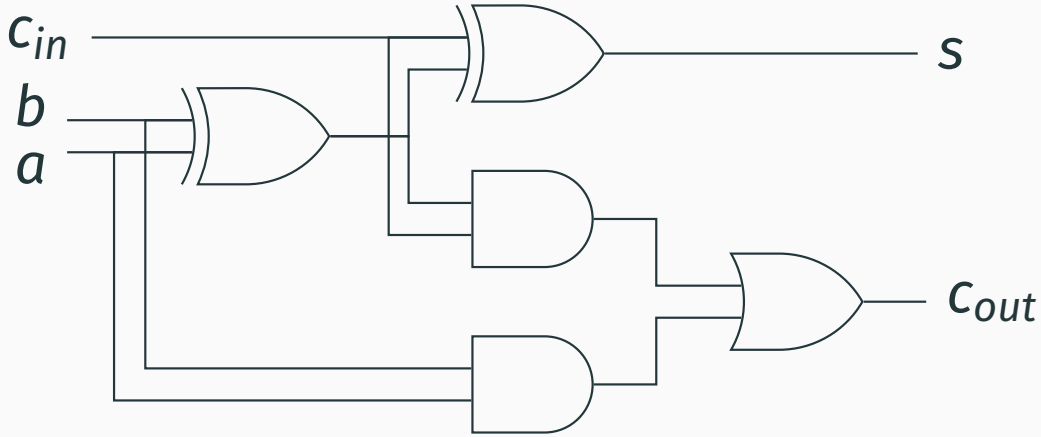
A cipher that is homomorphic for addition **and** multiplication (of integers modulo p, or just modulo 2) is said to be fully homomorphic, since it is able to evaluate

- polynomials over several variables;
- all Boolean combinatorial circuits.

Arithmetization of circuits:

Logic gate	modulo 2	modulo $p > 2$
NOT 	$x + 1$	$1 - x$
AND 	xy	xy
OR 	$x + y + xy$	$1 - (1 - x)(1 - y)$
XOR 	$x + y$	$x + y - 2xy$

Example of a circuit: 1-bit full adder

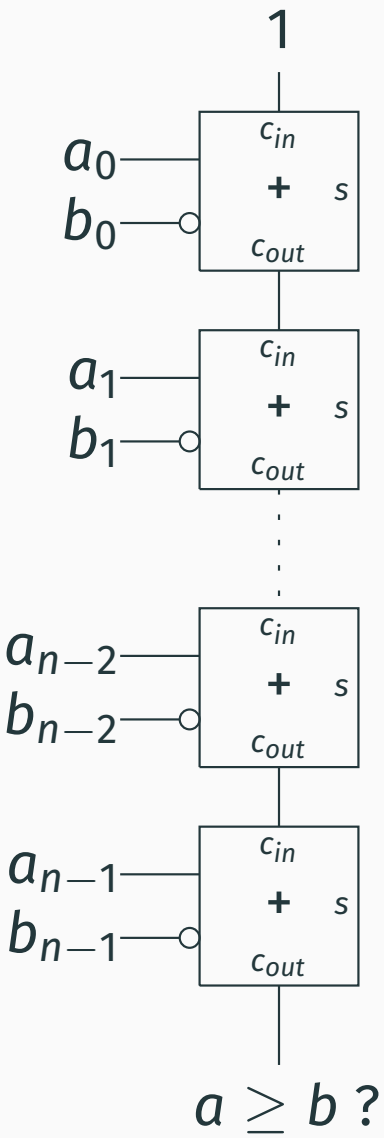


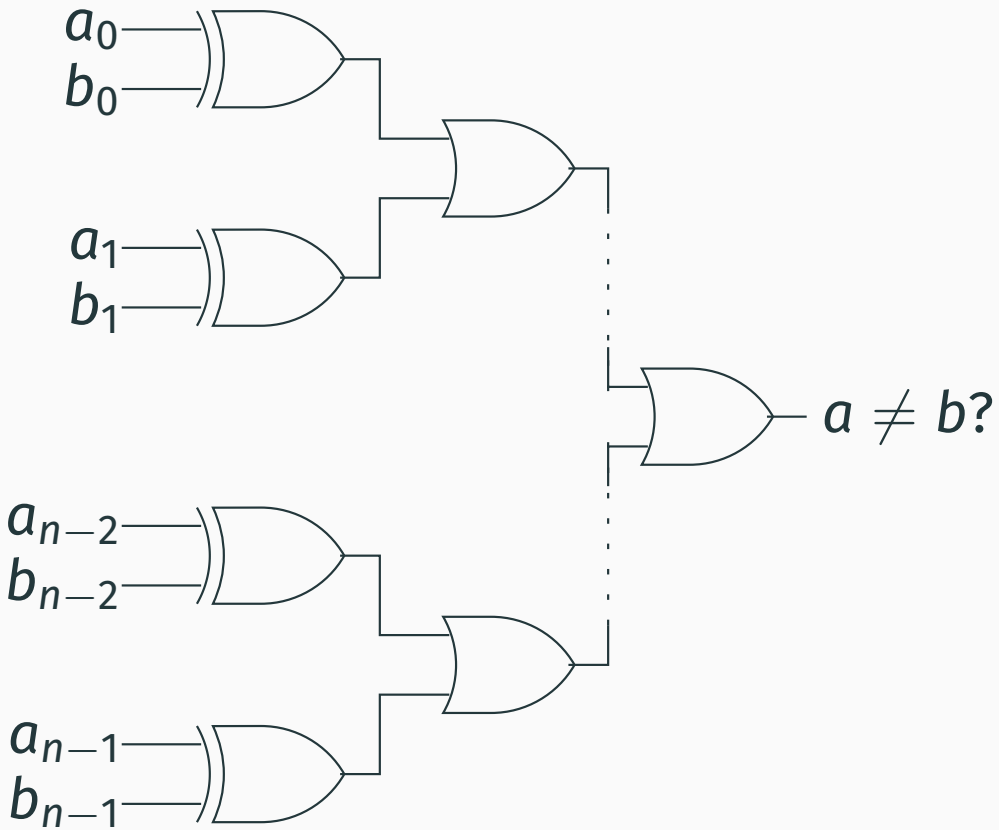
The corresponding polynomials:

$$s = a + b + c_{in} \pmod 2$$

$$c_{out} = ab + (a + b)c_{in} + ab(a + b)c_{in} = ab + ac_{in} + bc_{in} \pmod 2$$

Example of circuits: n-bit comparators





Fully homomorphic encryption is hard

No known cipher is secure and fully homomorphic.

Rabin: May I add a remark? If you then propose to have different p and q for each customer, which is quite difficult and impractical, sometimes a non-innocent by-stander has knowledge of how much money you deposited. The other problem again exists. You must assume at least spotty partial information about the data which is going to be protected.

Rivest: All the systems I've presented, I think, are susceptible to variations of that kind of attack. I do not consider any of them very satisfactory for precisely those kinds of reasons.

(Questions and answers on the seminal article *On data banks and privacy homomorphisms* by Rivest, Adleman, Dertouzos, 1978.)

Somewhat homomorphic encryption (SHE)

We can construct somewhat homomorphic ciphers: homomorphic for a limited number of additions and multiplications.

This makes it possible to homomorphically evaluate circuits of bounded depth.

The bootstrap procedure (introduced by Craig Gentry in 2009) obtains fully homomorphic encryption from a suitable somewhat homomorphic cipher.

Encryption = adding noise

An approach followed by several recent cryptosystems (based on Euclidean lattices or the LWE problem, for example).

$$\text{Encryption} \quad = \quad \text{adding noise to the cleartext.}$$

The noise has a hidden structure that enables us to cancel the noise if we know the private key:

$$\text{Decryption} \quad = \quad \begin{array}{l} \text{reducing the noise with the help of the private key;} \\ \text{recovering the nearest cleartext} \end{array}$$

A “noisy” cipher based on integer arithmetic

(van Dijk, Gentry, Halevy, Vaikuntanathan, *Fully homomorphic encryption over the integers*, Eurocrypt 2010)

Private key: an odd integer p.

Symmetric encryption (using the private key) of a bit b:

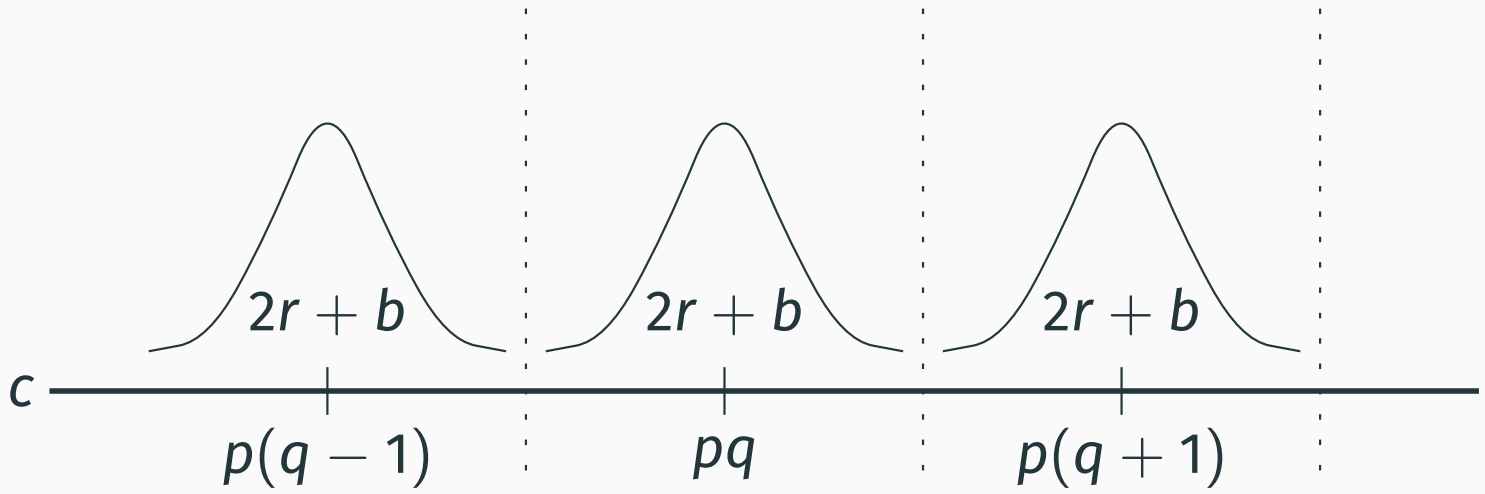
$$\mathcal{E}_p(b) = pq + 2r + b \qquad \begin{array}{l} q \text{ random nonnegative integer} \gg p \\ r \text{ random integer, } |r| < p/4 \end{array}$$

Decryption:

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

where $\lfloor \cdot \rceil$ is rounding to the nearest integer, and $\oplus$ is xor.

Encryption and decryption, graphically



From the ciphertext c and the private key p, we recover $q = \lfloor c/p \rceil$, and

$$(c - pq) \bmod 2 = (2r + b) \bmod 2 = b$$

Since p is odd, pq is odd iff q is odd, therefore

$$b = LSB(c - pq) = LSB(c) \oplus LSB(pq) = LSB(c) \oplus LSB(q)$$

(Note: q is enough to mask b; r is added to thwart the approximate GCD attack.)

The approximate GCD problem

Given numbers x_i of the form $pq_i + e_i$ with $|e_i| \leq E$,
can we find p ?

If $E = 0$: it is enough to compute $\gcd(x_i, x_j)$ for a few $i \neq j$.

If $E = 1$: likewise, but consider $x_i + 1$ and $x_i - 1$ as well.

If E, p, q_i are large: known algorithms are exponential in E.

Recommended sizes by van Dijk *et al*:

$$e_i : \lambda \text{ bits} \qquad p : \lambda^2 \text{ bits} \qquad q_i : \lambda^5 \text{ bits}$$

where λ is the security parameter.

Public key encryption

Public key pk = a set of encryptions of zero $pq_i + 2r_i$
(with q_i random and r_i random, $|r_i| \ll p$).

Public key encryption:

$$\mathcal{E}_{pk}(b) = \sum_{x \in S} x + 2r + b$$

S random subset of pk

r random integer, $|r| < p/4$

van Dijk *et al* recommend to use λ^5 encryptions of 0 (??), resulting in a public key of size λ^{10} bits (!!).

Note: if b is public or already encrypted, we can use a trivial encryption

$$\mathcal{E}_0(b) = b$$

which decrypts to b for any private key p.

Somewhat homomorphic encryption

For addition:

$$
\begin{aligned}
&(pq_1 + 2r_1 + b_1) + (pq_2 + 2r_2 + b_2) \\
&\quad = \quad p(q_1 + q_2) + 2(r_1 + r_2 + (b_1 + b_2)/2) + (b_1 + b_2) \bmod 2 \\
&\quad = \quad \text{an encryption of } b_1 \oplus b_2 \text{ if } r_1 + r_2 \text{ is small enough}
\end{aligned}
$$

For multiplication:

$$
\begin{aligned}
&(pq_1 + 2r_1 + b_1) \cdot (pq_2 + 2r_2 + b_2) \\
&\quad = \quad p(\cdots) + 2(2r_1r_2 + r_1b_2 + r_2b_1) + b_1b_2 \\
&\quad = \quad \text{an encryption of } b_1 \wedge b_2 \text{ if } r_1r_2 \text{ is small enough}
\end{aligned}
$$

Noise management

The noise $N(c)$ of a ciphertext is

$$N(c) = \text{number of bits of } r \quad \text{where } c = pq + 2r + b$$

In order for c to correctly decrypt to b, we need

$$N(c) < \text{number of bits of } p - 2$$

Homomorphic addition increases noise slowly:
$r_1, r_2 \mapsto r_1 + r_2 + (b_1 + b_2)/2$, hence

$$N(c_1 + c_2) \leq \max(N(c_1), N(c_2)) + 1$$

Homomorphic multiplication increases noise rapidly:
$r_1, r_2 \mapsto 2(2r_1r_2 + r_1b_2 + r_2b_1)$, hence

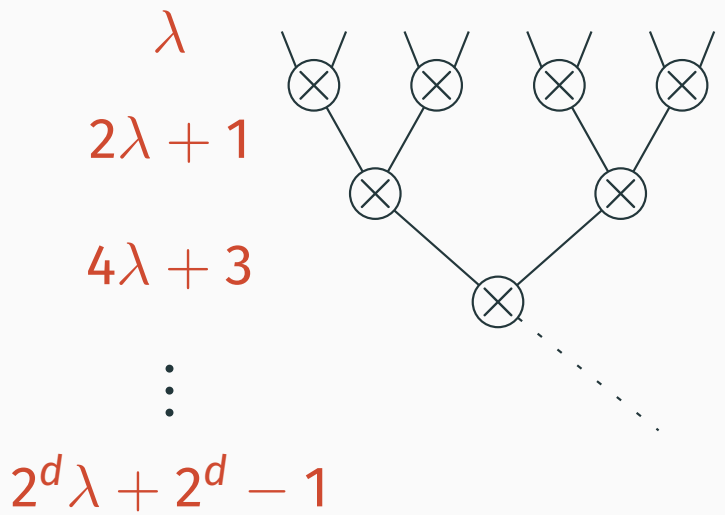
$$N(c_1 \cdot c_2) \leq N(c_1) + N(c_2) + 1$$

Multiplicative depth of a circuit

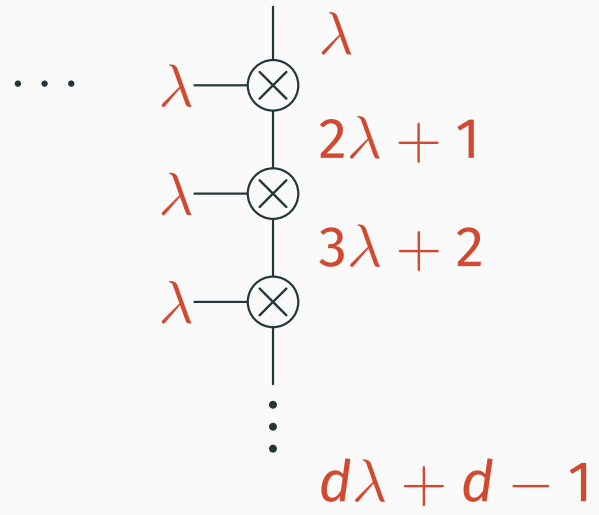
Multiplicative depth = max number of AND/OR gates between an input and an output.

If the initial noise is λ bits, and if p has λ^2 bits, the multiplicative depth of an homomorphically-evaluated circuit is limited to $\approx \log_2 \lambda$ in the worst case and $\approx \lambda$ in the best case.

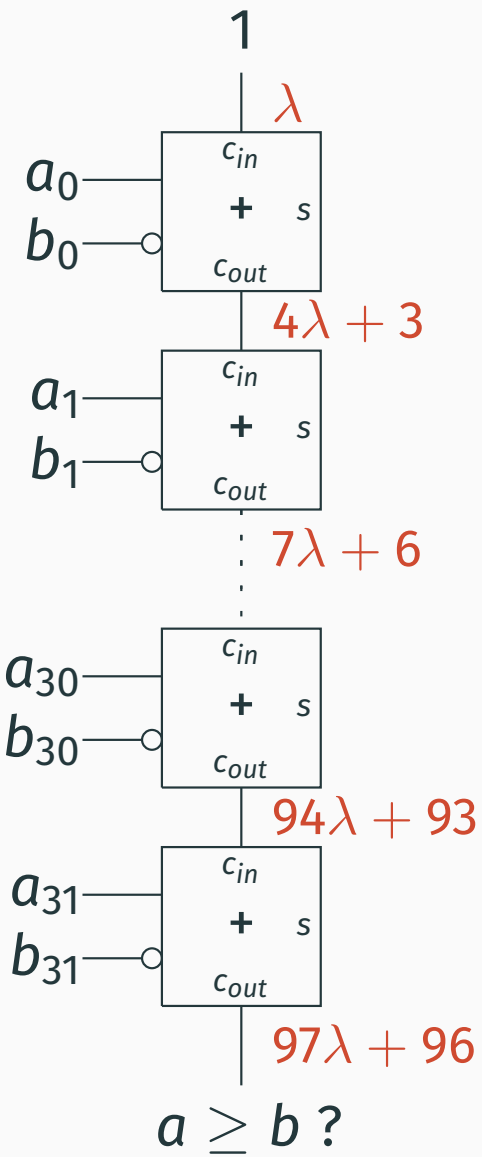
Exponential noise growth:

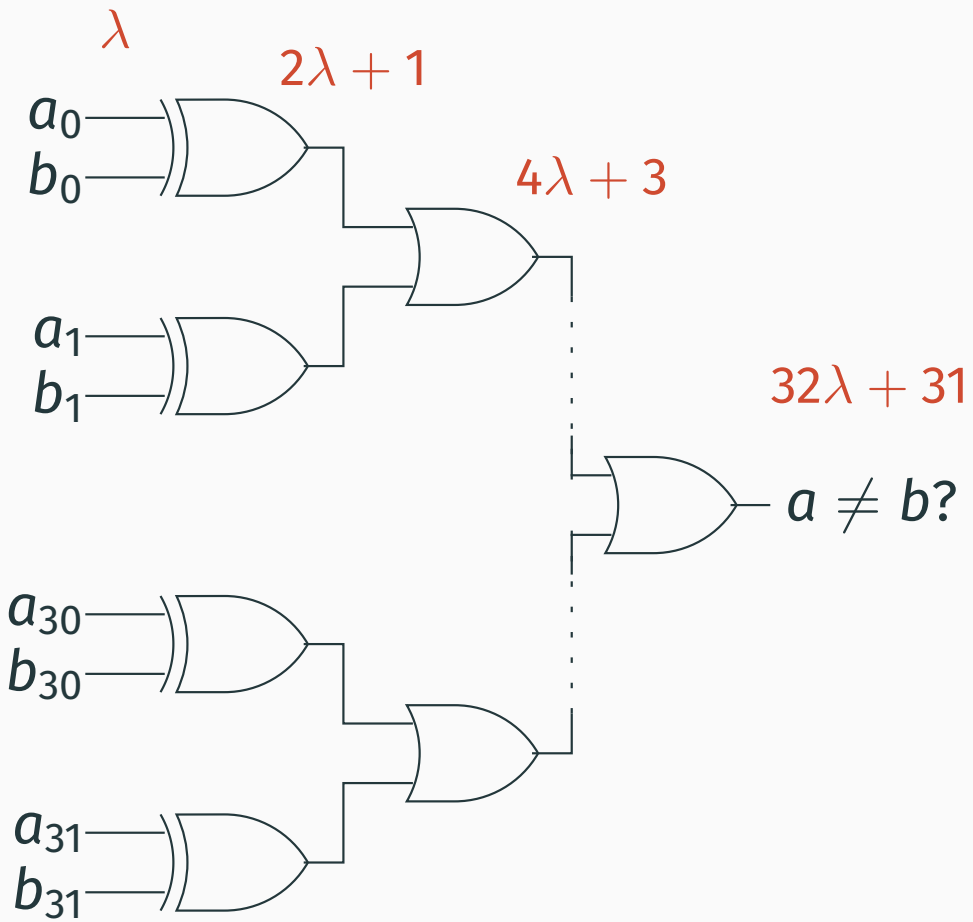


Linear noise growth:



Noise in a 32-bit comparator circuit

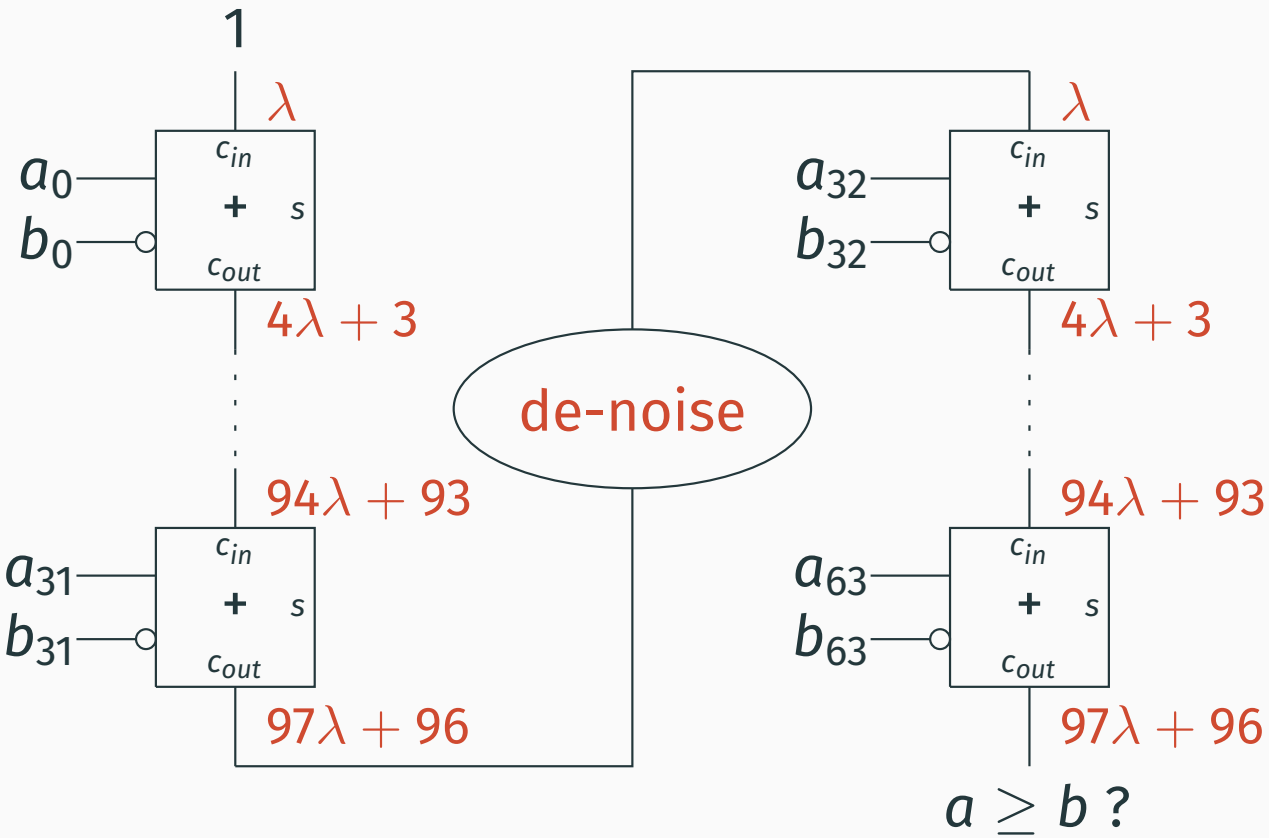




Fully homomorphic encryption via bootstrapping

Reducing noise at points of a circuit

To evaluate arbitrary circuits homomorphically, we must be able to *reduce noise* at the points of the circuit where noise can go over threshold.



Noise reduction by decryption and re-encryption

How can we reduce noise?

It suffices to decrypt then re-encrypt!

$$pq + 2r + b \xrightarrow{\mathcal{D}_p} b \xrightarrow{\mathcal{E}_p} pq' + 2r' + b$$

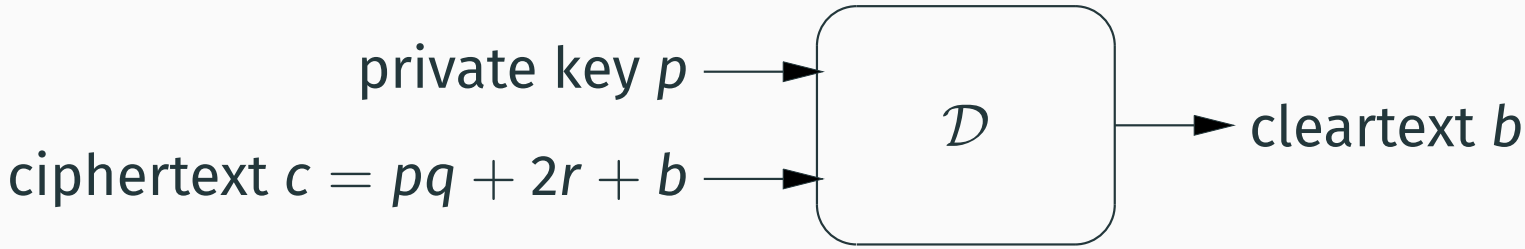
Very noisy ciphertext
$(r \approx p/4)$

Minimally noisy ciphertext
$(r' \ll p/4)$

Problem: the circuit evaluator doesn't know the decryption key p.

Noise reduction by homomorphic decryption

(Craig Gentry, *A fully homomorphic encryption scheme*, PhD, Stanford, 2009.)



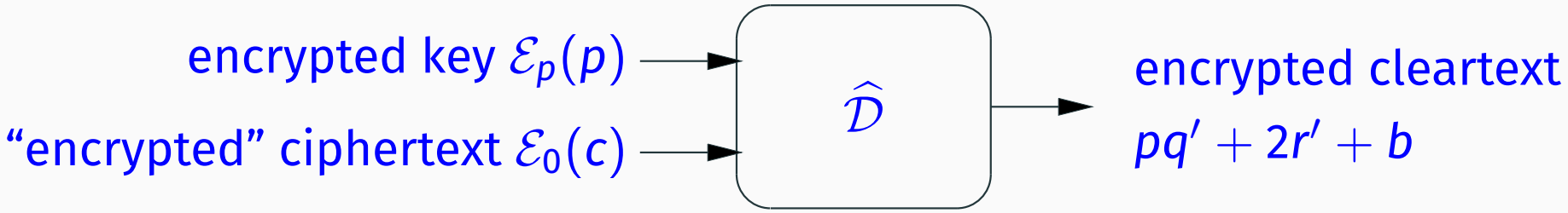
The decryption algorithm $\mathcal{D}$ can be implemented by a circuit.

$$\mathcal{D}_p(c) = LSB(\lfloor c/p \rceil) \oplus LSB(c)$$

Assume that the multiplicative depth of this circuit is small enough to evaluate it using our somewhat homomorphic cipher.

Noise reduction by homomorphic decryption

(Craig Gentry, *A fully homomorphic encryption scheme*, PhD, Stanford, 2009.)



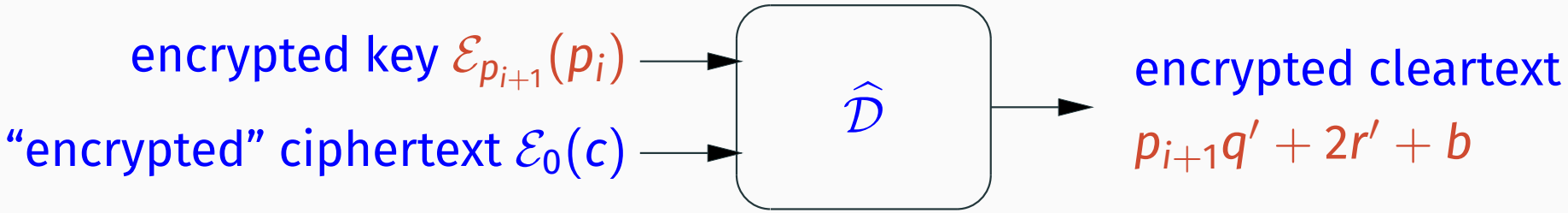
The homomorphic evaluation $\widehat{\mathcal{D}}$ of the circuit $\mathcal{D}$ takes as inputs the encrypted private key and the (trivially) encrypted ciphertext c, and outputs an encryption of the cleartext.

This results in a ciphertext equivalent to c, but whose noise level only depends on the multiplicative depth of the circuit $\mathcal{D}$.

Noise reduction achieved!

Noise reduction by homomorphic decryption

(Craig Gentry, *A fully homomorphic encryption scheme*, PhD, Stanford, 2009.)



To avoid "circular security" (encrypting a key with itself), we can switch keys at bootstrapping time.

We have a sequence of keys $p_1, \ldots, p_i, \ldots$

The evaluator receives the encrypted keys
$\mathcal{E}_{p_2}(p_1), \ldots, \mathcal{E}_{p_{i+1}}(p_i), \ldots$.

Multiplicative depth of the decryption circuit

$$\mathcal{D}_p(c) = LSB(\lfloor c/p \rceil) \oplus LSB(c)$$

Division = multiplication by a quasi-inverse of p:

$$\mathcal{D}_p(c) = LSB(\lfloor cp'/2^N \rceil) \oplus LSB(c) \qquad \text{with} \quad \frac{p'}{2^N} \approx \frac{1}{p}$$

Even then, the multiplication circuit computing cp' is a high-degree polynomial ($\geq$ number of bits of p) and cannot be evaluated by our somewhat homomorphic cipher.

Simplifying decryption

Gentry proposes to “grease” the cipher so that decryption is just one scalar product with low multiplicative depth.

Public key: as before + some fixed-point numbers $y_1, \ldots, y_N$.

Private key: a vector of N bits $\mathbf{s} = (s_1, \ldots, s_n)$, with Hamming weight $\alpha \ll N$, such that $1/p \approx \sum_i s_i \cdot y_i$

Encryption of b: an integer $c = pq + 2r + b$ as before + some fixed-point numbers $z_1, \ldots, z_N$ such that $c/p \approx \sum_i s_i \cdot z_i$

Decryption $LSB(\sum s_i \cdot z_i) \oplus LSB(c)$ can be performed by a circuit with maximal noise $N \cdot \alpha \cdot \mathrm{poly}(\log \alpha)$.

Summary

Summary

In this lecture:

- Several efficient weakly homomorphic ciphers (homomorphic for one operation).
- One inefficient somewhat homomorphic cipher (homomorphic for a limited number of $+$ and $\times$ operations).
- The bootstrap procedure to obtain (laboriously) fully homomorphic encryption from an appropriate somewhat homomorphic cipher.

In the next lecture: how to make all this more efficient and more usable.

References

References

Background on encryption:

- Jean-Philippe Aumasson, *Serious cryptography: A practical introduction to modern encryption*, 2nd ed, No Starch Press, 2024. Chapters 1, 3, 9.

An introduction to FHE by bootstrapping:

- Craig Gentry, *Computing Arbitrary Functions of Encrypted Data*, CACM 53(3), 2010. https://cacm.acm.org/research/computing-arbitrary-functions-of-encrypted-data/