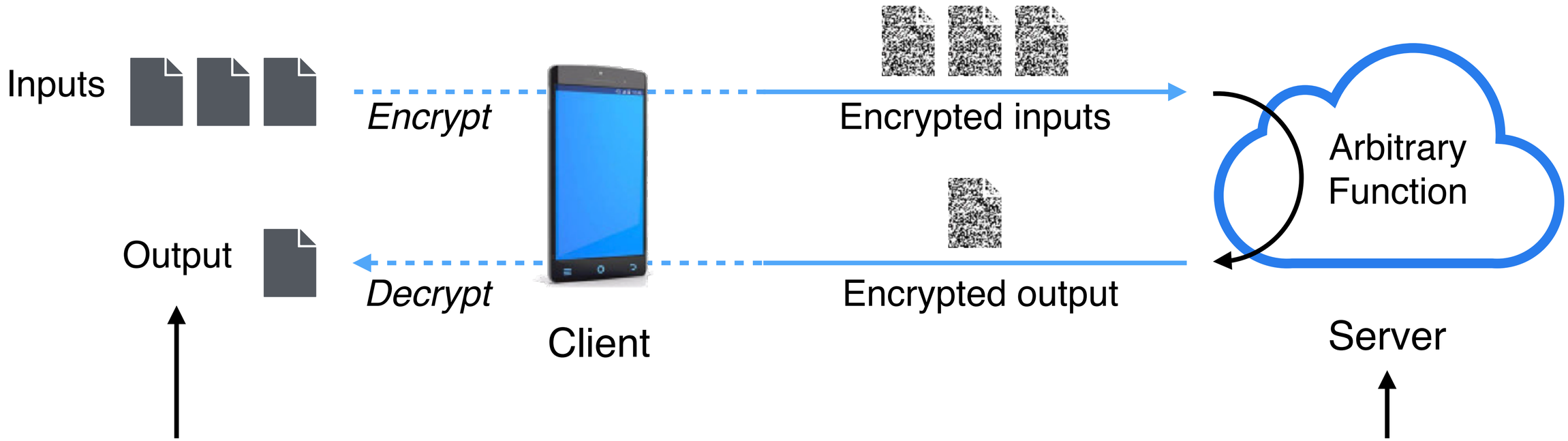
Techniques in Cryptography and Cryptanalysis

Brice Minaud

email: brice.minaud@ens.fr

MPRI, 2025

Fully Homorphic Encryption



Public-Key Encryption Scheme

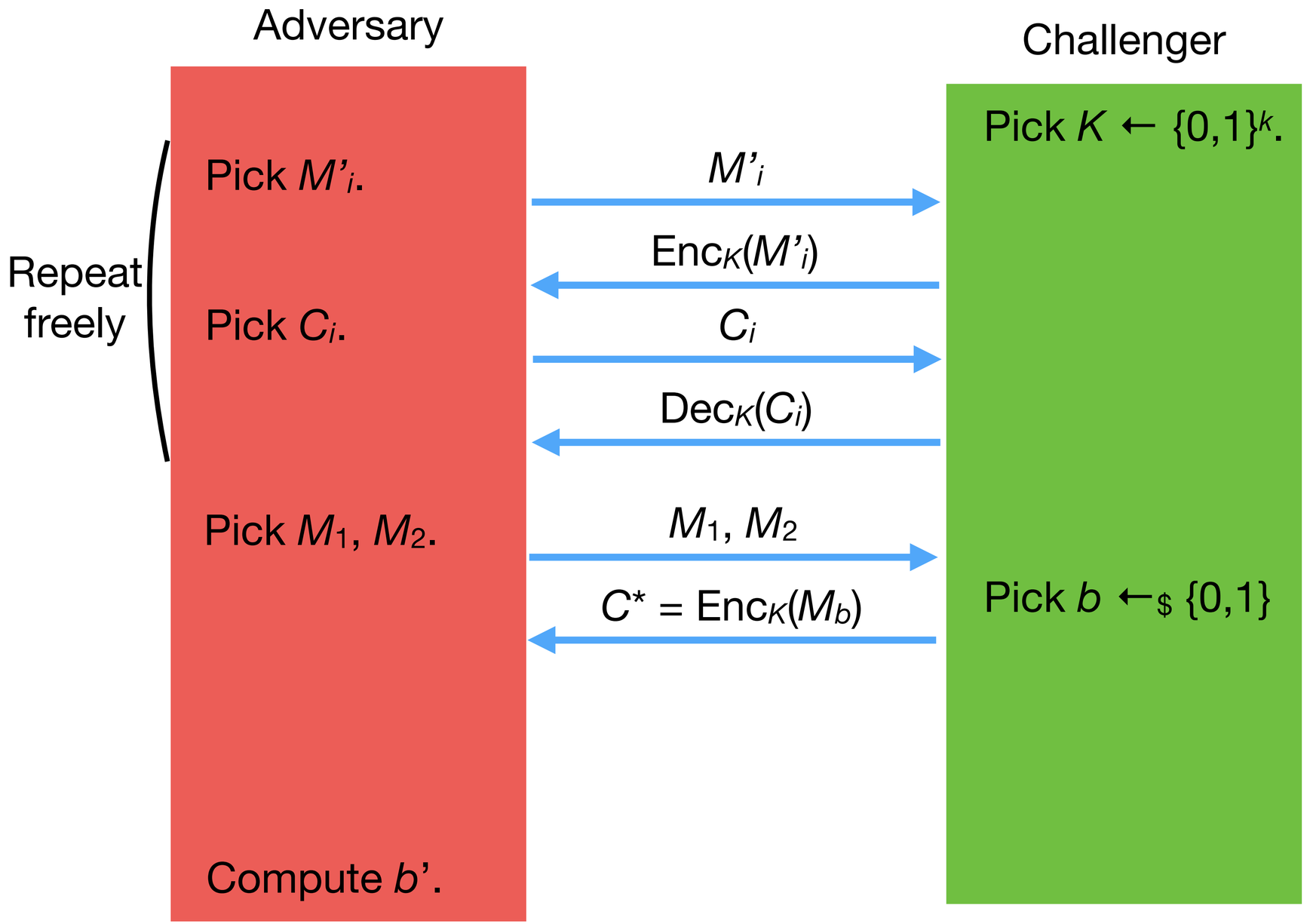
Over:

- Keyspace K.
- Plaintext domain P.
- Ciphertext domain C.

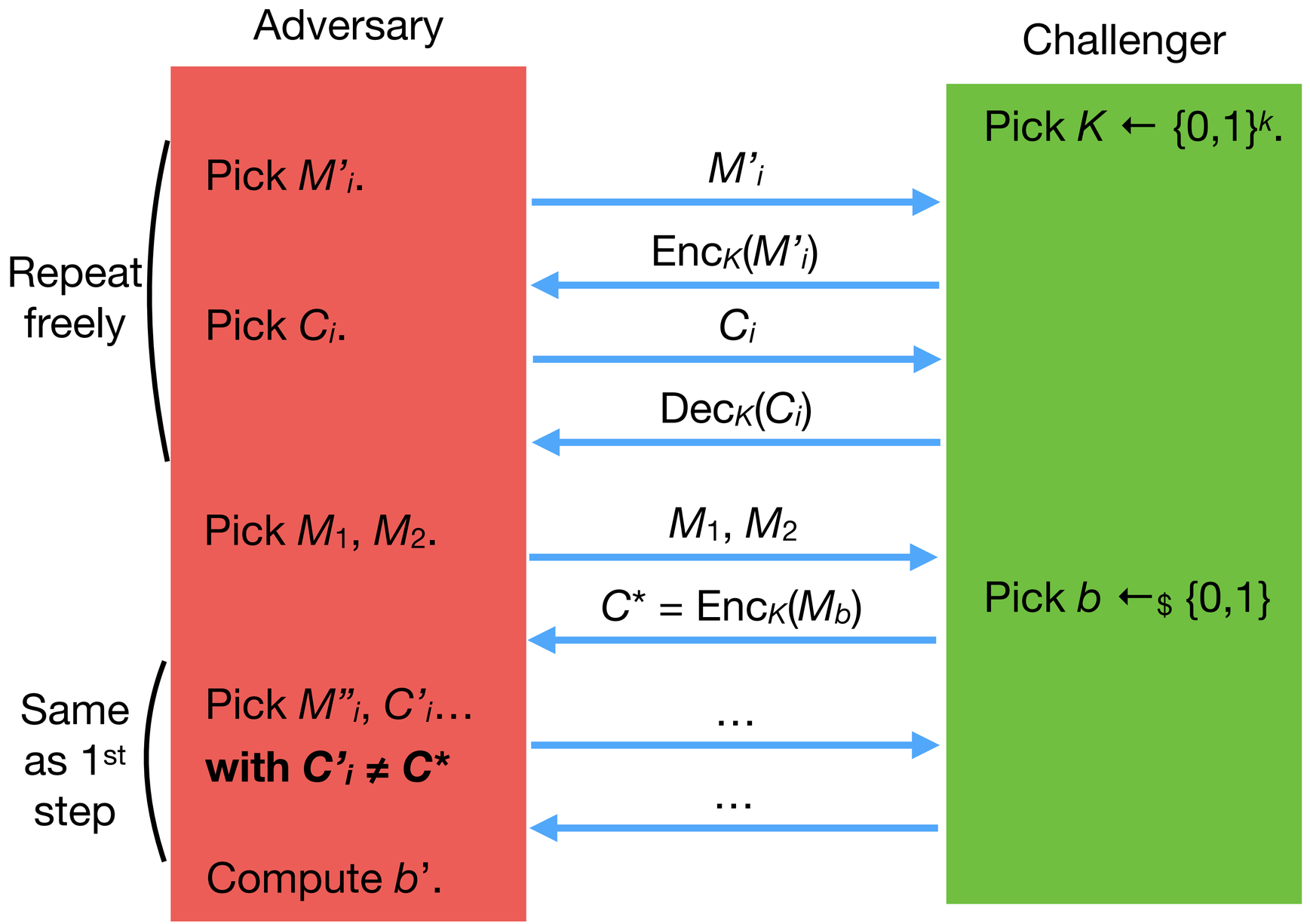
Define a triplet of PPT algorithms:

- **KeyGen**: on input 1^λ, generates an encryption key ek, and a decryption key sk.
- **Enc**: on input $m \in P$ and ek, produce an encryption $c \in C$ of m.
- **Dec**: on input $c \in C$ and sk, produce a decryption $m \in P \cup \{\perp\}$ of c.

IND-CCA1: indistinguishability under Chosen-Ciphertext Attacks



IND-CCA2: indistinguishability under adaptive Chosen-Ciphertext Attacks



Public-key encryption: “proper” definition

Public-Key Encryption.

Defined by triplet of alogrithms (KeyGen, Enc, Dec) over message space **M**, ciphertext space **C**, key space **K**.

- **KeyGen**: on input 1^λ, generates an encryption key ek, and a decryption key sk.
- **Enc**: on input $m \in P$ and ek, produce an encryption $c \in C$ of m.
- **Dec**: on input $c \in C$ and sk, produce a decryption $m \in P \cup \{\perp\}$ of c.

Correctness: for all $M \in$ M,

$$\mathrm{Dec}_K(\mathrm{Enc}_K(M)) = M.$$

Security: IND-CCA (for instance).

Caveat: “security” above only covers **confidentiality**, not **integrity**.

Homomorphic Schemes

Domains

Plaintext domain M, ciphertext domain C and function class F.

Algorithms

KeyGen: on input 1^λ ouputs ek, sk, evk.

- ek may be public.
- sk is private.
- evk is public (e.g. bootstrapping key).

Enc(ek,.): $M \rightarrow C$.

Dec(sk,.): $C \rightarrow M \cup \{\perp\}$.

Eval(evk,.): $F \times C^* \rightarrow C$.

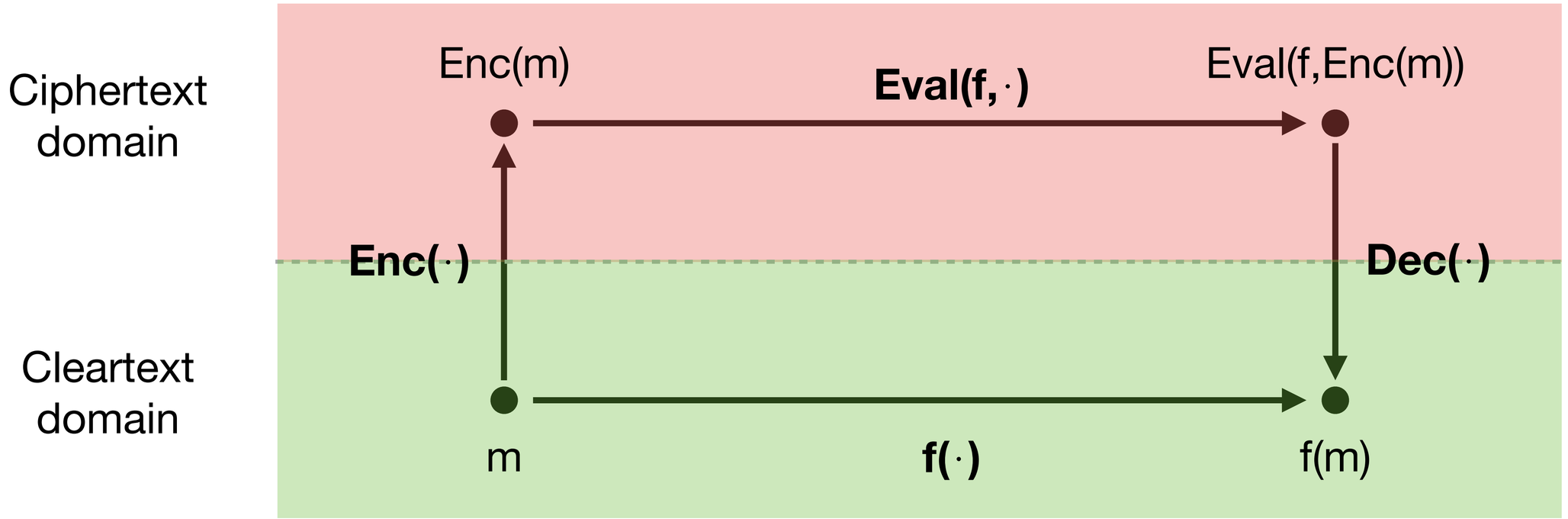
- KeyGen and Enc are PPT.
- Eval and Dec may be deterministic (and often are).

Next few slides adapted from Renaud Sirdey (CEA).

Intuition

For “any” function f, we want something like:

$$\mathrm{Dec}(\mathrm{Eval}(f,\mathrm{Enc}(m_1),\ldots,\mathrm{Enc}(m_k))=f(m_1,\ldots,m_k).$$



Intuition

For “any” function f, we want something like:

$$\mathrm{Dec}(\mathrm{Eval}(f,\mathrm{Enc}(m_1),\ldots,\mathrm{Enc}(m_k))=f(m_1,\ldots,m_k).$$

- We want (at least) CPA security.
  I.e. reduction to a hard problem P.

- We want (at most) polynomial overhead.
  If f is in “O(g(n))” then $\mathrm{Eval}(f,\cdot)$ is in “O(poly(λ)g(n))”.

- And… compactness,
  I.e. size($\mathrm{Eval}(f,\cdot)$) should not depend on size(f).

Questions

Can you imagine a (trivial) non-compact FHE scheme?

Can a deterministic FHE scheme be secure?

Answer(s) to 2nd question

- **If “secure” means “CPA”:** no!

  IND-CPA implies non-determinism, even for normal PKE.

- **If “secure” means something weaker:** still no for FHE!
  - Assume the scheme has plaintext domain Z_t.
  - Get c_0=Enc(0) and c_1=Enc(1).
  - Assume we have c=Enc(x).
  - Eval($\cdot \leq t/2$;c) and returns either c0 or c1.

    Then we can decrypt by dichomotic search…

FHE security

- **CPA** is achieved by all mainstream FHE.

- **CCA1** is achievable.
  - In theory or in restricted cases (open question for research).

- **CCA2** is not achievable.
  - Malleability contradicts CCA2.

- But there are also new notions and new security problems with FHE.

Quick reminder: ElGamal Encryption

$\mathbb{G} = \langle g \rangle$, $|\mathbb{G}| = p$

Secret key: $x \in \mathbb{Z}_p$

Public key: $h = g^x$

Enc$(m) = (g^r, m \cdot h^r)$, where $r \leftarrow_\$ \mathbb{Z}_p$

Dec$(k,c) = c/k^x$

Security reduces to DDH assumption over $\mathbb{G}$.

Multiplicatively homomorphic: given $(k,c) = \text{Enc}(m)$, $(k',c') = \text{Enc}(m')$, (kk',cc') is an encryption of mm'.

“Additively homorphic” ElGamal

ElGamal is **multiplicatively** homomorphic.

To build additively homomorphic scheme, use:

$$\text{MyEncrypt}(m) = \text{ElGamal-encrypt}(g^m)$$

Works because $m \mapsto g^m$ is homomorphism from $(\mathbb{Z}_p,+)$ to $(\mathbb{G}, \cdot)$, and morphisms compose.

Problem: how to define MyDecrypt?

ElGamal-decrypt(MyEncrypt(m)) = g^m

But getting m from g^m = Discrete Logarithm problem...*

*Still doable if m has low entropy, cf. voting protocols.

Making Discrete Logarithm easy

Let $N = pq$ RSA modulus.

Let $\mathbb{G} = \mathbb{Z}_{N^2}$. Let $g = N+1$.

Discrete log over $\mathbb{G}$ in base g: Given $g^x \bmod N^2$, find x.

This is actually easy!

Indeed via the binomial formula (binôme de Newton in French):

$$(N+1)^x = 1 + xN \bmod N^2$$

The Paillier cryptosystem

Let $N = pq$ RSA modulus.

Let $\mathbb{G} = \mathbb{Z}_{N^2}$. Let $g = N+1$.

KeyGen. Public Key: N, Secret Key: $\phi(N)$.

Encryption. Message $m \in \mathbb{Z}_N$, randomness $r \leftarrow_\$ \mathbb{Z}_N$

$$\mathrm{Enc}(m) = g^m r^N \bmod N^2$$

Decryption. Ciphertext $c \in \mathbb{Z}_{N^2}$

$$c^{\phi(N)} = g^{m\phi(N)} r^{N\phi(N)} \bmod N^2$$

$$= g^{m\phi(N)} \bmod N^2$$

$$\Rightarrow \quad \mathrm{Dec}(c) = \mathrm{DL}(c^{\phi(N)})/\phi(N)$$

Security of Paillier cryptosystem

Decisional composite residuosity assumption (DCRA)

Cannot distinguish $r^N \bmod N^2$ where $r \leftarrow_\$ \mathbb{Z}_N$ from uniform element of $\mathbb{Z}_{N^2}$.

Paillier encryption is a “one-time pad” of g^m with r^N

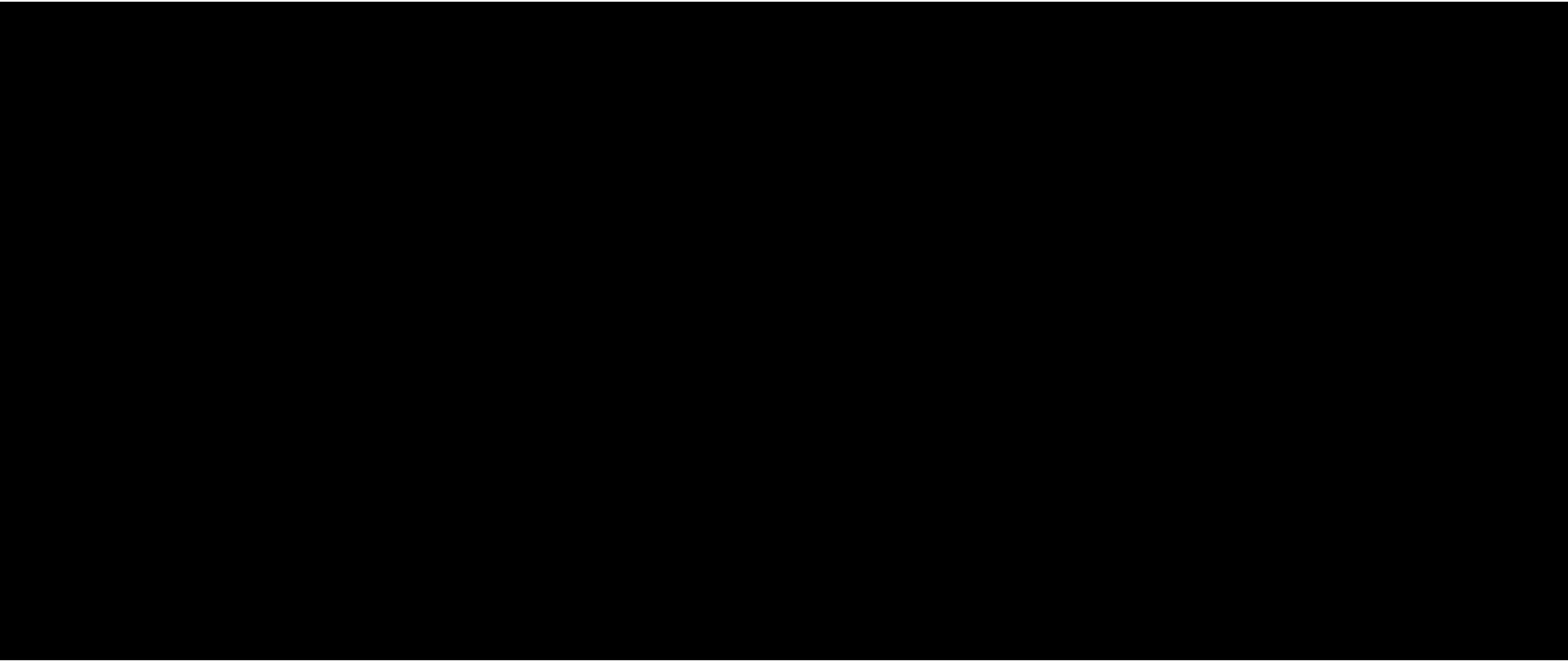
→ CPA security follows immediately from DCRA.

DCRA = distinguish wether an element is an N-th power or not. Easy if factorization of N = pq is known (same as quadratic residuosity).

CCA security: requires non-standard assumptions, or small tweaks.

Question

- What are the homomorphic properties of the scheme?



Can try it here: https://www.linksight.nl/en/content/homomorphic-encryption/

Approximate GCD problem

AGCD problem.

For large prime p and large integer B, distinguish:

- oracle that outputs $x = qp + r$ for uniform $q < B$, and $r \ll p$.
- oracle that outputs uniform $x < pB$.

(For security, set $p \approx 2^{\lambda + \log \lambda}$, $B \approx 2^{\lambda \log \lambda}$, $r \approx 2^{\lambda}$.)

A lattice attack

Let:

$$x = qp + r$$

$$x' = q'p + r'$$

Then:

$$q'x - qx' = q'r - qr' \ll x$$

⇒ Given $x_0, \ldots, x_t$, let:

$$\mathbf{B} = \begin{pmatrix} 2^{\lambda+1} & x_1 & x_2 & \cdots & x_t \\ & -x_0 & & & \\ & & -x_0 & & \\ & & & \ddots & \\ & & & & -x_0 \end{pmatrix}.$$

Then $(q_0,\ldots,q_t)B = (2^{\lambda+1}q_0, q_0x_1 - q_1x_0, \ldots, q_0r_t - q_tr_0)$ is small.

⇒ short vector in the lattice spanned by rows of B.

Adapted from Martin Albrecht, 2020.

Approximate GCD problem

AGCD problem.

For large prime p and large integer B, distinguish:

- oracle that outputs $x = qp + r$ for uniform $q < B$, and $r \ll p$.
- oracle that outputs uniform $x < pB$.

(For security, set $p \approx 2^{\lambda + \log \lambda}$, $B \approx 2^{\lambda \log \lambda}$, $r \approx 2^{\lambda}$.)

Security:

Efficient SVP algorithm ⇒ efficient AGCD algorithm.

Conversely, efficient AGCD algorithm ⇒ efficient algorithm for LWE*.

*[CS16] https://eprint.iacr.org/2016/837

A simple cryptosystem

- **KeyGen**
  A large prime p.

- **Encryption** of $m \in \{0,1\}$:
  Randomly choose a large q and r and let
  $$c := qp+2r+m.$$
- **Decryption**
  $$m := (c \bmod p) \bmod 2.$$

This is CPA-secure assuming the hardness of the AGCD problem.

Questions

- Is decryption of fresh ciphertexts correct?
- What are the homomorphic properties of the scheme?
- Under what condition does decryption remain correct?

Learning with Errors (LWE)

Regev '05. Milestone result.

Pick s uniformly in $\mathbb{Z}_q^n$.

Oracle $O_\$$: returns (a,b) for a uniform in $\mathbb{Z}_q^n$, b uniform in $\mathbb{Z}_q$.
Oracle O_s: returns $(a, a \cdot s + e)$ for a uniform in $\mathbb{Z}_q^n$, e drawn from χ.

Typically, χ is a discrete Gaussian distribution with std deviation αq.

LWE. Let $s \in \mathbb{Z}_q^n$ be drawn uniformly at random. Given access to either $O_\$$ or O_s, distinguish between the two.

LWE (bounded samples). Let $A \in \mathbb{Z}_q^{m \times n}$ and $b, s \in \mathbb{Z}_q^n$ be drawn uniformly at random, and $e \in \mathbb{Z}_q^m$ drawn according to χ.

Distinguish between $(A, As + e)$, and (A, b).

Search and Decision variants

LWE (decisional). Let $s \in \mathbb{Z}_q^n$ be drawn uniformly at random. Given access to either $O_\$$ or O_s, distinguish between the two.

LWE (search). Let $s \in \mathbb{Z}_q^n$ be drawn uniformly at random. Given access to either O_s, find s.

Proposition 1: the two problems are equivalent up to polynomial reductions ("hybrid" technique).

Proposition 2: given an efficient algorithm that solves SIS with parameters n, m, q, β, there is an efficient algorithm that solves LWE with the same parameters, assuming (roughly) $\alpha\beta \ll 1$.

Secret-key encryption using LWE

Pick a secret s uniformly in $\mathbb{Z}_q^n$.

Secret key: s.

Encrypt: to encrypt one bit m: give $(a, a \cdot s + m\lfloor q/2 \rfloor + e)$.

Decrypt: compute $a \cdot s$ to retrieve $m\lfloor q/2 \rfloor + e$, output $m=1$ iff closer to $\lfloor q/2 \rfloor$ than to 0.

IND-CPA security sketch: $(a, a \cdot s + e)$ is indistinguishable from uniform, hence so is $(a, a \cdot s + m\lfloor q/2 \rfloor + e)$.

A public sampler for LWE

To make previous scheme public-key, we'd like a public "sampler" for LWE. Should not require knowing the secret s.

Setup:

- Pick a secret s uniformly in $\mathbb{Z}_q^n$.
- Publish m LWE(q,n,χ) samples for large enough m (value TBD).

That is, publish $(A, As+e)$ for $m \times n$ matrix A.

Now to get a fresh LWE sample:

- Pick x uniformly in $\{0,1\}^n$.
- Publish $({}^t xA,\ {}^t x(As+e))$.

With the right parameters, this yields a distribution statistically close to LWE(q,n,χ'), where if χ is Gaussian with variance σ^2, χ' is Gaussian with variance $m\sigma^2$.

Argument: Leftover Hash Lemma. Example: $m = 2n \log q$ suffices.
Remark: recognize the Ajtai hash function from SIS.

Public-key* Regev encryption

Regev ’05: **Regev encryption.**
Idea: same as secret-key scheme, but with public sampler.

Pick a secret s uniformly in $\mathbb{Z}_q^n$, A uniformly in $\mathbb{Z}_q^{m \times n}$.

Public key: $(A, b = As + e)$.

Secret key: s.

Encrypt: to encrypt one bit m: draw x in $\{0,1\}^m$, output:

$$({}^t xA, {}^t xb + m \lfloor q/2 \rfloor).$$

Decrypt: upon receipt of ciphertext (c,d), output 0 if $d - c \cdot s$ is closer to 0 than to $\lfloor q/2 \rfloor$, 1 otherwise.

Proof argument. Step 1: public key is indistinguishable from uniform. Step 2: assuming uniform public key, ciphertexts are statistically close to uniform.

Can be seen as generic SK→PK transform w/ homomorphism!

*malleability → not IND-CCA.

Questions

- Is decryption of fresh ciphertexts correct?
- What are the homomorphic properties of the scheme?
- Under what condition does decryption remain correct?

Magic trick 1: multiplicative homomorphism

An encryption of one bit m is essentially (up to leftover hash lemma):

$$\mathrm{Enc}(m) = (a,\ a \cdot s + m\lfloor q/2 \rfloor + e) = (a,b).$$

Let $c = (a,b)$ be the ciphertext, and $s' = (-s,1)$, decryption is essentially:

$$c \cdot s' = m\lfloor q/2 \rfloor + e$$

Given two ciphertexts c and c', define:*

$$\mathrm{mult}(c,c') = \lfloor 2/q\ c \otimes c' \rceil$$

To decrypt, compute:

$$\begin{aligned}\mathrm{mult}(c,c') \cdot (s' \otimes s') &\approx 2/q\ (c \otimes c') \cdot (s' \otimes s') \\ &= 2/q\ (c \cdot s')(c' \cdot s') \\ &\approx q/2\ mm' + me' + m'e + 2/q\ ee' \\ &= q/2\ mm' + e''.\end{aligned}$$

*Notation: $(x_1,\ldots,x_m) \otimes (y_1,\ldots,y_n) = (x_1y_1,\ldots,x_1y_n,\ x_2y_1,\ldots,x_2y_n,\ \ldots\ \ldots\ \ldots,\ x_my_1,\ldots,x_my_n)$.

Full homomorphism?

Additive homomorphism:

$$\mathrm{add}(c,c') = c + c'$$

Decrypted with s':

$$\mathrm{add}(c,c')\cdot s' = (m+m')\, q/2 + \text{noise}.$$

Multiplicative homomorphism:

$$\mathrm{mult}(c,c') = \lfloor 2/q\ c \otimes c' \rceil$$

Decrypted with $s'' = s' \otimes s'$:

$$\mathrm{mult}(c,c')\cdot s'' = mm'\, q/2 + \text{noise}.$$

But ciphertext size increases quadratically, cannot really go on...

Magic trick 1 (part 2): relinearization via key switching

Sample fresh secret key t.

"Encrypt" every $s_i s_j$ with t:
let (a_{ij}, b_{ij}) be such that

$$b_{ij} = a_{ij} \cdot t + s_i s_j + \text{noise} \approx a_{ij} \cdot t + s_i s_j$$
$$\Rightarrow s_i s_j \approx b_{ij} - a_{ij} \cdot t$$

Idea: to decrypt mult(c,c'), use $b_{ij} - a_{ij} \cdot t$ in place of $s_i s_j$:

$$\begin{aligned}
\text{mult}(c,c') \cdot (s' \otimes s') &= \text{mult}(c,c') \cdot (s_i s_j)_{1 \le i,j \le n} \\
&\approx \text{mult}(c,c') \cdot (b_{ij} - a_{ij} \cdot t)_{1 \le i,j \le n} \\
&= \text{mult}(c,c') \cdot (b - At) \\
&= \text{mult}(c,c') \cdot ((b|A)(1,-t)) \\
&= ((b|A)^T \text{mult}(c,c')) \cdot (1,-t)
\end{aligned}$$

where b=vector with entries (b_{ij}),
A=matrix with rows (a_{ij})

$\Rightarrow$ redefine mult'(c,c') = $(b|A)^T$mult(c,c').

$\Rightarrow$ this gives encryption of mm' under key $t' = (1,-t)$.

Are we fully homomorphic yet?

Encrypt $(s_i s_j)_{1 \le i,j \le n}$ with secret $t = s$. Add $(b|A)$ to public key.

Additive homomorphism:

$$\mathrm{add}(c,c') = c + c'$$

Multiplicative homomorphism:

$$\mathrm{mult}(c,c') = 2/q \; (b|A)^T (c \otimes c')$$

Size of ciphertext is unchanged, everything decrypts under s.

But the noise increases with every operation.

Correctness is lost if the noise exceeds q/2.

Somewhat Homomorphic Encryption

Right now: only limitation is noise increase.

What we have is:

Somewhat Homomorphic Encryption.

∀ **circuit C,** ∃ **FHE scheme** powerful enough to compute C.

How?

Sketch: for given C, can upper-bound max noise at the output.
→ Pick initial noise of fresh ciphertexts small enough that C is guaranteed to compute correctly.

The deeper C is (especially multiplicative depth), the smaller the noise-to-modulus ration, & the more expensive every FHE operation...

Fully Homomorphic Encryption

Somewhat Homomorphic Encryption (SWHE).

∀ **circuit C,** ∃ **FHE scheme** that can compute C.

The dream:

Fully Homomorphic Encryption (FHE).

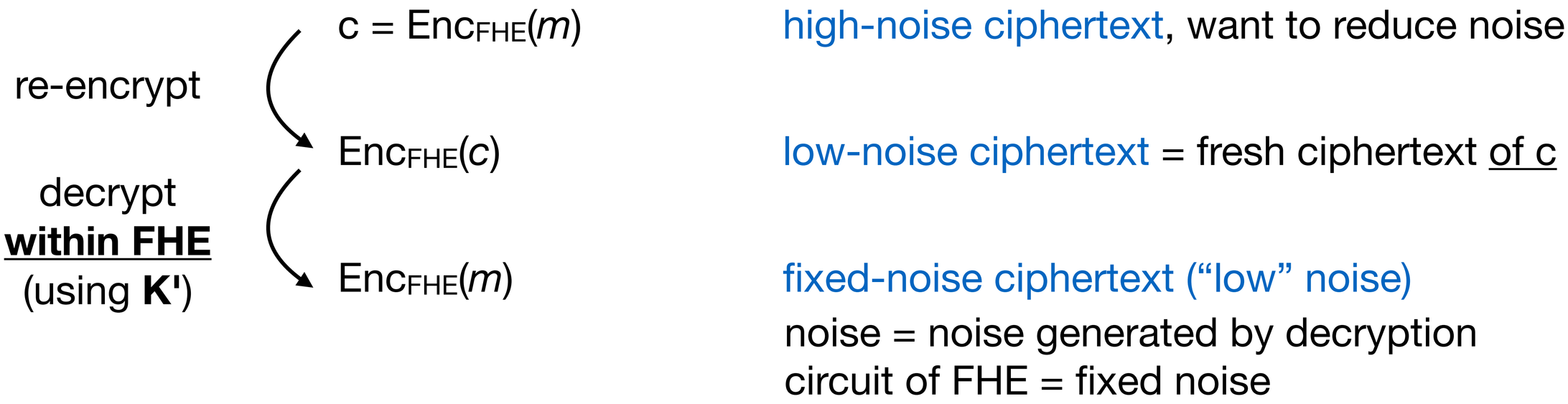
∃ **FHE scheme,** ∀ **circuit C,** FHE can compute C.

→ Fixed cost for FHE regardless of C.
→ Unbounded computation, without need to know C in advance.

“Bootstrapping” SWHE into FHE

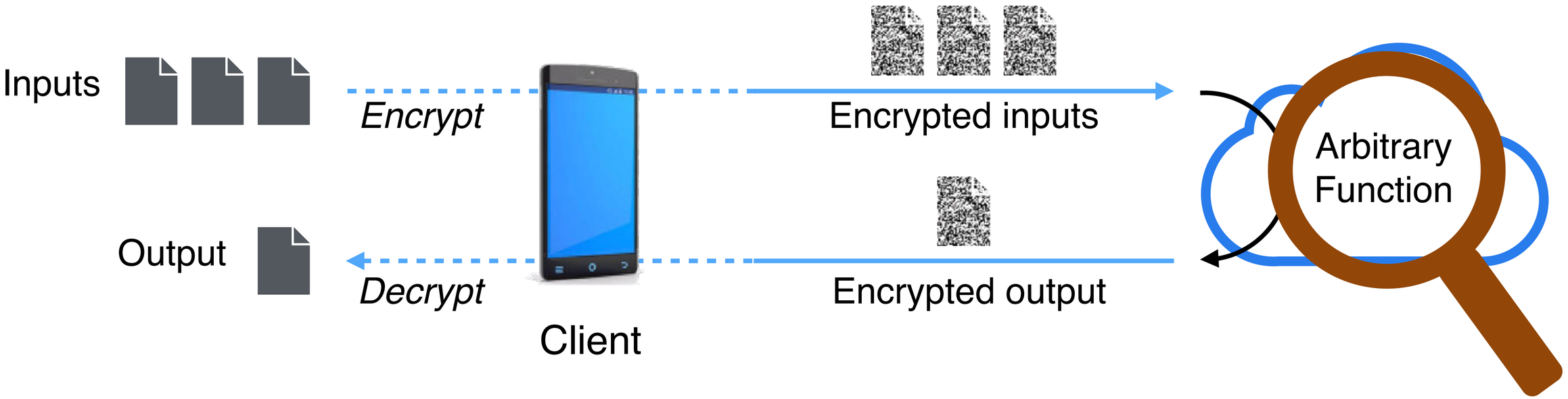
Groundbreaking idea by Craig Gentry (2009).

Add to public key: **$K' = Enc_{FHE}(K_{FHE})$** [“circular security”]



SWFHE that can correctly compute its own decryption circuit
+ at least 1 operation ⇒ FHE.

Going back to the computation model



Models of computation

Algorithms

```c
#include <stdio.h>

unsigned long fib(unsigned short n)
{
        if (n <= 1)
                return 1;
        else
                return fib(n-1) + fib(n-2);
}

int main()
{
        printf("result: %lu\n",fib(5));

        return 0;
}
```

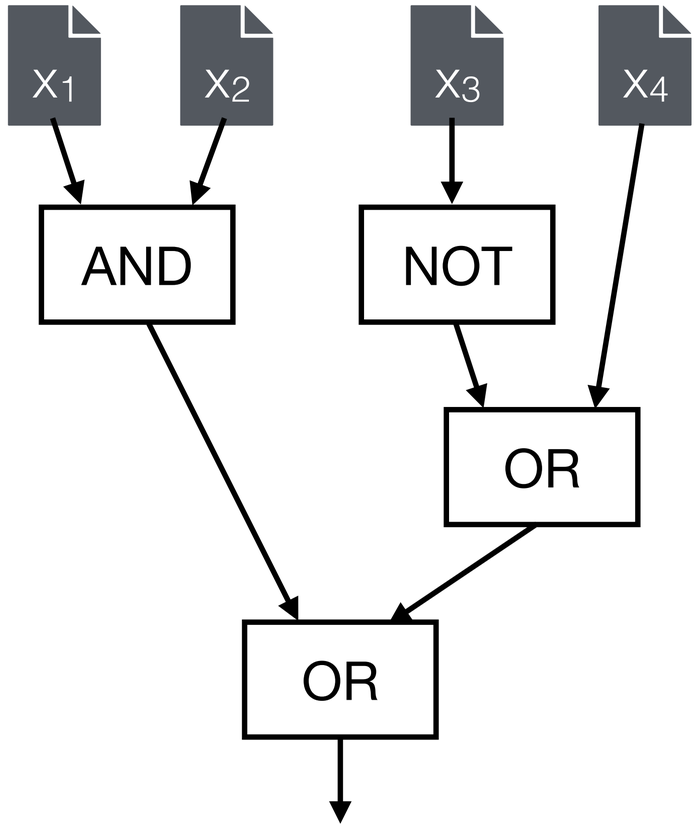
In theory: Turing machines.

Usually **RAM** machines.

ORAM

Conversion

Circuits



Pure circuits: no notion of **memory**.

FHE

ORAM vs FHE/MPC/ZK

Algorithms (with RAM)

- ORAM

- FHE with RAM function: https://eprint.iacr.org/2022/1703

- Conversely, ORAM w/ O(1) communication overhead via FHE: https://eprint.iacr.org/2019/736

Circuits

- Fully Homorphic Encryption*
- Multi-Party Computation*
- zk-SNARKS
- Neural networks

No notion of “memory”.*

*except when combined with ORAM (in a non-trivial way).

The “strange” FHE computer

Some basic implications of computing in the circuit model...

- **No proper “if”/no proper branching.**
- **No notion of memory.**

  Everything potentially computed on is a circuit input. No notion of pointer.
- **Always performs worst-case complexity.**

See e.g. https://github.com/CEA-LIST/Cingulata

Main FHE flavors

- BFV, BGV:
  - Large plaintext domain.
  - Heavy SIMD //-ism => competitive amortized performances.
  - Some support for non linear ops (beyond polynomial approx.).
  - No efficient bootstrapping.
  - Multiplicative depth dependency.
  - Multikey and threshold variants.
- CKKS:
  - Approximate computations (no message scaling).
  - Large plaintext domain.
  - Heavy SIMD //-ism => competitive amortized performances.
  - No support for non linear ops (beyond poly. Approx).
  - No efficient bootstrapping.
  - Multiplicative depth dependency.
  - Weaker than BFV or BGV with respect to passive attackers.
  - Multikey and threshold variants.
- TFHE (aka CGGI):
  - Efficient bootstrapping.
  - Functionnal bootstrapping => easy non linear ops.
  - Multiplicative-depth independance.
  - Small plaintext domain (32 values max).
  - No batching.
  - Multikey and threshold variants are WIP.

HEAAN
CRYPTO LAB

ZAMA

Slide content by Renaud Sirdey (CEA). Some sample performance for TFHE on ML tasks:
https://www.zama.ai/post/making-fhe-faster-for-ml-beating-our-previous-paper-benchmarks-with-concrete-ml

More magic tricks

- **Hide *to the server*** what circuit is computed.

  Use universal circuit. Then encode desired circuit as data (FHE ciphertexts).

- **Hide *to the client*** what circuit was computed.

  Simplest way: noise flooding.

- **Reduce client → server ciphertext expansion to nothing!**

  Use *transciphering*: at setup, server receives $K = \mathrm{Enc}_{\mathrm{FHE}}(\text{AES-key})$.*

  Then client can send all ciphertexts in the form c = AES-encrypt(m).

  Server computes $\mathrm{Enc}_{\mathrm{FHE}}(c)$, then decrypts *within* FHE using K.

  ⇒ Server gets $\mathrm{Enc}_{\mathrm{FHE}}(m)$.

*Best realized with specialized symmetric encryption schemes (not AES).