CISPA
HELMHOLTZ CENTER FOR INFORMATION SECURITY

Homomorphic Encryption

CS523 2025

Sylvain Chatel | March 04, 2025 | v1.0.2

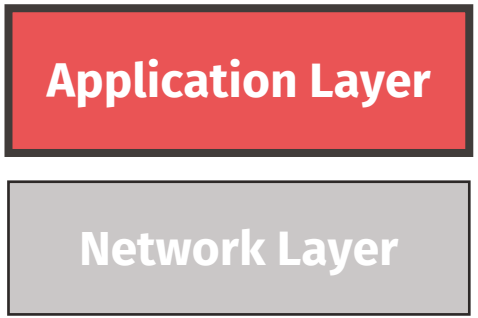
Some slides/ideas adapted from C. Mouchet, D. Fiore, JP Hubaux, C. Troncoso, and W. Lueks

Introduction
Homomorphic encryption

Lecture aim: study the cryptographic technique and related **toolbox for privacy engineering**

tool
for building PETS

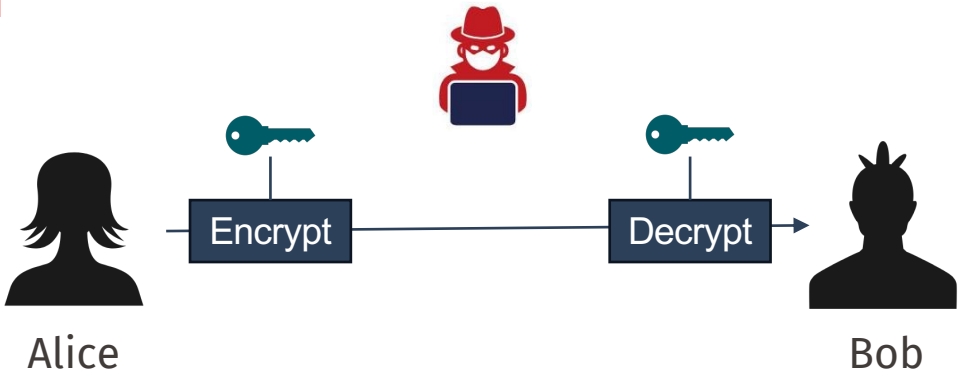
cryptographic
primitive



Goals
What should you learn today?

- Basic understanding of **homomorphic encryption**
- Understand **when to use** homomorphic encryption
- Understand **key properties:**
  - Communication and computation cost
  - Trust assumptions
  - Guarantees with respect to inputs
- Understand **practical issues** when using homomorphic encryption

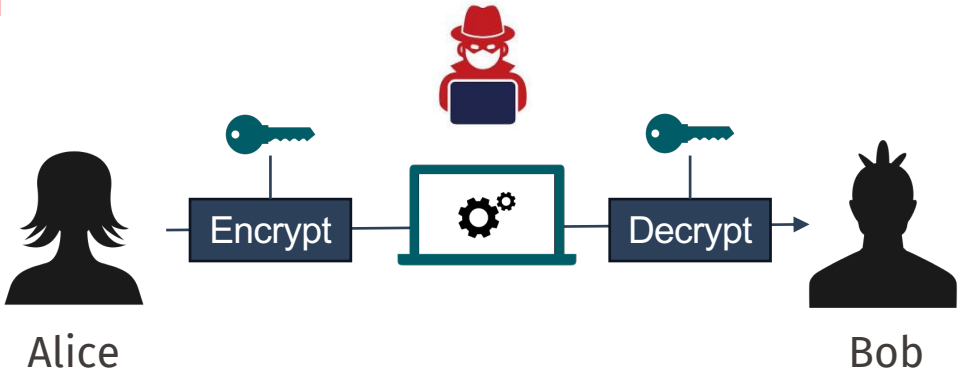
Introduction
Encryption



Encryption aims at data confidentiality

- In transit
- In storage

Introduction
Encryption

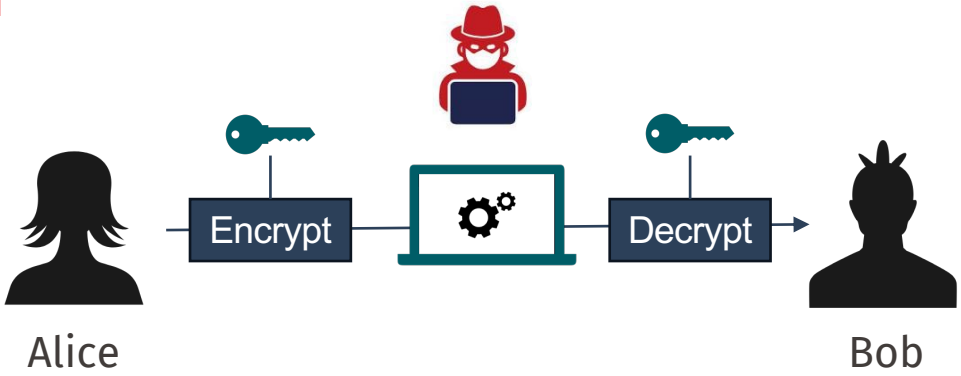


Encryption aims at data confidentiality

- In transit
- In storage

Homomorphic Encryption aims at also confidentiality in **computation**

Introduction
Encryption



Encryption aims at data confidentiality

In transit

In storage

Homomorphic Encryption aims at also confidentiality in **computation**

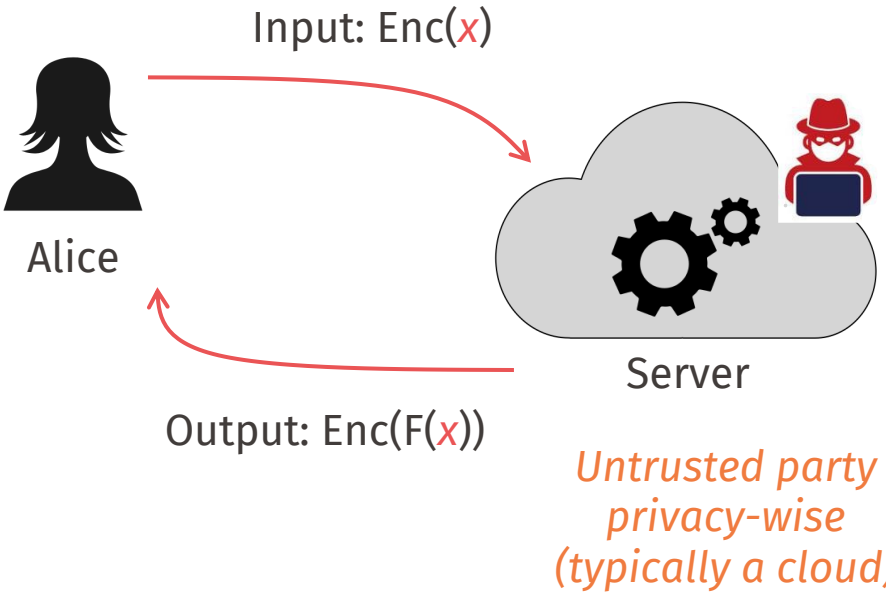
Health Data Analysis

Financial Fraud Prevention

Govt Data: e.g., demographics, vote, etc.

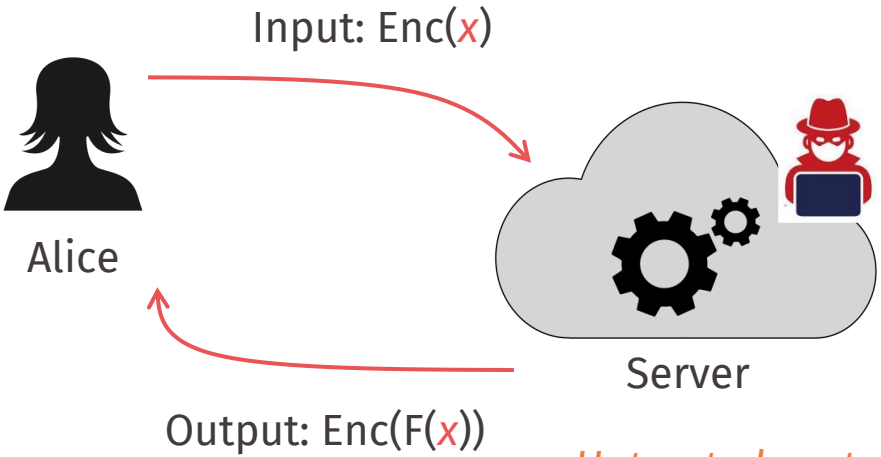
Overview
Homomorphic encryption

- A cryptographic primitive that enables the computation of functions in the encrypted domain



Overview
Homomorphic encryption

- A cryptographic primitive that enables the computation of functions in the encrypted domain



Untrusted party
privacy-wise
(typically a cloud)

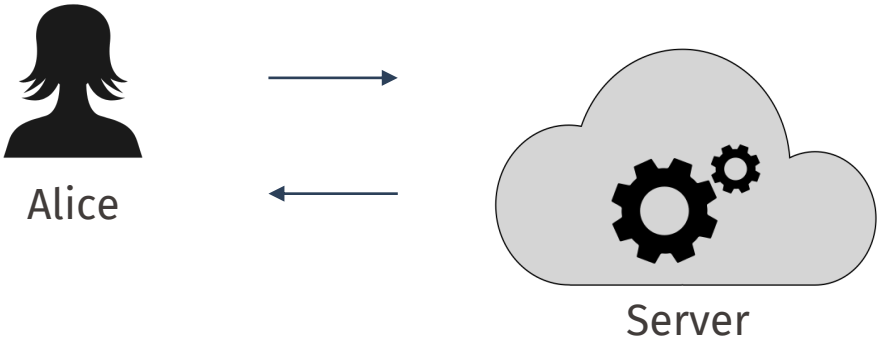
- **Security property**: the computing party cannot learn any information about the input or output
- **Correctness**: output is correct

- **Threat Model**: usually Honest-but-Curious

Overview
Homomorphic Encryption System Model

Homomorphic encryption: classically, has one computing party, and

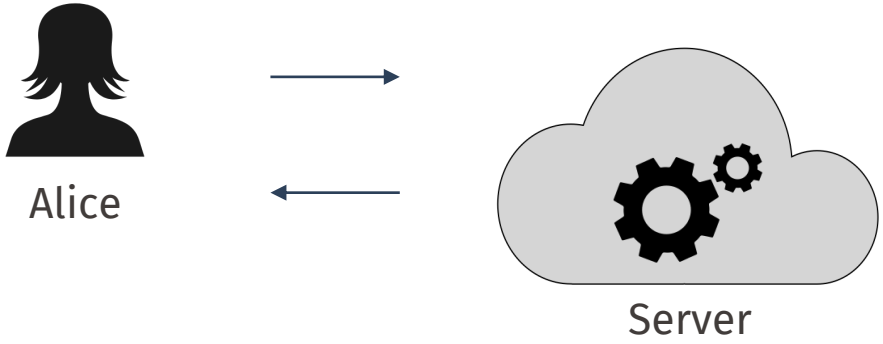
- **one** party providing the input and reading the result, or

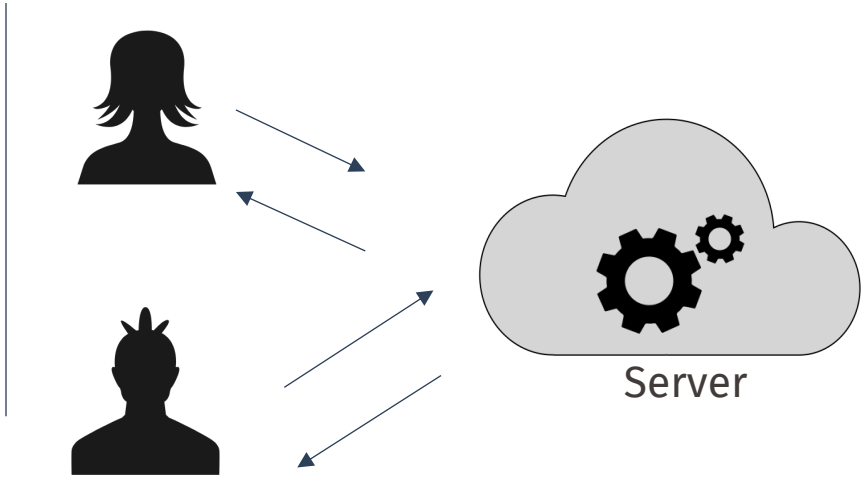


Overview
Homomorphic Encryption System Model

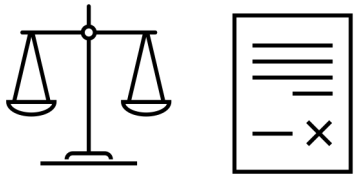
Homomorphic encryption: classically, has one computing party, and

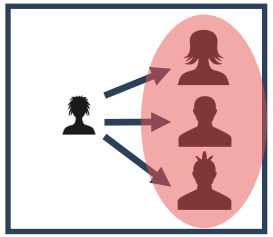
- **one** party providing the input and reading the result, or
- ***n* parties** providing the input and **another** reading the result, or
- ***n* parties** providing data and learning the result (not in this class)

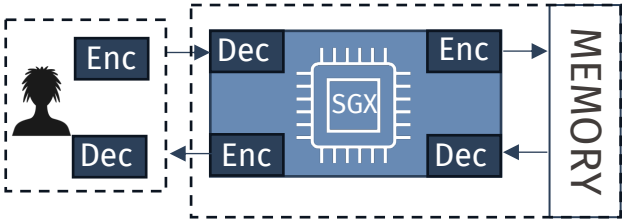




What about Alternatives?
NDAs, SMC, TEE







Non-Disclosure Agreements

Low tech

Lengthy Process

Limited protection

Secret Sharing-SMC

Non-collusion assumption

If they do, they can join their shares and recover all the secrets!

Hard to find in practice

Trusted Execution Env.

Trust in manufacturer

Can be vulnerable to physical attacks

Homomorphic Encryption
Objectives Consolidated

Enable **computation** on **encrypted data**

- Computation performed before decryption
- Decryption key known only to the receiver

Outsourced computation setting: party with data provides an encrypted version of the data to a *single* untrusted computation party (*a server*). No need for a non-collusion assumption.

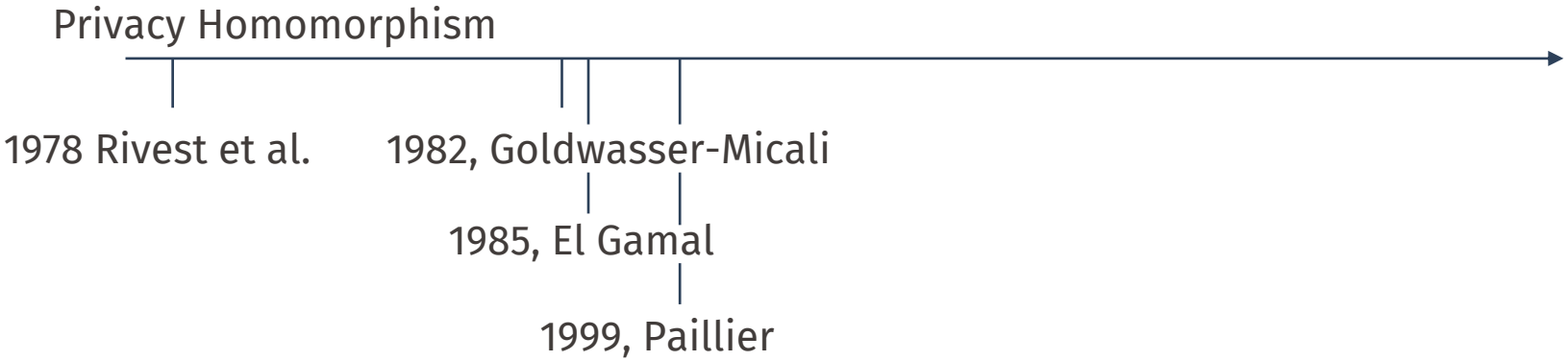
Rely on **solely on the security of the cryptographic primitives**

Homomorphic Encryption
History

Privacy Homomorphism

1978 Rivest et al.

Homomorphic Encryption
History



- RSA 1977: modular multiplication
- Goldwasser-Micali: XOR
- El Gamal: modular multiplication
- Paillier: modular addition

Homomorphic Encryption
History

Privacy Homomorphism

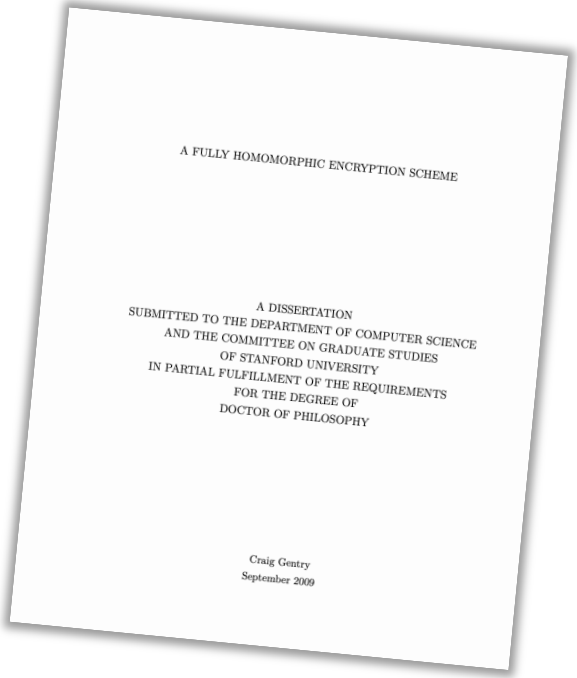
1978 Rivest et al.

1982, Goldwasser-Micali

1985, El Gamal

1999, Paillier

2009, Gentry

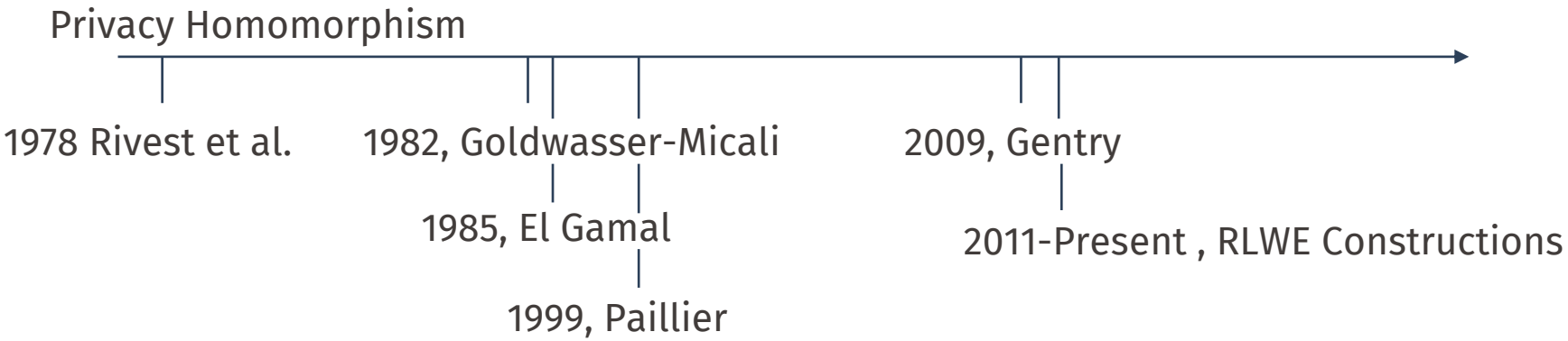


A FULLY HOMOMORPHIC ENCRYPTION SCHEME

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

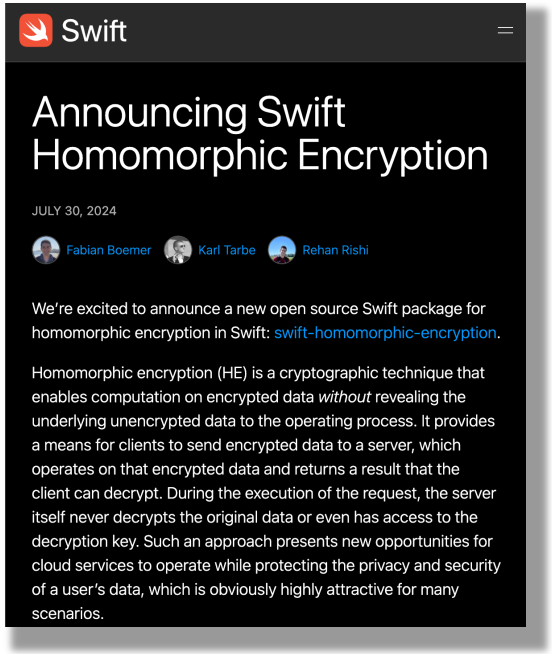
Craig Gentry
September 2009

Homomorphic Encryption
History



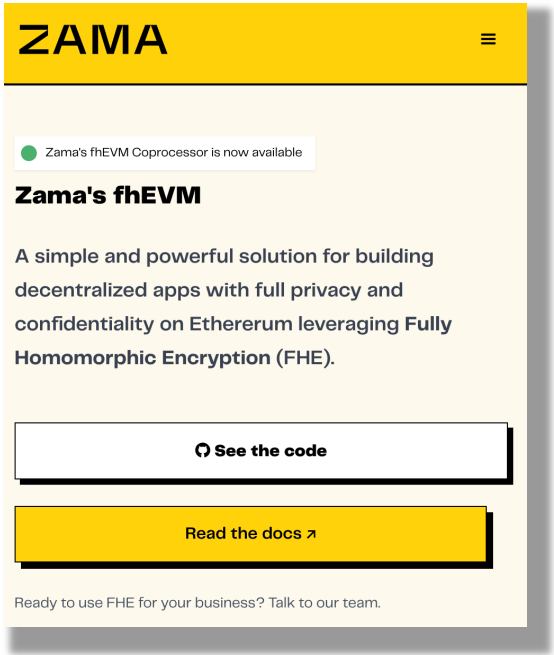
- Lattice-based schemes (e.g., BGV, BFV, CGGI, CKKS)
- Many libraries (e.g., Helib, SEAL, HEAAN, PALISADE, **OpenFHE, Lattigo, TFHE-rs**.
- Standardization 2017-soon

Homomorphic Encryption Applications



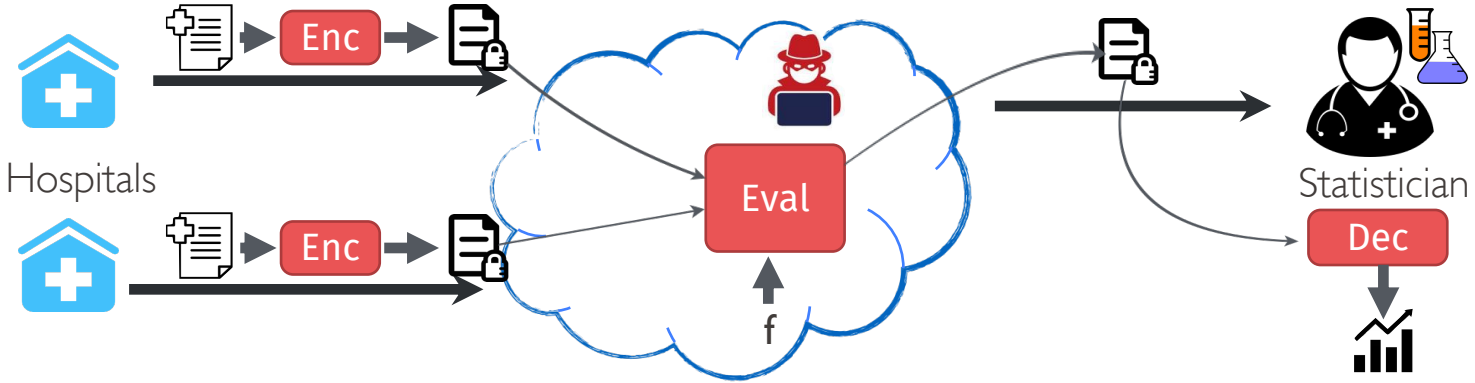
Apple Live Caller ID - iOS18

Tune Insight Federated Learning



Zama confidentiality for Etherium

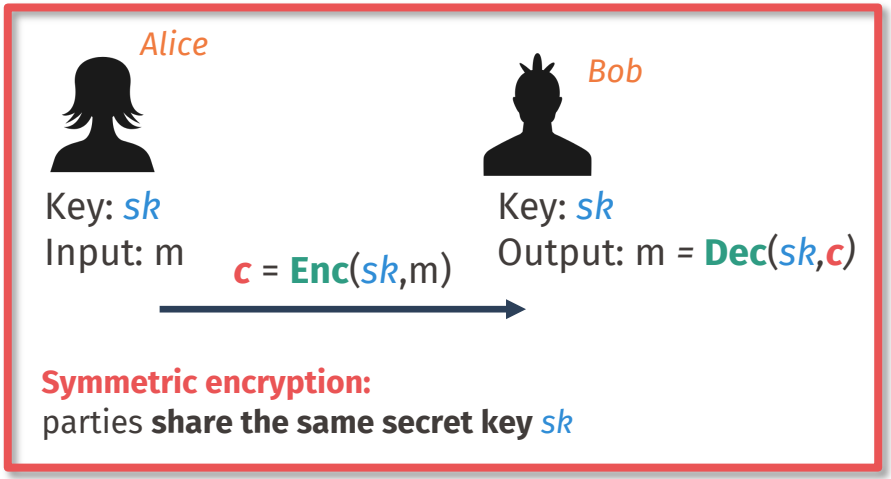
Example application
Privacy-preserving statistics on medical data

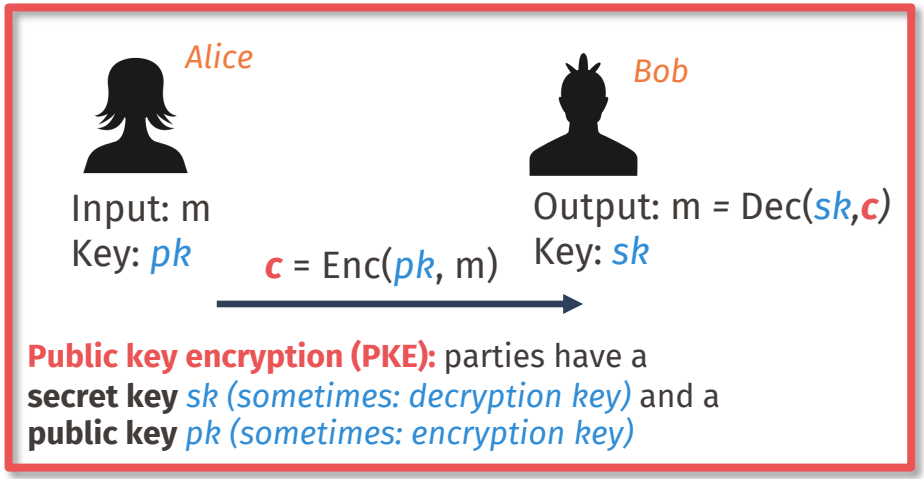


- **Goal:** a statistician wants to compute statistics on joint data from several hospitals
- **Privacy concern:** single data entries are privacy-sensitive patients' data
- **Using a cloud server and FHE**
  - Hospitals store encrypted patients' data on the cloud
  - Cloud compute statistics on joint (encrypted) datasets. Statistician decrypts results

Definitions and Properties

Formal definition
Cryptographic Notation





KeyGen$(1^k) \rightarrow (sk, pk)$ — *Generates a private/public key pair for a security parameter k*

Enc$(pk, m; r) \rightarrow \mathbf{c}$ — *Encrypt the message m with randomness r to a ciphertext c*

Dec$(sk, \mathbf{c}) \rightarrow m$ — *Decrypt a ciphertext m to obtain the message m*

Formal definition
Homomorphic encryption

A **homomorphic encryption scheme** is given by the following four algorithms:

- **KeyGen**$(1^k) \rightarrow (sk, pk)$ — *Generates a private/public key pair for a security parameter* k
- **Enc**$(pk, m; r) \rightarrow \boldsymbol{c}$ — *Encrypt the message m with randomness r to a ciphertext* **c**
- **Dec**$(sk, \boldsymbol{c}) \rightarrow m$ — *Decrypt a ciphertext* **c** *to obtain the message m*
- **Eval**$(pk, f, \boldsymbol{c_1},\ldots,\boldsymbol{c_n}) \rightarrow \boldsymbol{c}'$ — *Evaluate function f on the encrypted input* $\boldsymbol{c_i}$ *to obtain a new ciphertext* **c'**

***Eval**() is what makes HE different from standard PK encryption*

*(in some schemes **Eval** uses a special key evk instead of pk. If this is the case, then **KeyGen** generates a triplet)*

*(The **KeyGen** and **Enc** are randomized algorithms. The randomness parameter is often implicit)*

Properties
Homomorphic encryption

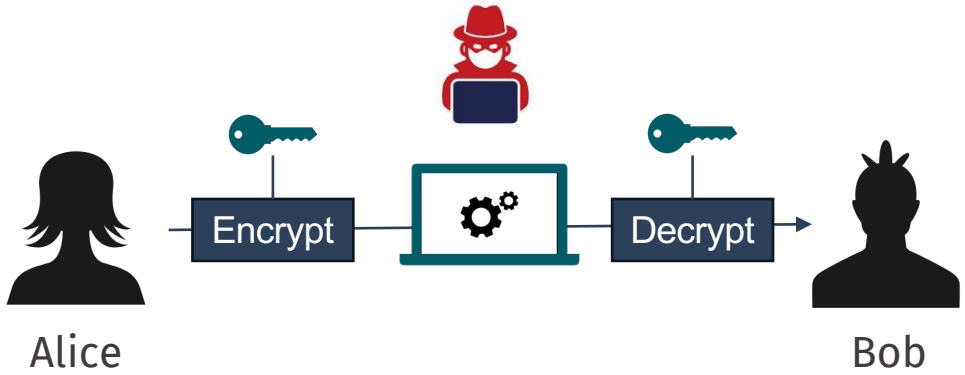
Correctness

Security

Evaluation correctness

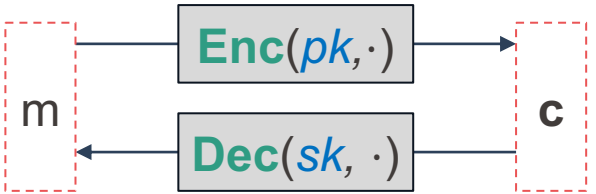
Composition

Compactness



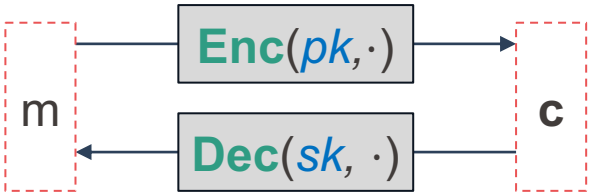
Homomorphic Encryption
Encryption Correctness

Correctness – *intuition:*



Homomorphic Encryption
Encryption Correctness

Correctness – *intuition:* applying the deterministic **decryption** function to the randomized **encryption** always **returns the initial plaintext**.



Correctness.

$\forall$ r$\in$ Rand, $\forall$ m $\in \mathcal{P}$ and (*sk*, *pk*)$\leftarrow$**KeyGen**(1^k):

Dec(*sk*, **Enc**(*pk*, *m*; r)) = m

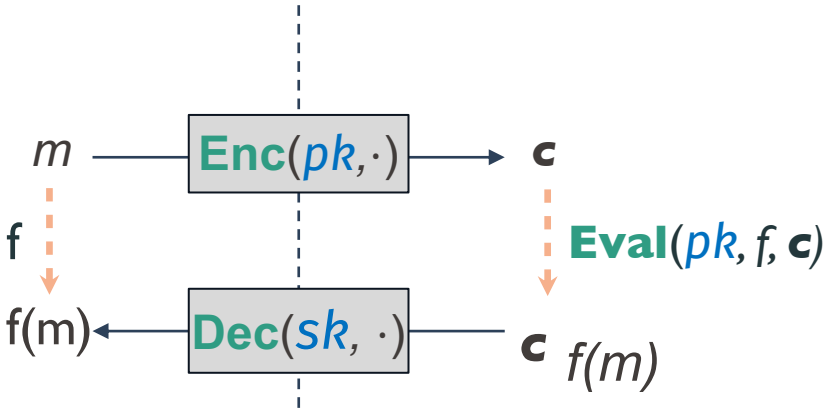
__Dec__(sk, ·) is the deterministic inverse of the randomized __Enc__(pk, ·) function

$\mathcal{P}$ is the plaintext space and Rand a source of randomness

Homomorphic Encryption
Homomorphic Correctness

Objective: Evaluate the plaintext function f over the encryption in the ciphertext space

Recall an HE scheme:
KeyGen$(1^k) \rightarrow (sk, pk)$
Enc$(pk, m; r) \rightarrow \boldsymbol{c}$
Dec$(sk, \boldsymbol{c}) \rightarrow m$
Eval$(pk, f, \boldsymbol{c}) \rightarrow \boldsymbol{c}_{f()}$



Homomorphic Encryption
Wait... Homomorphic?

Homomorphism: mapping between two sets that preserves their algebraic structure

Group homomorphism
Given two groups (G, +), (H, ⊠)

h: G →H is a *group-homomorphism* if ∀x, y ∈ G:

$$h(x + y) = h(x) \boxtimes h(y)$$

For an **additive** HE scheme, **Dec**(*sk*, ·): $\mathcal{C} \rightarrow \mathcal{P}$ is an homomorphism between

($\mathcal{C}$, **Eval.**Add) and ($\mathcal{P}$, +)

*A **group** consists of a **set of elements** and **one operation** that combines two elements of the set into another element of the set (operation must be associative, there is an identity element, and all elements have inverse)*
*A **ring** consists of a set of elements and **two operations** that combine two elements of the set into another element of the set (multiplication does not need to be commutative, and does not need inverse)*

Homomorphic Encryption
Homomorphic Correctness

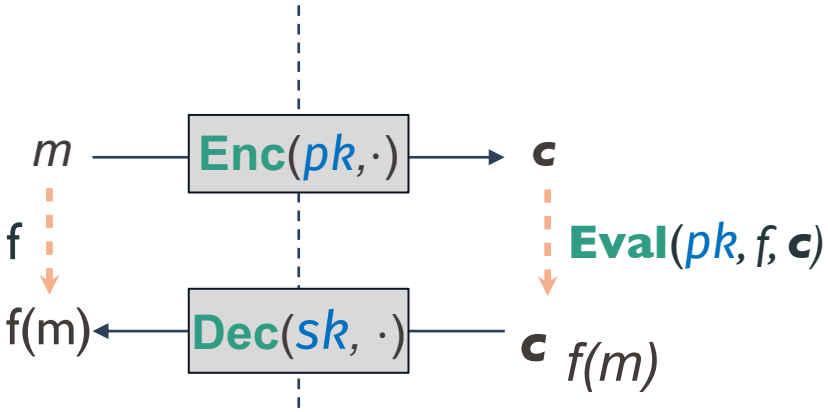
Objective: Evaluate the plaintext function f over the encryption in the ciphertext space

For a set $\mathcal{F}$ of admissible functions

$\forall m \in \mathcal{P}$, $\forall f \in \mathcal{F}$, and

$(sk, pk) \leftarrow \mathbf{KeyGen}(1^k)$:

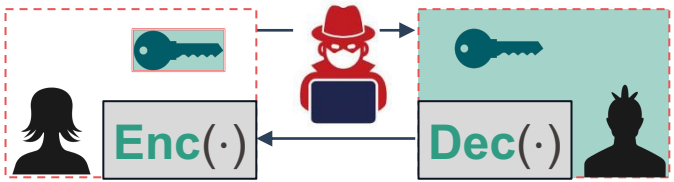
$\mathbf{Dec}(sk, \mathbf{Eval}(pk, f, \mathbf{Enc}(pk, m))) = f(m)$



This definition is relaxed in practice

Homomorphic Encryption
Security

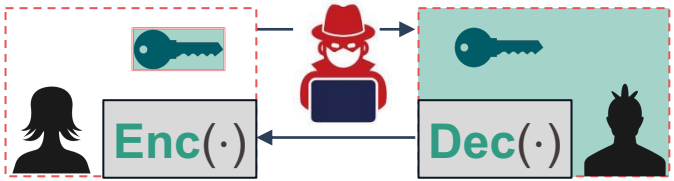
Security – *intuition:*



Homomorphic Encryption
Security

Security – *intuition:* The adversary learns nothing from the ciphertexts and public keys

How do we capture this?

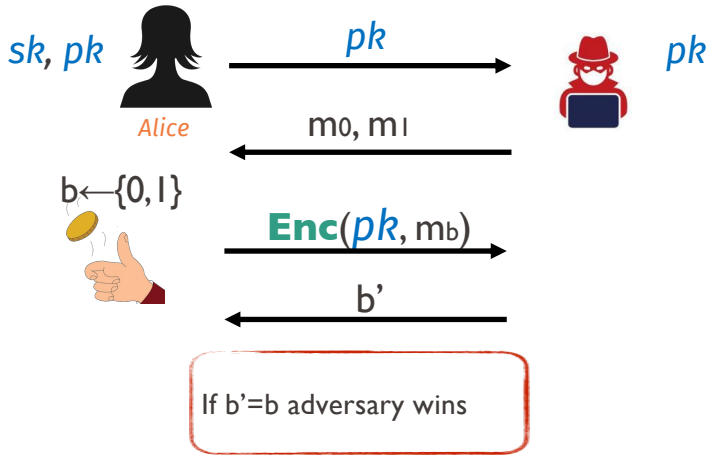


Game-based security definition: method to capture a security definition using a game between an adversary and a challenger. If the adversary has a non-negligible advantage to win the game, the adversary "breaks" the security property.

FYI

Semantic security (IND-CPA) of Homomorphic encryption

Game-based security: is a method to capture a security definition using a game between an adversary and a challenger. If the adversary wins the game, the adversary "breaks" the security property.



$(sk, pk) \leftarrow \mathbf{KeyGen}(1^k)$

$(m_0, m_1) \leftarrow \mathcal{A}(pk)$ // *adversary chooses two messages*

$b \leftarrow \{0,1\}$; // *random bit is chosen*

$\mathbf{c} \leftarrow \mathbf{Enc}(pk, m_b)$ // *message b is encrypted*

$b' \leftarrow \mathcal{A}(\mathbf{c})$ // *adversary must figure out b*

Semantic security. HE is secure if $\frac{1}{2}\text{-Pr}[b' = b] = \text{negl}(k)$

Semantic security – intuition

seeing encryption of messages does not give the adversary information that helps them guessing the encrypted message better than random guess

Homomorphic Encryption
Is this an HE scheme?

Consider the following scheme built on top of a IND-CPA secure **PKE** scheme:

KeyGen(1^k) = **PKE.KeyGen**(1^k)

Enc(*pk*, *m*) = {$\boldsymbol{c}_{PKE}$ = **PKE.Enc**(*pk*, *m*); return **c**=($\boldsymbol{c}_{PKE}$, nil)}

Eval(*pk*, f, ***c***) = {$\boldsymbol{c}_{PKE}$ = ***c***[0]; return ($\boldsymbol{c}_{PKE}$, f)}

Dec(*sk*, ***c***) = {$\boldsymbol{c}_{PKE}$, f ⟵ ***c***; return f(**PKE. Dec**(*sk*, $\boldsymbol{c}_{PKE}$))}

1. Is it correct Encryption scheme?
2. Is it IND-CPA secure?
3. Is it useful?

Homomorphic Encryption
Is this an HE scheme?

Consider the following scheme built on top of a IND-CPA secure **PKE** scheme:

KeyGen(1^k) = **PKE.KeyGen**(1^k)

Enc(*pk*, *m*) = {$\boldsymbol{c}_{PKE}$ = **PKE.Enc**(*pk*, *m*); return **c**=($\boldsymbol{c}_{PKE}$, nil)}

Eval(*pk*, f, ***c***) = {$\boldsymbol{c}_{PKE}$ = ***c***[0]; return ($\boldsymbol{c}_{PKE}$, f)}

Dec(*sk*, ***c***) = {$\boldsymbol{c}_{PKE}$, f ⟵ ***c***; return f(**PKE. Dec**(*sk*, $\boldsymbol{c}_{PKE}$))}

1. Is it correct Encryption scheme? Yes, straightforward from the decryption
2. Is it IND-CPA secure? Yes, as it would break the IND-CPA from **PKE**
3. Is it useful? **No**, the evaluation of f is performed at decryption…

Homomorphic Encryption
Compactness

Compactness – *intuition*: the ciphertext size should not be growing through homomorphic operations

Compactness. HE *compactly evaluates* a family of functions $\mathcal{F}$ if

$$\forall(sk, pk) \leftarrow \mathbf{KeyGen}(1^k), \quad \forall f \in \mathcal{F}, \quad \forall m_i \in \mathcal{P}:$$

There exists a polynomial p() such that the size of

$$|\mathbf{Eval}(pk, f, \mathbf{c}_1, \ldots, \mathbf{c}_n)| < p(k)$$, with k the security parameter, *independent of* f();

i.e., the complexity of **Dec** is independent of f().

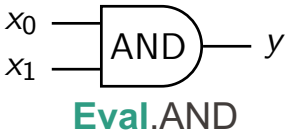
This definition is relaxed in practice

Homomorphic Encryption
Composition

Composition – *intuition:* Build HE computation from a set of simple operation (think Circuits and CPU)

Define **Eval** as circuit of simple “gates”. E.g., Boolean Algebra

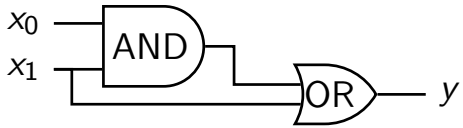
x_0	x_1	$\mathrm{AND}(x_0, x_1)$
0	0	0
0	1	0
1	0	0
1	1	1



Eval.AND

x_0	x_1	$\mathrm{OR}(x_0, x_1)$
0	0	0
0	1	1
1	0	1
1	1	1

Eval.OR



x_0	x_1	$\mathrm{OR}(\mathrm{AND}(x_0, x_1), x_1)$
0	0	0
0	1	1
1	0	0
1	1	1

Figure copied from Christian Mouchet’s Course on Computing on Encrypted Data – HPI 2024

Homomorphic Encryption
Composition

Composition – *intuition:* Build HE computation from a set of simple operation (think Circuits and CPU)

$\forall f \in \mathcal{F}$, $f: \mathcal{P}^n \to \mathcal{P}$, $\forall (m_1,\ldots, m_n) \in \mathcal{P}^n$, $(sk, pk) \leftarrow \mathbf{KeyGen}(1^k)$, and $\forall i \in [n]$ $\boldsymbol{c_i} = \mathbf{Enc}(pk, m_i)$

- **Correctness for n-ary functions:**

$$\mathbf{Dec}(sk, \mathbf{Eval}(pk, f, \boldsymbol{c_1},\ldots,\boldsymbol{c_n}) = f(m_1, \ldots, m_n)$$

- **Composability:**

$$\mathbf{Dec}(sk, \mathbf{Eval}(pk, f, \boldsymbol{c_1},\ldots,\boldsymbol{c_n}) = f(\mathbf{Dec}(sk, \boldsymbol{c_1}), \ldots, \mathbf{Dec}(sk, \boldsymbol{c_n}))$$

Informally extends correctness to any *valid* ciphertext (fresh or **Eval**)

Homomorphic Encryption
Summary of the properties

$\forall f \in \mathcal{F}$, f: $\mathcal{P}^n \rightarrow \mathcal{P}$, $\forall (m_1,\ldots, m_n) \in \mathcal{P}^n$,

$(sk, pk) \leftarrow \mathbf{KeyGen}(1^k)$, and $\forall i \in [n]$ $c_i = \mathbf{Enc}(pk, m_i)$

Correctness

$\mathbf{Dec}(sk, \mathbf{Enc}(pk, m_1)) = f(m_1)$

IND-CPA Security

$\text{Adv}^{\text{IND-CPA}}_{\text{HE},\mathcal{A}} = \text{negl}(k)$

Evaluation correctness and composition

$\mathbf{Dec}(sk, \mathbf{Eval}(pk, f, c_1,\ldots,c_n) = f(m_1, \ldots, m_n)$

$\mathbf{Dec}(sk, \mathbf{Eval}(pk, f, c_1,\ldots,c_n) = f(\mathbf{Dec}(sk, c_1), \ldots, \mathbf{Dec}(sk, c_n))$

Compactness

$|\mathbf{Eval}(pk,f,c_1,\ldots,c_n)| < p(k)$ for p polynomial Independent of f

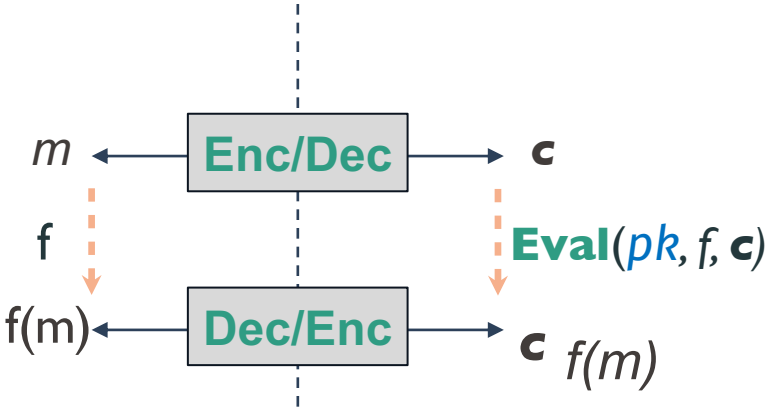
HE scheme:

$\mathbf{KeyGen}(1^k) \rightarrow (sk, pk)$

$\mathbf{Enc}(pk, m; r) \rightarrow c$

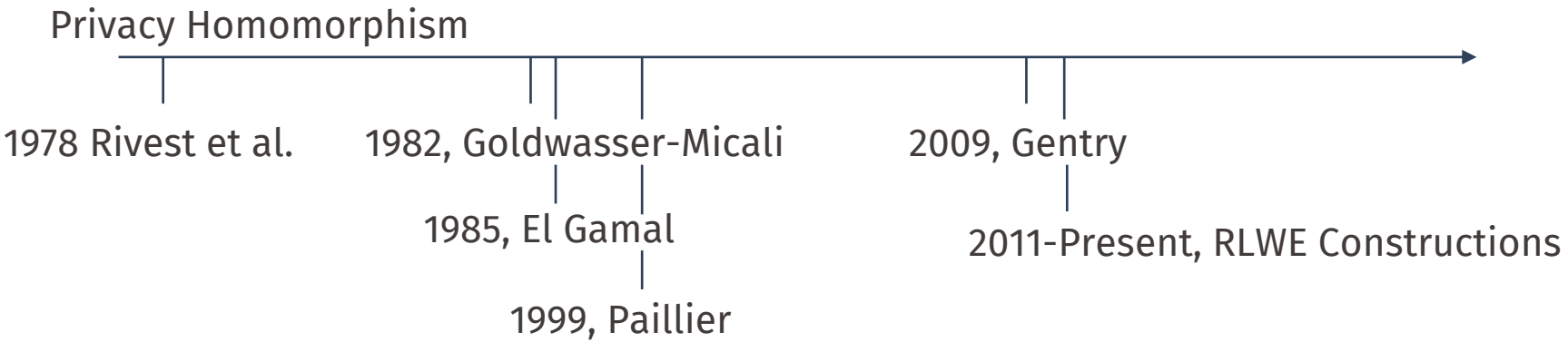
$\mathbf{Dec}(sk, c) \rightarrow m$

$\mathbf{Eval}(pk, f, c_1, \ldots, c_n) \rightarrow c_{f()}$



Constructions

Homomorphic Encryption
History



We will cover

- ~~RSA~~
- El Gamal
- ~~Paillier~~
- RLWE

El Gamal Encryption
Properties

ElGamal(G, g):
Let G be a cyclic group of order q=|G| generated by g

KeyGen$\rightarrow$(sk, pk):
$sk \leftarrow \$\{1,\ldots, q-1\}$, $pk = g^{sk}$

Enc(pk, $m \in G$)$\rightarrow$ $\boldsymbol{c}$
$r \leftarrow \$\{1,\ldots, q-1\}$
$\boldsymbol{c} := (m \cdot pk^r, g^r)$

Dec(sk, $\boldsymbol{c}$)$\rightarrow m$
$m := \boldsymbol{c}[0] / \boldsymbol{c}[1]^{sk}$

Security – Security from the Discrete Logarithm:
Given g and $y = g^x \bmod p$, find x

Correctness –

Evaluation – Component-wise _____

Compact –

El Gamal Encryption
Properties

ElGamal(G, g):
Let G be a cyclic group of order q=|G| generated by g

KeyGen$\rightarrow$(sk, pk):
$sk \leftarrow \$\{1,\ldots, q-1\}$, $pk = g^{sk}$

Enc(pk, $m \in G$)$\rightarrow$ $\boldsymbol{c}$
r $\leftarrow \$\{1,\ldots, q-1\}$
$\boldsymbol{c} := (m \cdot pk^r, g^r)$

Dec(sk, $\boldsymbol{c}$)$\rightarrow m$
$m := \boldsymbol{c}[0] / \boldsymbol{c}[1]^{sk}$

Security – Security from the Discrete Logarithm:
Given g and $y = g^x$ mod p, find x

Correctness – Follows by definition
Dec(sk, **Enc**(pk, m)) = **Dec**(sk, $(m \cdot pk^r, g^r)$)
$= m \cdot pk^r \cdot (g^r)^{-sk} = m \cdot g^{r \cdot sk} \cdot (g^r)^{-sk} = m$

Evaluation – Component-wise _____

Compact –

El Gamal Encryption
Properties

ElGamal(G, g):
Let G be a cyclic group of order q=|G| generated by g

KeyGen$\rightarrow$(sk, pk):
$sk \leftarrow\$\{1,\ldots, q-1\}$, $pk=g^{sk}$

Enc(pk, $m \in G$)$\rightarrow$ $\boldsymbol{c}$
$r \leftarrow\$\{1,\ldots, q-1\}$
$\boldsymbol{c} := (m \cdot pk^r, g^r)$

Dec(sk, $\boldsymbol{c}$)$\rightarrow m$
$m := \boldsymbol{c}[0] / \boldsymbol{c}[1]^{sk}$

Security – Security from the Discrete Logarithm:
Given g and $y = g^x \bmod p$, find x

Correctness – Follows by definition
Dec(sk, **Enc**(pk, m)) = **Dec**(sk, $(m \cdot pk^r, g^r)$)
$= m \cdot pk^r \cdot (g^r)^{-sk} = m \cdot g^{r \cdot sk} \cdot (g^r)^{-sk} = m$

Evaluation – Component-wise **multiplication**
$\boldsymbol{c}_1$= **Enc**(pk, m_1) and $\boldsymbol{c}_2$= **Enc**(pk, m_2)
Mul($\boldsymbol{c}_1$, $\boldsymbol{c}_2$) = $(\boldsymbol{c}_1[0] \cdot \boldsymbol{c}_2[0], \boldsymbol{c}_1[1] \cdot \boldsymbol{c}_2[1])$
$= (m_1 \cdot pk^{r_1} \cdot m_2 \cdot pk^{r_2}, g^{r_1} \cdot g^{r_2})$
$= (m_1 \cdot m_2 \cdot pk^{r_1+r_2}, g^{r_1+r_2}) = \boldsymbol{c}$
Dec(sk, $\boldsymbol{c}$) = $m_1 \cdot m_2$

Compact –

El Gamal Encryption
Properties

ElGamal(G, g):
Let G be a cyclic group of order q=|G| generated by g

KeyGen$\rightarrow$(sk, pk):
$sk \leftarrow\$\{1,\ldots, q-1\}$, $pk=g^{sk}$

Enc(pk, $m \in G$)$\rightarrow$ $\boldsymbol{c}$
$r \leftarrow\$\{1,\ldots, q-1\}$
$\boldsymbol{c} := (m \cdot pk^r, g^r)$

Dec(sk, $\boldsymbol{c}$)$\rightarrow m$
$m := \boldsymbol{c}[0] / \boldsymbol{c}[1]^{sk}$

Security – Security from the Discrete Logarithm:
Given g and $y = g^x \bmod p$, find x

Correctness – Follows by definition
Dec(sk, **Enc**(pk, m)) = **Dec**(sk, $(m \cdot pk^r, g^r)$)
$= m \cdot pk^r \cdot (g^r)^{-sk} = m \cdot g^{r \cdot sk} \cdot (g^r)^{-sk} = m$

Evaluation – Component-wise **multiplication**
$\boldsymbol{c}_1$= **Enc**(pk, m_1) and $\boldsymbol{c}_2$= **Enc**(pk, m_2)
Mul($\boldsymbol{c}_1$, $\boldsymbol{c}_2$) = $(\boldsymbol{c}_1[0] \cdot \boldsymbol{c}_2[0], \boldsymbol{c}_1[1] \cdot \boldsymbol{c}_2[1])$
$= (m_1 \cdot pk^{r_1} \cdot m_2 \cdot pk^{r_2}, g^{r_1} \cdot g^{r_2})$
$= (m_1 \cdot m_2 \cdot pk^{r_1+r_2}, g^{r_1+r_2}) = \boldsymbol{c}$
Dec(sk, $\boldsymbol{c}$) = $m_1 \cdot m_2$

Compact – yes

Additive El Gamal Encryption Properties

AddElGamal(G, g):
Let G be a cyclic group of order q=|G| generated by g

KeyGen$\rightarrow$(*sk*, *pk*):
sk $\leftarrow$\${1,…, q-1}, *pk*=g^{sk}

Enc(*pk*, *m* $\in$ G)$\rightarrow$ ***c***
r $\leftarrow$\${1,…, q-1}
c := $(g^{m} \cdot pk^{r}, g^{r})$

Dec(*sk*, ***c***)$\rightarrow$*m*
$m := \log_g (c[0] / c[1]^{sk})$

Useful plaintext space:

$(\mathbb{Z}_n, +)$ closed additive group of integers modulo n.

Supports any linear combination:

$$f(x_1, \dots, x_\ell) = \sum_{k=1}^{\ell} \delta_i \cdot x_i \bmod n$$

We encode the plaintext in the exponent $m \rightarrow g^m$

c_1 = **Enc**(*pk*, m_1) and c_2 = **Enc**(*pk*, m_2)

Mul(c_1, c_2) = $(c_1[0] \cdot c_2[0], c_1[1] \cdot c_2[1])$
$= (g^{m_1} \cdot pk^{r_1} \cdot g^{m_2} \cdot pk^{r_2}, g^{r_1} \cdot g^{r_2})$
$= (g^{m_1+m_2} \cdot pk^{r_1+r_2}, g^{r_1+r_2}) = c$

Dec(*sk*, ***c***) = $m_1 + m_2$

Not efficient for general case

Exercise

Paillier Properties

Paillier(L):

KeyGen$\rightarrow(sk, pk)$:

p,q$\leftarrow \mathbb{P}_L$

$n := pq$

$pk=(n, g = n+1)$, $sk=\phi(n)$

Enc$(pk, m \in \mathbb{Z}_n) \rightarrow \boldsymbol{c}$

$r \leftarrow \mathbb{Z}_n^*$

$\boldsymbol{c} := g^m \cdot r^n \bmod n^2$

Dec$(sk, \boldsymbol{c}) \rightarrow m = \frac{(\boldsymbol{c}^{sk} \bmod n^2)-1}{n} sk^{-1} \bmod n$

Security – Security from the Decisional Composite Residuosity assumption:

Evaluation –

Correctness – Uses two facts

1. $\forall x \in \mathbb{Z}_{n^2}^*, (x^n)^{\phi(n)} = 1 \bmod n^2$
2. The base-g DL in $\mathbb{Z}_{n^2}^*$ is easy to compute

Exercise

Textbook RSA Encryption
Properties

RSA(L):

KeyGen$\rightarrow$(*sk*, *pk*):
p,q$\leftarrow\mathbb{P}$ s.t. log(pq)≥L
n := pq, e $\leftarrow \mathbb{Z}^*_{\phi(n)}$
pk=(n,e), *sk*=e^{-1} mod $\phi(n)$

Enc(*pk*, $m \in \mathbb{Z}^*_n$)$\rightarrow$ ***c***= m^e mod n
Dec(*sk*, ***c***)$\rightarrow$*m*=$\boldsymbol{c}^{sk}$ mod n

Security – Security based on factoring hardness:
Given n = pq s.t. p,q$\leftarrow\mathbb{P}$, find p and q
Equivalent to finding $\phi(n)$

Correctness –

Evaluation –

Compact –

Lattice-Based Constructions

Fully Homomorphic Encryption
A simple scheme for intuition

$\mathcal{R}_q = \mathbb{Z}_q[X]/(X^N + 1)$

IdealHE:

KeyGen$\rightarrow$(*sk*,$\cdot$):
$sk := s \leftarrow \$\, \mathcal{R}_q$

Enc(*sk*, $m \in \mathcal{R}_q$)$\rightarrow$ ***c***
$a \leftarrow \$\, \mathcal{R}_q$
$\boldsymbol{c} := (-a \cdot sk + m,\ a)$

Dec(*sk*, ***c***)$\rightarrow m$
$m = \boldsymbol{c}[0] + sk \cdot \boldsymbol{c}[1]$

Consider the polynomial ring $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^N + 1)$
i.e., degree $N - 1$ polynomials with coefficients in $\mathbb{Z}_q$

Idea inspired from Christian Mouchet's Course on Computing on Encrypted Data – HPI 2024

Fully Homomorphic Encryption
A simple scheme for intuition – Correctness

IdealHE: $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^N + 1)$

KeyGen$\rightarrow(sk,\cdot)$:
$sk := s \leftarrow \$\, \mathcal{R}_q$

Enc$(sk, m \in \mathcal{R}_q) \rightarrow \boldsymbol{c}$
$a \leftarrow \$\, \mathcal{R}_q$
$\boldsymbol{c} := (-a \cdot sk + m,\ a)$

Dec$(sk, \boldsymbol{c}) \rightarrow m$
$m = \boldsymbol{c}[0] + sk \cdot \boldsymbol{c}[1]$

Consider the polynomial ring $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^N + 1)$
i.e., degree $N - 1$ polynomials with coefficients in $\mathbb{Z}_q$

Correctness – $\forall\ sk \in \mathcal{R}_q$ and $\forall\ m \in \mathcal{R}_q$

$$\mathbf{Dec}(sk, \mathbf{Enc}(sk, m)) = (-a \cdot sk + m) + sk \cdot a$$
$$= (-as + m) + s \cdot a = m$$

Fully Homomorphic Encryption
A simple scheme for intuition – Addition

$\mathcal{R}_q = \mathbb{Z}_q[X]/(X^N + 1)$

IdealHE:

KeyGen$\rightarrow(sk,\cdot)$:
$sk := s \leftarrow \$\, \mathcal{R}_q$

Enc$(sk, m \in \mathcal{R}_q) \rightarrow \boldsymbol{c}$
$a \leftarrow \$\, \mathcal{R}_q$
$\boldsymbol{c} := (-a \cdot sk + m,\ a)$

Dec$(sk, \boldsymbol{c}) \rightarrow m$
$m = \boldsymbol{c}[0] + sk \cdot \boldsymbol{c}[1]$

Evaluation – $\forall\ sk \in \mathcal{R}_q$ and $\forall\ m_1, m_2 \in \mathcal{R}_q$

$\boldsymbol{c}_1 =$ **Enc**(sk, m_1) and $\boldsymbol{c}_2 =$ **Enc**(sk, m_2)

Fully Homomorphic Encryption
A simple scheme for intuition – Addition

$\mathcal{R}_q = \mathbb{Z}_q[X]/(X^N + 1)$

IdealHE:

KeyGen$\rightarrow(sk,\cdot)$:
$sk := s \leftarrow \$\, \mathcal{R}_q$

Enc$(sk, m \in \mathcal{R}_q) \rightarrow \boldsymbol{c}$
$a \leftarrow \$\, \mathcal{R}_q$
$\boldsymbol{c} := (-a \cdot sk + m,\ a)$

Dec$(sk, \boldsymbol{c}) \rightarrow m$
$m = \boldsymbol{c}[0] + sk \cdot \boldsymbol{c}[1]$

Evaluation – $\forall\ sk \in \mathcal{R}_q$ and $\forall\ m_1, m_2 \in \mathcal{R}_q$

$\boldsymbol{c}_1 =$ **Enc**(sk, m_1) and $\boldsymbol{c}_2 =$ **Enc**(sk, m_2)

- Addition

$$\begin{aligned}\boldsymbol{c}_1 + \boldsymbol{c}_2 &= (-a_1 \cdot sk + m_1, a_1) + (-a_2 \cdot sk + m_2, a_2)\\ &= (-a_1 \cdot sk + m_1 - a_2 \cdot sk + m_2, a_1 + a_2)\\ &= (-(a_1+a_2) \cdot sk + m_1 + m_2, a_1 + a_2\\ &= (-b \cdot sk + m_1 + m_2, b_{add}) \text{ with } b_{add} = a_1 + a_2\end{aligned}$$

$\boldsymbol{c}_1 + \boldsymbol{c}_2 =$ **Enc**$(sk, m_1 + m_2)$

So Add$(\boldsymbol{c}_1, \boldsymbol{c}_2) = \boldsymbol{c}_1 + \boldsymbol{c}_2$

Fully Homomorphic Encryption
A simple scheme for intuition – Multiplication

$\mathcal{R}_q = \mathbb{Z}_q[X]/(X^N + 1)$

IdealHE:

KeyGen$\rightarrow(sk,\cdot)$:
$sk := s \leftarrow \$\, \mathcal{R}_q$

Enc$(sk, m \in \mathcal{R}_q) \rightarrow \boldsymbol{c}$
$a \leftarrow \$\, \mathcal{R}_q$
$\boldsymbol{c} := (-a \cdot sk + m,\ a)$

Dec$(sk, \boldsymbol{c}) \rightarrow m$
$m = \boldsymbol{c}[0] + sk \cdot \boldsymbol{c}[1]$

Evaluation – $\forall\ sk \in \mathcal{R}_q$ and $\forall\ m_1, m_2 \in \mathcal{R}_q$

$\boldsymbol{c}_1 =$ **Enc**(sk, m_1) and $\boldsymbol{c}_2 =$ **Enc**(sk, m_2)

- **Multiplication?**

$$\begin{aligned}\boldsymbol{c}_1 \cdot \boldsymbol{c}_2 &= (-a_1 \cdot sk + m_1, a_1) \cdot (-a_2 \cdot sk + m_2, a_2) \\ &= ((-a_1 \cdot sk + m_1) \cdot (-a_2 \cdot sk + m_2), a_1 \cdot a_2) \\ &= sk^2 a_1 a_2 - sk(a_1 m_2 + m_1 a_2) + m_1 m_2, a_1 \cdot a_2)\end{aligned}$$

Cross-terms that cannot be reconstructed at decryption!

Fully Homomorphic Encryption
A simple scheme for intuition – Multiplication

$\mathcal{R}_q = \mathbb{Z}_q[X]/(X^N + 1)$

IdealHE:

KeyGen$\rightarrow(sk,\cdot)$:
$sk := s \leftarrow \$\, \mathcal{R}_q$

Enc$(sk, m \in \mathcal{R}_q) \rightarrow \boldsymbol{c}$
$a \leftarrow \$\, \mathcal{R}_q$
$\boldsymbol{c} := (-a \cdot sk + m,\ a)$

Dec$(sk, \boldsymbol{c}) \rightarrow m$
$m = \boldsymbol{c}[0] + sk \cdot \boldsymbol{c}[1]$

Evaluation – $\forall\ sk \in \mathcal{R}_q$ and $\forall\ m_1, m_2 \in \mathcal{R}_q$

$\boldsymbol{c}_1 =$ **Enc**(sk, m_1) and $\boldsymbol{c}_2 =$ **Enc**(sk, m_2)

- **Multiplication?**

$$\begin{aligned}\boldsymbol{c}_1 \cdot \boldsymbol{c}_2 &= (-a_1 \cdot sk + m_1, a_1) \cdot (-a_2 \cdot sk + m_2, a_2) \\ &= ((-a_1 \cdot sk + m_1) \cdot (-a_2 \cdot sk + m_2), a_1 \cdot a_2) \\ &= sk^2 a_1 a_2 - sk(a_1 m_2 + m_1 a_2) + m_1 m_2, a_1 \cdot a_2)\end{aligned}$$

Cross-terms that cannot be reconstructed at decryption!

Fully Homomorphic Encryption
A simple scheme for intuition – Multiplication

$\mathcal{R}_q = \mathbb{Z}_q[X]/(X^N + 1)$

IdealHE:

KeyGen$\rightarrow(sk,\cdot)$:
$sk := s \leftarrow \$\, \mathcal{R}_q$

Enc$(sk, m \in \mathcal{R}_q) \rightarrow \boldsymbol{c}$
$a \leftarrow \$\, \mathcal{R}_q$
$\boldsymbol{c} := (-a \cdot sk + m,\ a)$

Dec$(sk, \boldsymbol{c}) \rightarrow m$
$m = \boldsymbol{c}[0] + sk \cdot \boldsymbol{c}[1]$

Evaluation – $\forall\ sk \in \mathcal{R}_q$ and $\forall\ m_1, m_2 \in \mathcal{R}_q$

$\boldsymbol{c}_1 =$ **Enc**(sk, m_1) and $\boldsymbol{c}_2 =$ **Enc**(sk, m_2)

- **Multiplication: Tensor product**

$$\boldsymbol{c} = \boldsymbol{c}_1 \times \boldsymbol{c}_2 = \begin{pmatrix} \boldsymbol{c}_1[0] \cdot \boldsymbol{c}_2[0] \\ \boldsymbol{c}_1[0] \cdot \boldsymbol{c}_2[1] + \boldsymbol{c}_1[1] \cdot \boldsymbol{c}_2[0] \\ \boldsymbol{c}_1[1] \cdot \boldsymbol{c}_2[1] \end{pmatrix}$$

$$\boldsymbol{c} = \boldsymbol{c}_1 \times \boldsymbol{c}_2 = \begin{pmatrix} sk^2 c_{mul} - sk\, b_{mul} + m_1 m_2 \\ -2\, sk\, c_{mul} + b_{mul} \\ c_{mul} \end{pmatrix}$$

$c_{mul} = a_1 \cdot a_2$ $\qquad$ $b_{mul} = a_1 m_2 + m_1 a_2$

Fully Homomorphic Encryption
A simple scheme for intuition – Multiplication

$\mathcal{R}_q = \mathbb{Z}_q[X]/(X^N + 1)$

IdealHE:

KeyGen$\rightarrow(sk,\cdot)$:
$sk := s \leftarrow \$\, \mathcal{R}_q$

Enc$(sk, m \in \mathcal{R}_q) \rightarrow \boldsymbol{c}$
$a \leftarrow \$\, \mathcal{R}_q$
$\boldsymbol{c} := (-a \cdot sk + m,\ a)$

Dec$(sk, \boldsymbol{c}) \rightarrow m$
$m = \boldsymbol{c}[0] + sk \cdot \boldsymbol{c}[1]$

Evaluation – $\forall\ sk \in \mathcal{R}_q$ and $\forall\ m_1, m_2 \in \mathcal{R}_q$

$\boldsymbol{c}_1 =$ **Enc**(sk, m_1) and $\boldsymbol{c}_2 =$ **Enc**(sk, m_2)

- **Multiplication: Tensor product + Decryption**

$$\boldsymbol{c}_1 \times \boldsymbol{c}_2 = \begin{pmatrix} sk^2 c_{mul} - sk\, b_{mul} + m_1 m_2 \\ -2\, sk\, c_{mul} + b_{mul} \\ c_{mul} \end{pmatrix}$$

Define **Dec'_2**$(sk, \boldsymbol{c}) \rightarrow m = \boldsymbol{c}[0] + sk \cdot \boldsymbol{c}[1] + sk^2 \cdot \boldsymbol{c}[2]$

This extend to larger depth.

Degree-2 ciphertext

Relinearization: converts to degree 1-ciphertext

FYI

Fully Homomorphic Encryption
A simple scheme for intuition – Multiplication

$\mathcal{R}_q = \mathbb{Z}_q[X]/(X^N + 1)$

IdealHE:

KeyGen$\rightarrow(sk,\cdot)$:
$sk := s \leftarrow \$\, \mathcal{R}_q$

Enc$(sk, m \in \mathcal{R}_q) \rightarrow \boldsymbol{c}$
$a \leftarrow \$\, \mathcal{R}_q$
$\boldsymbol{c} := (-a \cdot sk + m,\ a)$

Dec$(sk, \boldsymbol{c}) \rightarrow m$
$m = \boldsymbol{c}[0] + sk \cdot \boldsymbol{c}[1]$

Compactness – Need for **relinearization**

We have seen that homomorphic multiplication **increases the size of the ciphertexts**.

Relinearization: converts a degree-2 ciphertext ***c*** *into a degree-1 ciphertext* ***c'***.

Requires: *a relinearization key* ***rlk*** = **Enc**$(sk,\ sk^2)$

$$\boldsymbol{c}' = \text{Relin}(\boldsymbol{rlk}, \boldsymbol{c}) = \begin{pmatrix} \boldsymbol{c}[0] + \boldsymbol{c}[2] \cdot \boldsymbol{rlk}[0] \\ \boldsymbol{c}[1] + \boldsymbol{c}[2] \cdot \boldsymbol{rlk}[1] \end{pmatrix}$$

Fully Homomorphic Encryption
A simple scheme for intuition – Summary

$\mathcal{R}_q = \mathbb{Z}_q[X]/(X^N + 1)$

IdealHE:

KeyGen$\rightarrow$(*sk*, ***rlk***):
- ***rlk*** = **Enc**(*sk*, sk^2)
- $sk := s \leftarrow \$\, \mathcal{R}_q$

Enc(*sk*, $m \in \mathcal{R}_q$)$\rightarrow$ ***c***
- $a \leftarrow \$\, \mathcal{R}_q$
- $\boldsymbol{c} := (-a \cdot sk + m,\ a)$

Relin(***rlk***, ***c***) $= (\boldsymbol{c}[0], \boldsymbol{c}[1]) + \boldsymbol{c}[2] \cdot \boldsymbol{rlk}$

Dec(*sk*, ***c***)$\rightarrow m = \boldsymbol{c}[0] + sk \cdot \boldsymbol{c}[1]$

Add($\boldsymbol{c}_1$, $\boldsymbol{c}_2$) = $\boldsymbol{c}_1 + \boldsymbol{c}_2$

Mul($\boldsymbol{c}_1$, $\boldsymbol{c}_2$) = **Relin**(***rlk***, $\boldsymbol{c}_1 \times \boldsymbol{c}_2$)

Correctness –

Evaluation –

Compactness –

Public Key –

Security –

Fully Homomorphic Encryption
A simple scheme for intuition – Summary

$\mathcal{R}_q = \mathbb{Z}_q[X]/(X^N + 1)$

IdealHE:

KeyGen$\rightarrow$(*sk*, ***rlk***):
- ***rlk*** = **Enc**(*sk*, sk^2)
- $sk := s \leftarrow \$\, \mathcal{R}_q$

Enc(*sk*, $m \in \mathcal{R}_q$)$\rightarrow$ ***c***
- $a \leftarrow \$\, \mathcal{R}_q$
- $\boldsymbol{c} := (-a \cdot sk + m,\ a)$

Relin(***rlk***, ***c***) =$(\boldsymbol{c}[0], \boldsymbol{c}[1]) + \boldsymbol{c}[2] \cdot \boldsymbol{rlk}$

Dec(*sk*, ***c***)$\rightarrow m = \boldsymbol{c}[0] + sk \cdot \boldsymbol{c}[1]$

Add($\boldsymbol{c}_1$, $\boldsymbol{c}_2$) = $\boldsymbol{c}_1 + \boldsymbol{c}_2$

Mul($\boldsymbol{c}_1$, $\boldsymbol{c}_2$) = **Relin**(***rlk***, $\boldsymbol{c}_1 \times \boldsymbol{c}_2$)

Correctness – Yes

Evaluation – Additions and Multiplications

Compactness – with **relinearization**

Public Key – Yes, ***pk*** = **Enc**(*sk*,*0*) and $u \cdot \boldsymbol{pk}$ to **Enc** for $u \leftarrow \mathcal{R}_q$

Security –

Fully Homomorphic Encryption
A simple scheme for intuition – Summary

$\mathcal{R}_q = \mathbb{Z}_q[X]/(X^N + 1)$

IdealHE:

KeyGen$\rightarrow$(*sk*, ***rlk***):
- ***rlk*** = **Enc**(*sk*, sk^2)
- $sk := s \leftarrow \$\, \mathcal{R}_q$

Enc(*sk*, $m \in \mathcal{R}_q$)$\rightarrow$ ***c***
- $a \leftarrow \$\, \mathcal{R}_q$
- $\boldsymbol{c} := (-a \cdot sk + m,\ a)$

Relin(***rlk***, ***c***) =$(\boldsymbol{c}[0], \boldsymbol{c}[1]) + \boldsymbol{c}[2] \cdot \boldsymbol{rlk}$

Dec(*sk*, ***c***)$\rightarrow m = \boldsymbol{c}[0] + sk \cdot \boldsymbol{c}[1]$

Add($\boldsymbol{c}_1$, $\boldsymbol{c}_2$) = $\boldsymbol{c}_1 + \boldsymbol{c}_2$

Mul($\boldsymbol{c}_1$, $\boldsymbol{c}_2$) = **Relin**(***rlk***, $\boldsymbol{c}_1 \times \boldsymbol{c}_2$)

Correctness – Yes

Evaluation – Additions and Multiplications

Compactness – with **relinearization**

Public Key – Yes, ***pk*** = **Enc**(*sk*,*0*) and $u \cdot \boldsymbol{pk}$ to **Enc** for $u \leftarrow \mathcal{R}_q$

Security – **No!** The problem is not hard. We need the error!

FYI

Lattice-Based Homomorphic Encryption
Background on lattices

A lattice is a mathematical object: a subset of $\mathbb{R}^n$ that is an additive subgroup and discrete

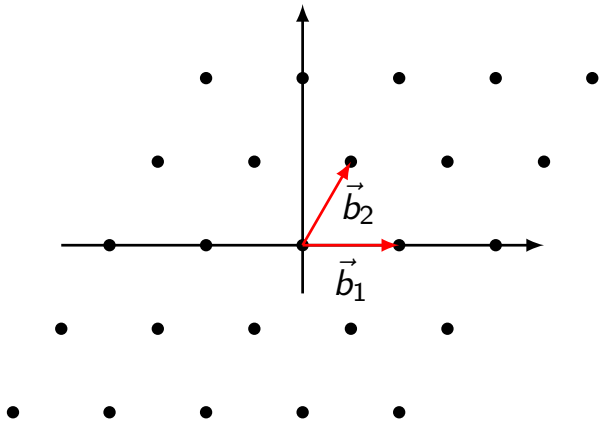
There exits many problems deemed hard related to lattices

e.g., Shortest Vector Problem: find non-zero vector of low norm

- Large mathematical objects

- Several lattice-based hard problems

- Assumed quantum resistant for appropriate parametrization



$$\mathcal{L} = \left\{ \sum_{i=1}^{n} z_i \vec{b}_i \quad | \quad z_i \in \mathbb{Z} \right\}$$

Figure copied from Christian Mouchet's Course on Computing on Encrypted Data – HPI 2024

Ring learning with error
Intuition

Define $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^N + 1)$ Degree $N - 1$ polynomials with coefficients in $\mathbb{Z}_q$

Given a secret vector $s \in \mathcal{R}_q$, the $\mathrm{RLWE}^s_{\mathcal{R}_q,\chi}$ distribution outputs:

$$(a\ b) = (a, as + e) \in \mathcal{R}_q^2$$

where $\vec{a} \leftarrow \mathcal{R}_q$ and $\mathrm{e} \leftarrow \chi$ m=poly(n), appropriate q, and χ of error rate $\alpha < 1$

Cannot distinguish between $\mathrm{RLWE}^s_{\mathcal{R}_q,\chi}$ and a uniform distribution

Ring learning with error-based FHE scheme
Issue with correctness

$$\mathcal{R}_q = \mathbb{Z}_q[X]/(X^N + 1)$$

FHE:

KeyGen$\rightarrow$(*sk*, $\cdot$): $a, b \leftarrow \mathcal{R}_q$

sk $:= s \leftarrow \$\, \mathcal{R}_q$

Enc(*sk*, $m \in \mathcal{R}_q$)$\rightarrow$ ***c***

$a \leftarrow \$\, \mathcal{R}_q$

c $:= (-a \cdot sk + m,\ a)$

Dec(*sk*, ***c***)$\rightarrow m =$ ***c***[0] + *sk* $\cdot$ ***c***[1]

For simplicity, the relinearization and PKE have been omitted. In practice, it is non-trivial to enable relinearization: use gadgets products

Disclaimer:

Idea: add noise everywhere!

$(a, b) = (a, as + e) \in \mathcal{R}_q^2$ with $\vec{a} \leftarrow \mathcal{R}_q$ and e $\leftarrow \chi$

Informal: RLWE assumption says this is indistinguishable from $(a, b \leftarrow \$\ \mathcal{R}_q)$.

Ring learning with error-based FHE scheme
Issue with correctness

$$\mathcal{R}_q = \mathbb{Z}_q[X]/(X^N + 1)$$

FHE:

KeyGen$\rightarrow$(*sk*, ·): $a, b \leftarrow \mathcal{R}_q$

sk $:= s \leftarrow \$\, \mathcal{R}_q$

Enc(*sk*, $m \in \mathcal{R}_q$)$\rightarrow$ ***c***

$a \leftarrow \$\, \mathcal{R}_q$ $e_{enc} \leftarrow \chi$

c $:= (-a \cdot sk + e_{enc} + m\,,\ \ a)$

Dec(*sk*, ***c***)$\rightarrow m$ = ***c***[0] +*sk* · ***c***[1]

For simplicity, the relinearization and PKE have been omitted. In practice, it is non-trivial to enable relinearization: use gadgets products

Disclaimer:

Idea: add noise everywhere!

$(a, b) = (a, as + e) \in \mathcal{R}_q^2$ with $\vec{a} \leftarrow \mathcal{R}_q$ and $e \leftarrow \chi$

Correctness – $\forall$ *sk* $\in \mathcal{R}_q$ and $\forall$ $m \in \mathcal{R}_q$

Dec(*sk*, **Enc**(*sk*, *m*)) = $(-a \cdot sk + e_{enc} + m) + sk \cdot a$

$= (-as + e_{enc} + m) + s \cdot a$

$= e_{enc} + m \neq m$

We need to perform error correction:

1. Message scaling
2. Noise scaling

Ring learning with error-based FHE scheme
Adding error correction – message scaling

$$\mathcal{R}_q = \mathbb{Z}_q[X]/(X^N + 1)$$

FHE:

KeyGen$\rightarrow$(*sk*, ·): $a, b \leftarrow \mathcal{R}_q$

sk $:= s \leftarrow \$\, \mathcal{R}_q$

Enc(*sk*, $m \in \mathcal{R}_q$)$\rightarrow$ ***c***

$a \leftarrow \$\, \mathcal{R}_q$ $e_{enc} \leftarrow \chi$

c $:= (-a \cdot sk + e_{enc} + \Delta m\,,\ a)$

Dec(*sk*, ***c***)$\rightarrow$ m

M := ***c***[0] +*sk* · ***c***[1]

m := [M/Δ]

For simplicity, the relinearization and PKE have been omitted. In practice,
Disclaimer: it is non-trivial to enable relinearization: use gadgets products

Let $\Delta \in \mathbb{Z}_q$ be a positive factor less than q.

If $\Delta > 2e$ and $\Delta m < q$, then the division and rounding remove the error.

We now have correctness again!

Ring learning with error-based FHE scheme
Adding error correction – message scaling

$$\mathcal{R}_q = \mathbb{Z}_q[X]/(X^N + 1)$$

FHE:

KeyGen$\rightarrow(sk, \cdot)$: $a, b \leftarrow \mathcal{R}_q$

$sk := s \leftarrow \$\, \mathcal{R}_q$

Enc$(sk, m \in \mathcal{R}_q) \rightarrow \boldsymbol{c}$

$a \leftarrow \$\, \mathcal{R}_q \; e_{enc} \leftarrow \chi$

$\boldsymbol{c} := (-a \cdot sk + e_{enc} + \Delta m\,,\; a)$

Dec$(sk, \boldsymbol{c}) \rightarrow$ m

$M := \boldsymbol{c}[0] + sk \cdot \boldsymbol{c}[1]$

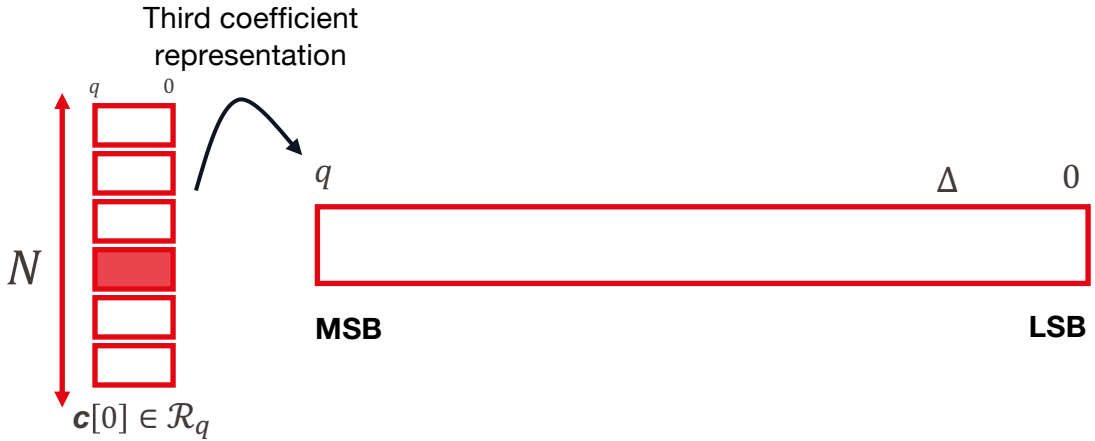
$m := [M/\Delta]$

For simplicity, the relinearization and PKE have been omitted. In practice, it is non-trivial to enable relinearization: use gadgets products

Disclaimer:

Let us look at the **impact** of the scale Δ

Consider the following representation of the polynomial coefficients



Ring learning with error-based FHE scheme
Adding error correction – message scaling

$$\mathcal{R}_q = \mathbb{Z}_q[X]/(X^N + 1)$$

FHE:

KeyGen$\rightarrow$(sk, $\cdot$): $a, b \leftarrow \mathcal{R}_q$

$sk := s \leftarrow \$\, \mathcal{R}_q$

Enc(sk, $m \in \mathcal{R}_q$)$\rightarrow$ $\boldsymbol{c}$

$a \leftarrow \$\, \mathcal{R}_q$ $e_{enc} \leftarrow \chi$

$\boldsymbol{c} := (-a \cdot sk + e_{enc} + \Delta m,\ a)$

Dec(sk, $\boldsymbol{c}$)$\rightarrow$ m

M := $\boldsymbol{c}$[0] +sk $\cdot$ $\boldsymbol{c}$[1]

m := [M/Δ]

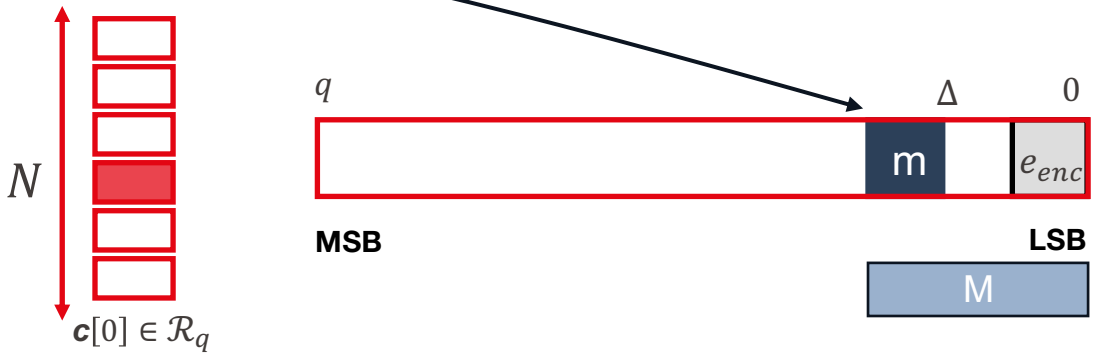
For simplicity, the relinearization and PKE have been omitted. In practice, it is non-trivial to enable relinearization: use gadgets products

Disclaimer:

Let us look at the **impact** of the scale

We display the scaled message *m* and the encryption noise e_{enc}

Third coefficient representation



Ring learning with error-based FHE scheme
Adding error correction – message scaling

$$\mathcal{R}_q = \mathbb{Z}_q[X]/(X^N + 1)$$

FHE:

KeyGen$\rightarrow$(sk, $\cdot$): $a, b \leftarrow \mathcal{R}_q$

$sk := s \leftarrow \$\, \mathcal{R}_q$

Enc(sk, $m \in \mathcal{R}_q$)$\rightarrow$ $\boldsymbol{c}$

$a \leftarrow \$\, \mathcal{R}_q$ $e_{enc} \leftarrow \chi$

$\boldsymbol{c} := (-a \cdot sk + e_{enc} + \Delta m\,,\ \ a)$

Dec(sk, $\boldsymbol{c}$)$\rightarrow$ m

M := $\boldsymbol{c}$[0] +$sk \cdot \boldsymbol{c}$[1]

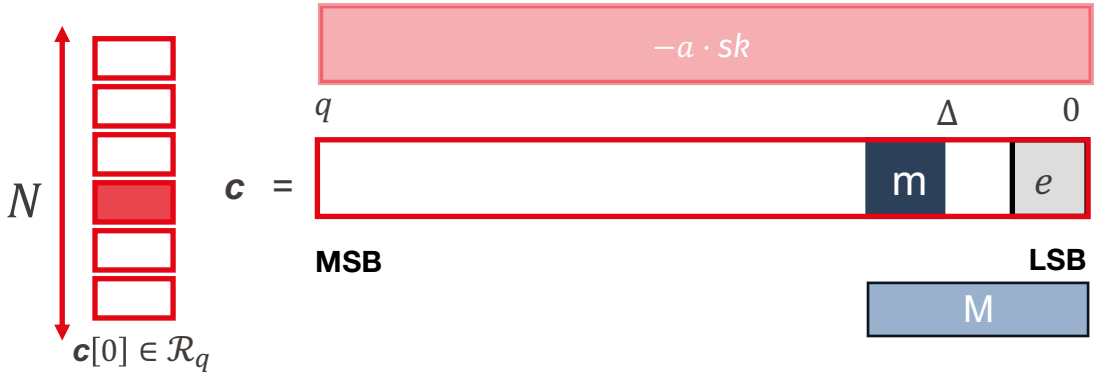
m := [M/Δ]

For simplicity, the relinearization and PKE have been omitted. In practice,
Disclaimer: it is non-trivial to enable relinearization: use gadgets products

Let us look at the **impact** of the scale

We now add the masking

Third coefficient representation



Ring learning with error-based FHE scheme

Adding error correction – message scaling and Ops

$$\mathcal{R}_q = \mathbb{Z}_q[X]/(X^N + 1)$$

FHE:

KeyGen$\rightarrow(sk, \cdot)$: $a, b \leftarrow \mathcal{R}_q$

$sk := s \leftarrow \$\, \mathcal{R}_q$

Enc$(sk, m \in \mathcal{R}_q) \rightarrow \boldsymbol{c}$

$a \leftarrow \$\, \mathcal{R}_q \; e_{enc} \leftarrow \chi$

$\boldsymbol{c} := (-a \cdot sk + e_{enc} + \Delta m\,,\; a)$

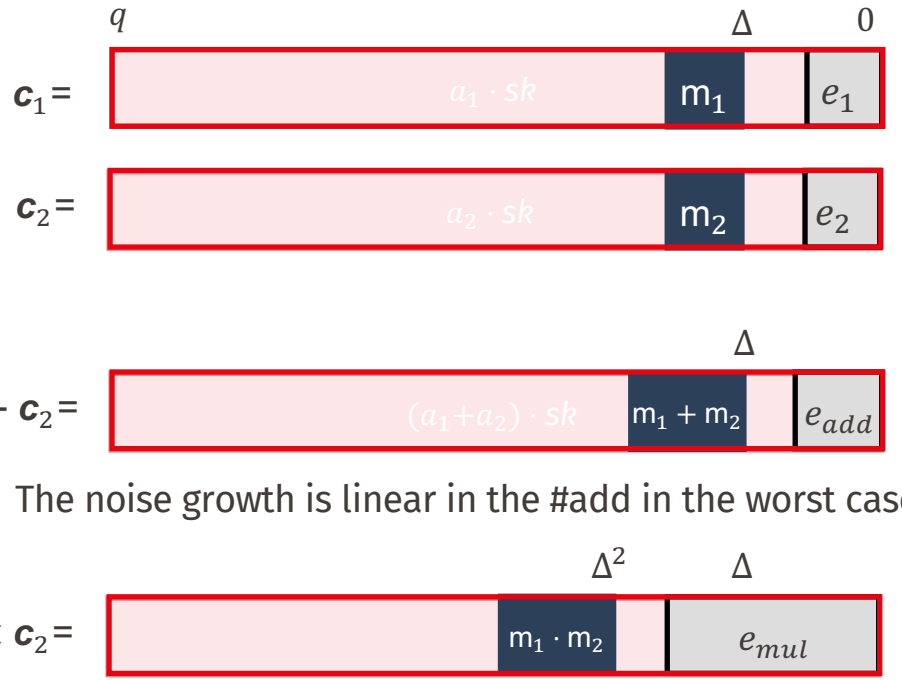
Dec$(sk, \boldsymbol{c}) \rightarrow$ m

M := $\boldsymbol{c}[0] + sk \cdot \boldsymbol{c}[1]$

m := $\lfloor M/\Delta \rceil$

For simplicity, the relinearization and PKE have been omitted. In practice, it is non-trivial to enable relinearization: use gadgets products

Disclaimer:



The noise growth is linear in the #add in the worst case.

The noise growth is quadratically in the #mult in the worst case.
Output message scaled by Δ^2

Ring learning with error-based FHE scheme

Adding error correction – message scaling and Ops

$$\mathcal{R}_q = \mathbb{Z}_q[X]/(X^N + 1)$$

FHE:

KeyGen$\rightarrow(sk, \cdot)$: $a, b \leftarrow \mathcal{R}_q$

$sk := s \leftarrow \$\, \mathcal{R}_q$

Enc$(sk, m \in \mathcal{R}) \rightarrow \boldsymbol{c}$

$a \leftarrow \$\, \mathcal{R}_q \; e_{enc} \leftarrow \chi$

$\boldsymbol{c} := (-a \cdot sk + e_{enc} + \Delta m,\ a)$

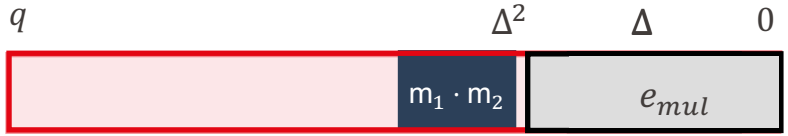
Dec$(sk, \boldsymbol{c}) \rightarrow$ m

M := $\boldsymbol{c}[0] + sk \cdot \boldsymbol{c}[1]$

m := $[M/\Delta]$

For simplicity, the relinearization and PKE have been omitted. In practice,
Disclaimer: it is non-trivial to enable relinearization: use gadgets products

$\boldsymbol{c_1} \times \boldsymbol{c_2} =$



What can we do? – Noise management

Ring learning with error-based FHE scheme
Adding error correction – message scaling and Ops

$$\mathcal{R}_q = \mathbb{Z}_q[X]/(X^N + 1)$$

FHE:

KeyGen$\rightarrow$(*sk*, ·): $a, b \leftarrow \mathcal{R}_q$

sk $:= s \leftarrow \$\, \mathcal{R}_q$

Enc(*sk*, $m \in \mathcal{R}_t$)$\rightarrow$ ***c***

$a \leftarrow \$\, \mathcal{R}_q$ $e_{enc} \leftarrow \chi$

c $:= (-a \cdot sk + e_{enc} + \Delta m\,,\ \ a)$

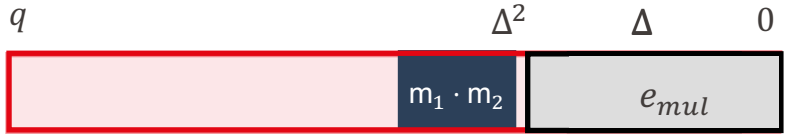
Dec(*sk*, ***c***)$\rightarrow$ m

M := ***c***[0] +*sk* · ***c***[1]

m := [M/Δ]

For simplicity, the relinearization and PKE have been omitted. In practice,
Disclaimer: it is non-trivial to enable relinearization: use gadgets products

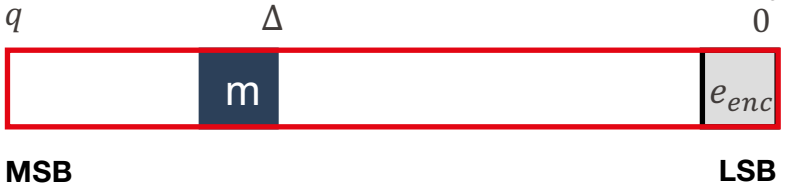
$\boldsymbol{c_1 \times c_2} =$



What can we do? – Noise management

1. **Select *Δ* wisely:**

Encrypt messages in $\mathcal{R}_t$, $t \ll q$. Set $\Delta = \lfloor \frac{q}{t} \rfloor$



2. **Rescale** from Δ^2 needed (multiply by $\frac{t}{q}$ in $\mathcal{R}$)

3. **Perform noise refresh:** Bootstrapping

Ring learning with error-based FHE scheme
Summary

Correctness – Yes

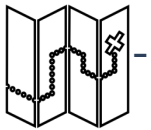
Evaluation – Additions and Multiplications

Compactness – with **relinearization**

Security – RLWE assumption

Note:

- Modulus q is large: **decompose** it in smaller primes (Chinese remainder Theorem)
- Plaintext space is $\mathcal{R}_t$: **batch** N values in $\mathbb{Z}_t$
- By selecting appropriate parameters, enable **fast multiplication** and **SIMD**



Amortize operations!
Pack as much as possible

$$\mathcal{R}_q = \mathbb{Z}_q[X]/(X^N + 1)$$

FHE:

KeyGen$\rightarrow$(*sk*, ***pk***, ***rlk***):
$sk := s \leftarrow \$\, \mathcal{R}_q$
rlk = **GenRLK**(*sk*) ***rlk*** = **GenPK**(*sk*)

Enc(*sk*, $m \in \mathcal{R}_t$)$\rightarrow$ ***c***
$a \leftarrow \$\, \mathcal{R}_q$ $e_{enc} \leftarrow \chi$, $\Delta = \lfloor \frac{q}{t} \rfloor$
$\boldsymbol{c} := (-a \cdot sk + e_{enc} + \Delta m\,,\ \ a)$

Relin(***rlk***, ***c***) $\rightarrow$***c***'

Dec(*sk*, ***c***)$\rightarrow$*m*

Add($\boldsymbol{c}_1$, $\boldsymbol{c}_2$) = $\boldsymbol{c}_1 + \boldsymbol{c}_2$

Mul($\boldsymbol{c}_1$, $\boldsymbol{c}_2$) = **Relin**(***rlk***,$\mathcal{R}(\boldsymbol{c}_1 \times \boldsymbol{c}_2)/\Delta$)

SIMD: single input multiple data

FHE in Practice
Challenges

1. ***Selecting cryptographic parameters***

Interdependencies between $N, q, t,$ and χ
Relies on estimators to assess the hardness

2. ***Circuit definition***

How to represent functions into circuits
Minimize the multiplicative depth
Optimize the costly operations (bootstrap, rescale, etc.)

3. ***Bearing with the costs***

FHE induces both computation and communication overhead
Not all plaintexts space can be easily handled

Naïve operation

$(((m_1 \cdot m_2) \cdot m_3) \cdot m_4)$

$+e$, $+e^2$, $+e^3$

Optimization

$((m_1 \cdot m_2) \cdot (m_3 \cdot m_4))$

$+e$, $+e$, $+e^2$

FHE in Practice
Parameterization

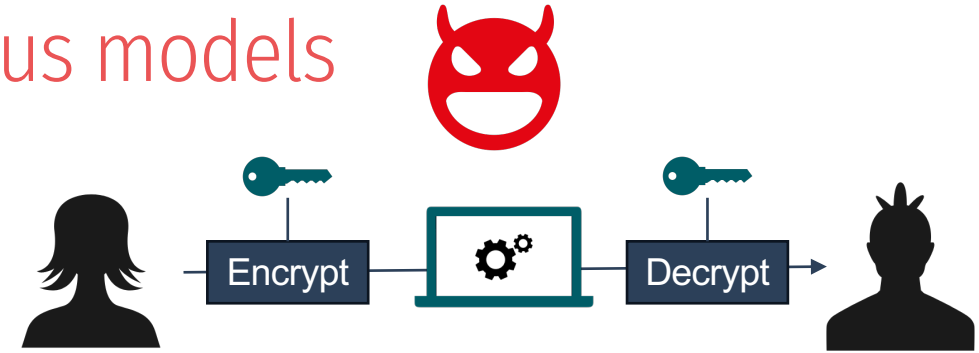
- **Dimension N:** between 2^{11} and 2^{16} — polynomial size (keys, ciphertexts)
- **Ciphertext space:** $\mathcal{R}_q$ with q 100s of bits — coefficient size of ciphertexts
- **Message space:** t application dependent — can be a small as 20 bits

Example: compute Squared Distance between two 2D

- *log N = 13, log q = 218, log t = 16*
- One ciphertext: 450kB
- **Setup:** 7.4 ms
- **Encode + Enc 1 vector:** 2.7 ms
- **Eval:** 7 ms
- **Dec + Decode:** 2 ms

But costs can quickly go up! For example for log N = 15 and depth 16, you might need >100ms for one multiplication

FHE in Practice
Extension to malicious models



Malicious threat model?

So far, the evaluator can only **evaluate** the function on the input it receives.

In reality, evaluator could be:

- **Honest but Curious:** Parties will follow the HE protocol honestly, but try to learn as much as possible from the messages they receive
- **Malicious:** Parties can arbitrarily deviate from the HE protocol to learn as much as possible

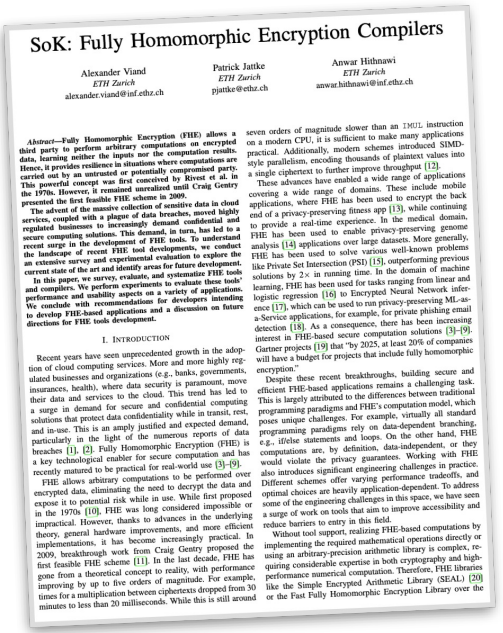
⇒ **Requires heavy machinery e.g., Zero-Knowledge Proofs**

Homomorphic Encryption
Other resources

Survey on Fully Homomorphic Encryption, Theory, and Applications

Chiara Marcolla, Victor Sucasas, *Member, IEEE*, Marc Manzano, Riccardo Bassoli, *Member, IEEE*, Frank H.P. Fitzek, *Senior Member, IEEE* and Najwa Aaraj

C. Marcolla et al. "Survey on fully homomorphic encryption, theory, and applications." *Proc. of the IEEE* 2022

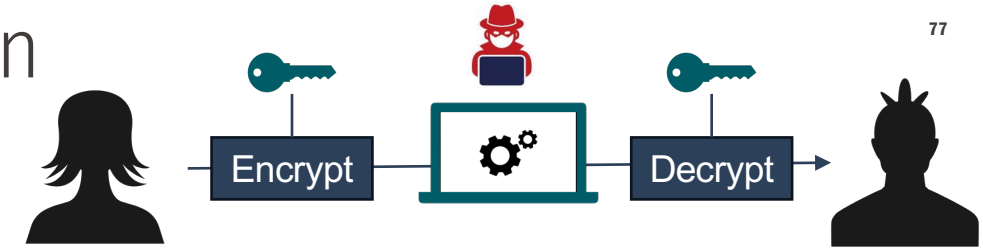


SoK: Fully Homomorphic Encryption Compilers

Alexander Viand, *ETH Zurich*, alexander.viand@inf.ethz.ch
Patrick Jattke, *ETH Zurich*, pjattke@ethz.ch
Anwar Hithnawi, *ETH Zurich*, anwar.hithnawi@inf.ethz.ch

A. Viand et al. "SoK: Fully homomorphic encryption compilers." *2021 IEEE SP*

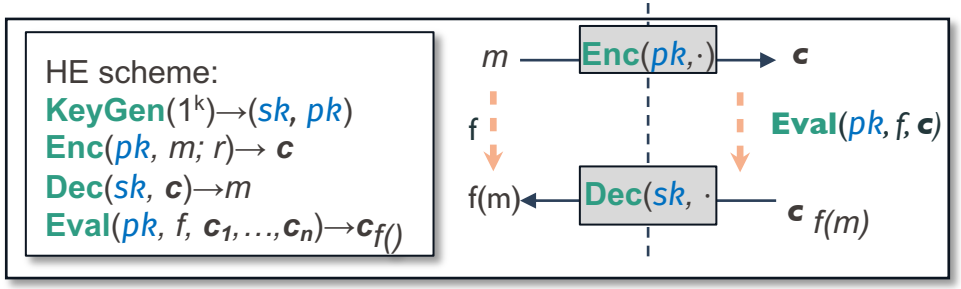
Homomorphic Encryption
Conclusion



Enables ***computation over encryption***

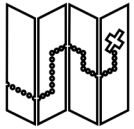
Has become increasingly ***practical***

Modern lattice-based schemes plausibly ***quantum resistant***

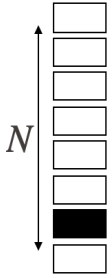


Challenges:

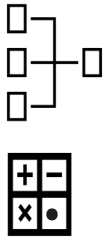
Parameters selection



Plaintext encoding



Amortization



Circuit design

Noise management

References
Software Libraries

- [HElib] https://github.com/shaih/HElib Halevi and Shoup
- [SEAL] https://www.microsoft.com/en-us/research/project/simple-encrypted-arithmetic-library/ SEAL
- [NFLlib] https://github.com/quarkslab/NFLlib French consortium
- [HEAAN] https://github.com/kimandrik/HEAAN Korean researchers
- [TFHE] https://tfhe.github.io/tfhe/ inpher + French researchers
- [PALISADE] https://git.njit.edu/palisade/PALISADE New Jersey Institute of Technology
- [cuHE] https://github.com/vernamlab/cuHE
- [Lattigo] https://github.com/tuneinsight/lattigo
- [OpenFHE] https://openfhe.org/
- [THFE-rs] https://docs.zama.ai/tfhe-rs