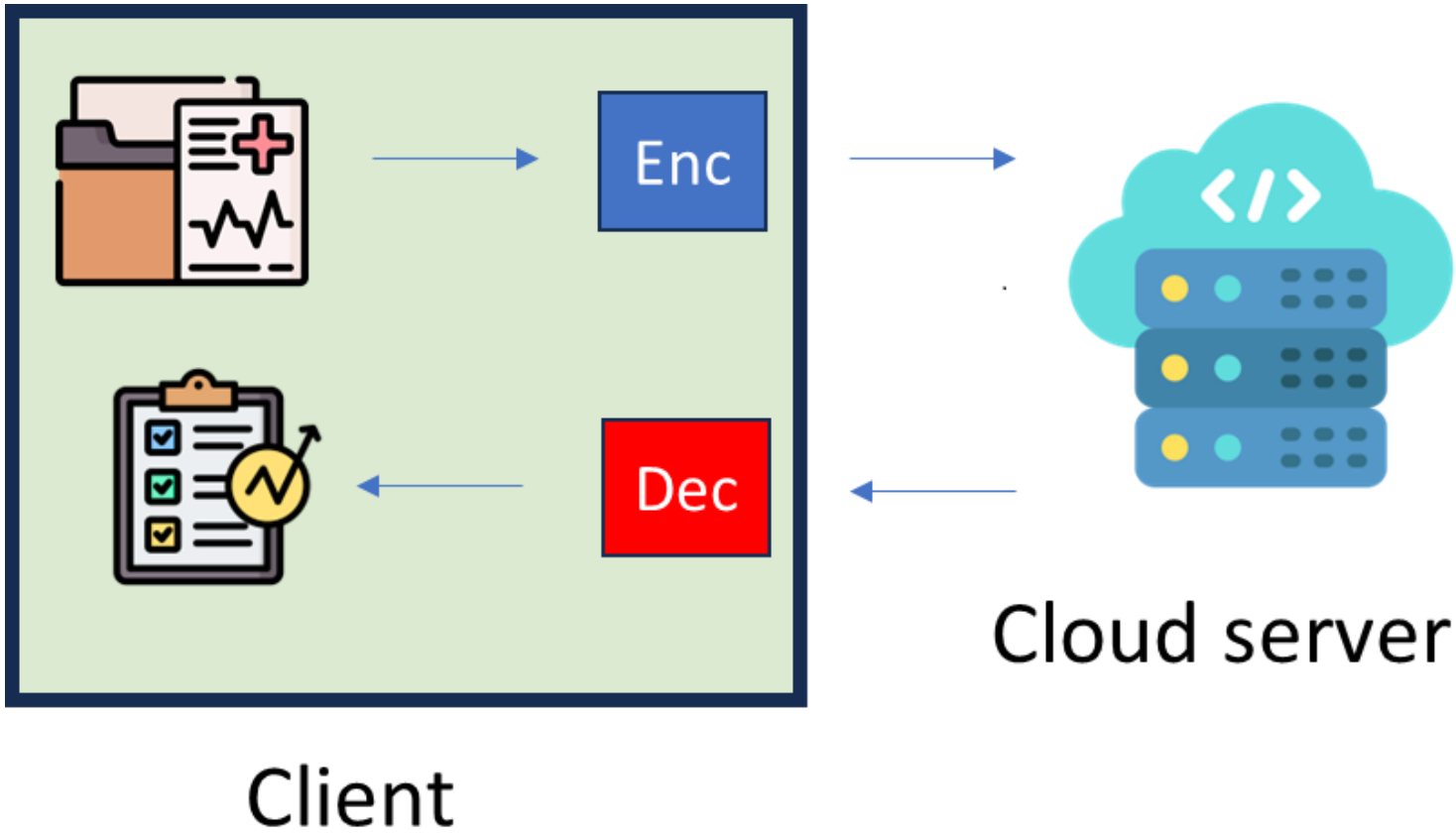
Homomorphic Encryption

Homomorphic encryption - Motivation

Outsource computation securely

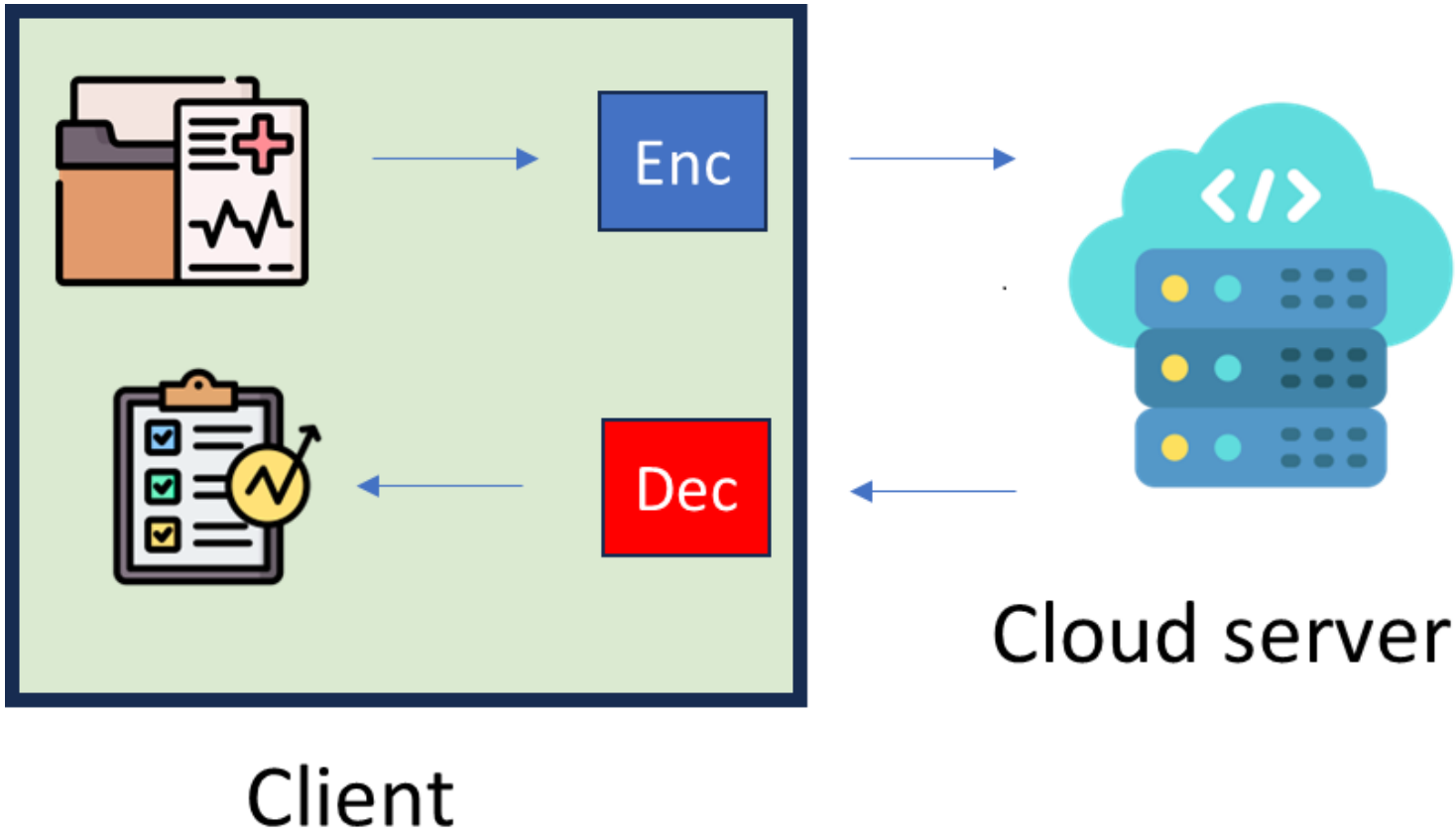
Dedicate expensive computations to servers without reveal data



Homomorphic encryption - Motivation

Outsource computation securely

Dedicate expensive computations to servers without reveal data



Homomorphic encryption - Application

- Secure machine learning and data analytics
- End-to-end internet encryption
- Encrypted facial recognition
- Encrypted Photo Filtering
- Private Smart Contracts
- Encrypted Health Predictions
- Encrypted Credit Card Approval

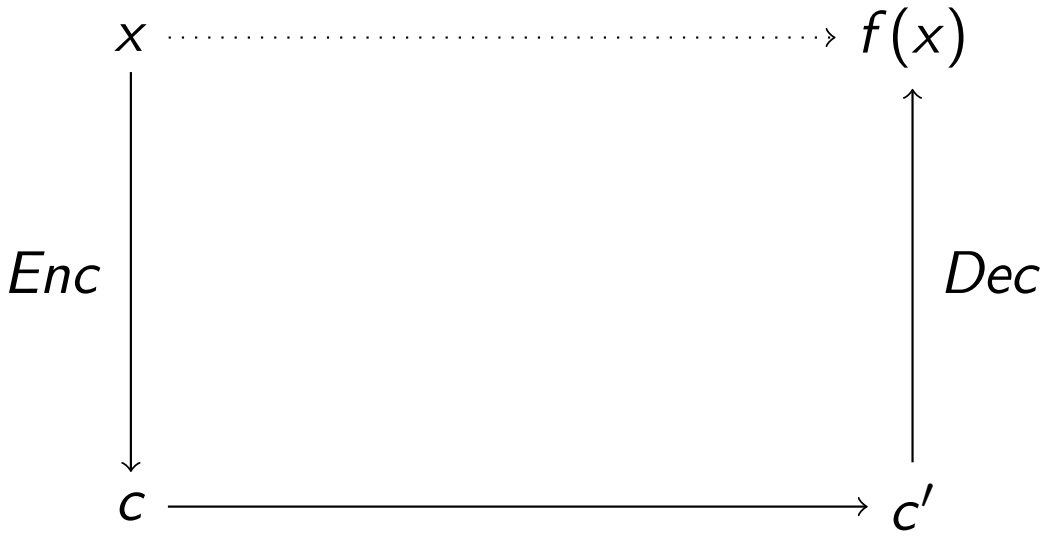
Homomorphic encryption - Framework

4-tuple of PPT algorithms (Gen, Enc, Dec, Eval)

- $(sk, ek) \leftarrow Gen(1^n)$.
  PPT Key generation algorithm generates a secret key as well as a (public) evaluation key.
- $c \leftarrow Enc(sk, m)$
  Encryption algorithm uses the secret key to encrypt message m.
- $c' \leftarrow Eval(ek, f, c)$
  Homomorphic evaluation algorithm uses the evaluation key to produce an "evaluated ciphertext" c'.
- $m \leftarrow Dec(sk, c)$
  Decryption algorithm uses the secret key to decrypt ciphertext c.

Correctness

$$Dec(sk, Eval(ek, f, Enc(x))) = f(x)$$



Types of HE

A function can be represented as either Boolean or arithmetic circuit. It is enough to compute the addition (XOR) and multiplication (AND) gate.

1. Partially homomorphic encryption: supports only one type of gate
2. Somewhat homomorphic encryption: supports both gates but only for subset of circuits
3. Leveled fully homomorphic encryption: supports both gates but for limited depth
4. Fully homomorphic encryption (FHE):

ElGamal

ElGamal scheme

- Parameters: cyclic group G of order q with generator $g \in G$.
- Secret key: $x \in G$
- Public key: (G, q, g, h)
- Encryption: $E(m) = (g^r, m \cdot h^r)$ for some $r \in \{0, \ldots, q-1\}$

The ElGamal encryption is partially homomorphic for multiplication:

- $E(m_i) = (g^{r_i}, m_i \cdot h^{r_i})$ for $i = 1, 2$,
- $E(m_1)E(m_2) = (g^{r_1} g^{r_2}, m_1 h^{r_1} m_2 h^{r_2}) = (g^{r_1+r_2}, m_1 m_2 h^{r_1+r_2}) = E(m_1 m_2)$ with $r = r_1 + r_2$.

Goldwasser-Micali

Quadratic residue

- Let $N = pq$ (p, q primes). Compute $x_p = y \mod p$ and $x_q = x \mod q$.
- If $x_p^{(p-1)/2} \equiv 1 \mod p$ and $x_q^{(q-1)/2} \equiv 1 \mod q$, then x is a quadratic residue $\mod N$

Goldwasser-Micali scheme

- Parameters: $n = pq$ (p, q primes), x-non-residue mod N.
- Public key: (x, n)
- Secret key: (p, q)
- Encryption (1 bit): $E(b) = x^b r^2 \mod n$ for some $r \in \{0 \ldots, n-1\}$.

Goldwasser-Micali is partially homomorphic for XOR:

- $E(b_1) = x^{b_1} r_1^2 \mod n$, $E(b_2) = x^{b_2} r_2^2 \mod n$
- $E(b_1)E(b_2) = x^{b_1+b_2}(r_1 r_2)^2 \mod n = E(b_1 \oplus b_2)$

Other PHEs

- Paillier: $E(m_1)E(m_2) = E(m_1 + m_2)$.
- RSA: $E(m_1)E(m_2) = E(m_1 m_2)$.
- Benaloh: $E(m_1)E(m_2) = E(m_1 + m_2 \mod c)$.
- ...

SWHE (SHE) over the integers

- Secret key: large odd integer n
- Public key: integers $\{x_i = q_i n + 2r_i\}_i$, $|r_i|$ (noise) is much smaller than n.
- Encrypt: message $m \in \{0, 1\}$. Choose a subset S of x_i's, and compute $c = \sum_{i \in S} x_i + m$.
- Decrypt: Take the LSB of ($c \mod n$).

Correctness

$c \mod n = \sum_{i \in S} x_i + m \mod n = \sum_{i \in S} q_i n + \sum_{i \in S} 2r_i + m \mod n = \sum_{i \in S} 2r_i + m$. It is clear, that the LSB is m.

Security

Reduction

If *apporoximate gcd* problem is hard, then the scheme is semantically secure.

Apporoximate GCD problem

Given many $x_i = q_i n + r_i$ (approx multiples of n), find n.

If we omit the terms r_i, then it is trivial to recover n using the Euclidean algorithm.

Addition

Is it possible to add encrypted messages?

- Let $c_i = E_K(m_i)$ for $i = 1, 2$ and $c = c_1 + c_2$.
- Then $c_i \mod n = 2r_i + m_i$ for some r_i.
- $c \mod n = (c_1 + c_2) \mod n = 2(r_1 + r_2) + m_1 + m_2 \mod n$.
- If $2(r_1 + r_2) + m_1 + m_2 < n$, then LSB of c is $m_1 + m_2$.

Remark

We can add elements until the noise (sum of r_i) is below n.

Multiplication

Is it possible to multiply encrypted messages?

- Let $c_i = E_K(m_i)$ for $i = 1, 2$ and $c = c_1 \cdot c_2$.
- $c_i \mod n = 2r_i + m_i$ for some r_i.
- $c \mod n = (c_1 \cdot c_2) \mod n = (2r_1 + m_1)(2r_2 + m_2) \mod n$.
- If $(2r_1 + m_1)(2r_2 + m_2) < n$ then the LSB of c is $m_1 \cdot m_2$.

Remark

We can multiply elements until the noise is below n.

Function

Can we compute any f function using the SWHE algorithm?

- We get the correct result if the noise is below the threshold (n).
- Addition: increase the noise linearly
- Multiplication: increase the noise exponentially

Function

Can we compute any f function using the SWHE algorithm?

- We get the correct result if the noise is below the threshold (n).
- Addition: increase the noise linearly
- Multiplication: increase the noise exponentially

Summary

- Good news: For a function f if we choose n large enough, it is possible to compute f.
- Bad news: If f has many multiplication, n has to be too large.

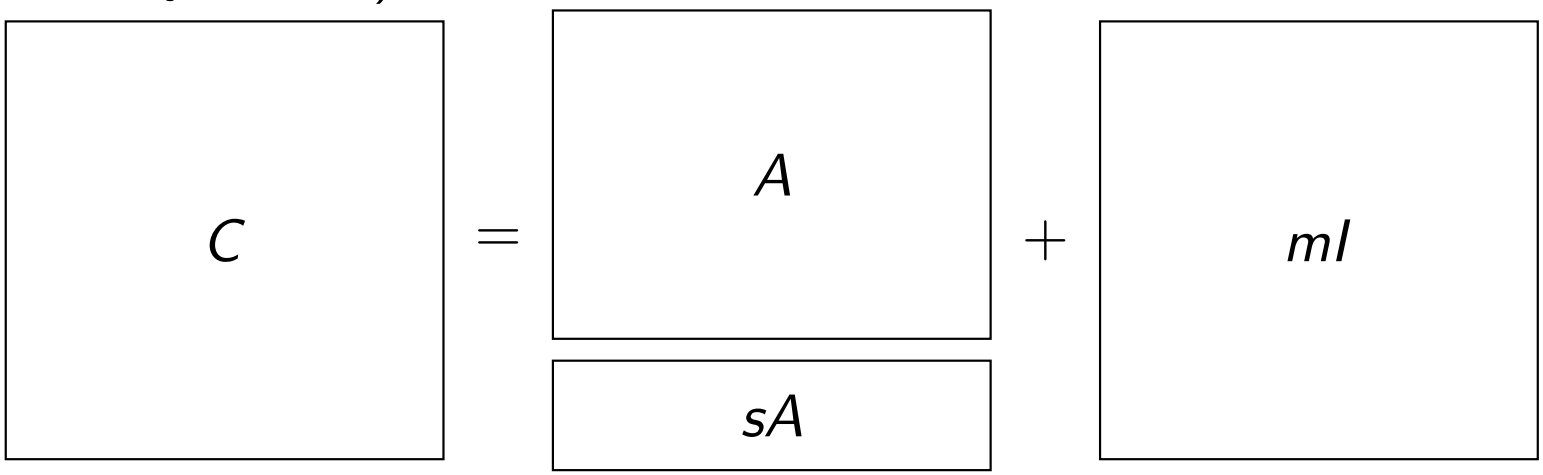
.

FHE

- Problem with SWHEs: The parameter size shouldn't depend on the function.
- Idea: Reduce the noise after every addition/multiplication.
- How to do this while the data is still encrypted?
- Bootstrapping - it was very slow, but it become more and more efficient over the years.

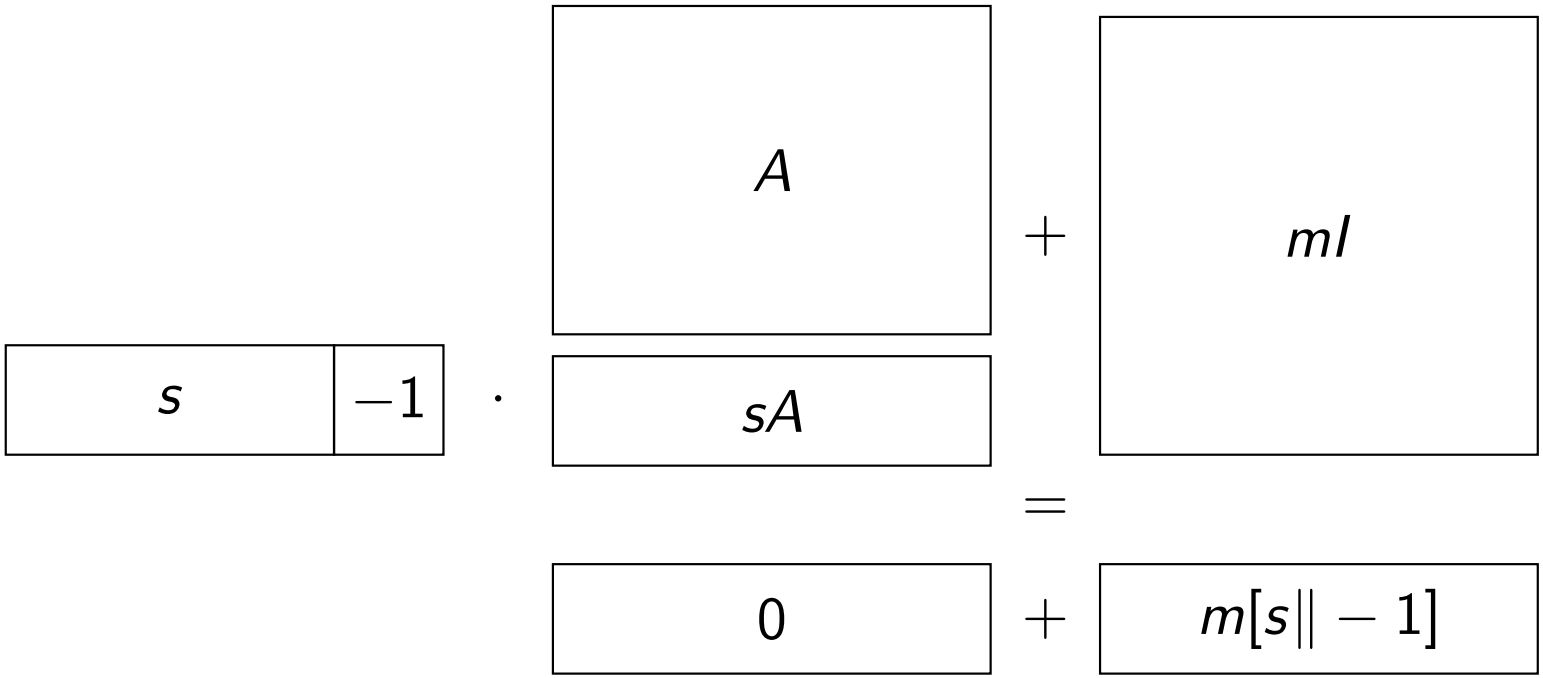
Idea of modern FHE

- Parameters: A random $(n+1) \times n$ matrix over $\mathbb{Z}_q$.
- Private key: $s \in \mathbb{Z}_q^n$
- Encryption: message $m \in \{0,1\}$, $C = \begin{bmatrix} A \\ sA \end{bmatrix} + mI$ (I is the identity matrix)



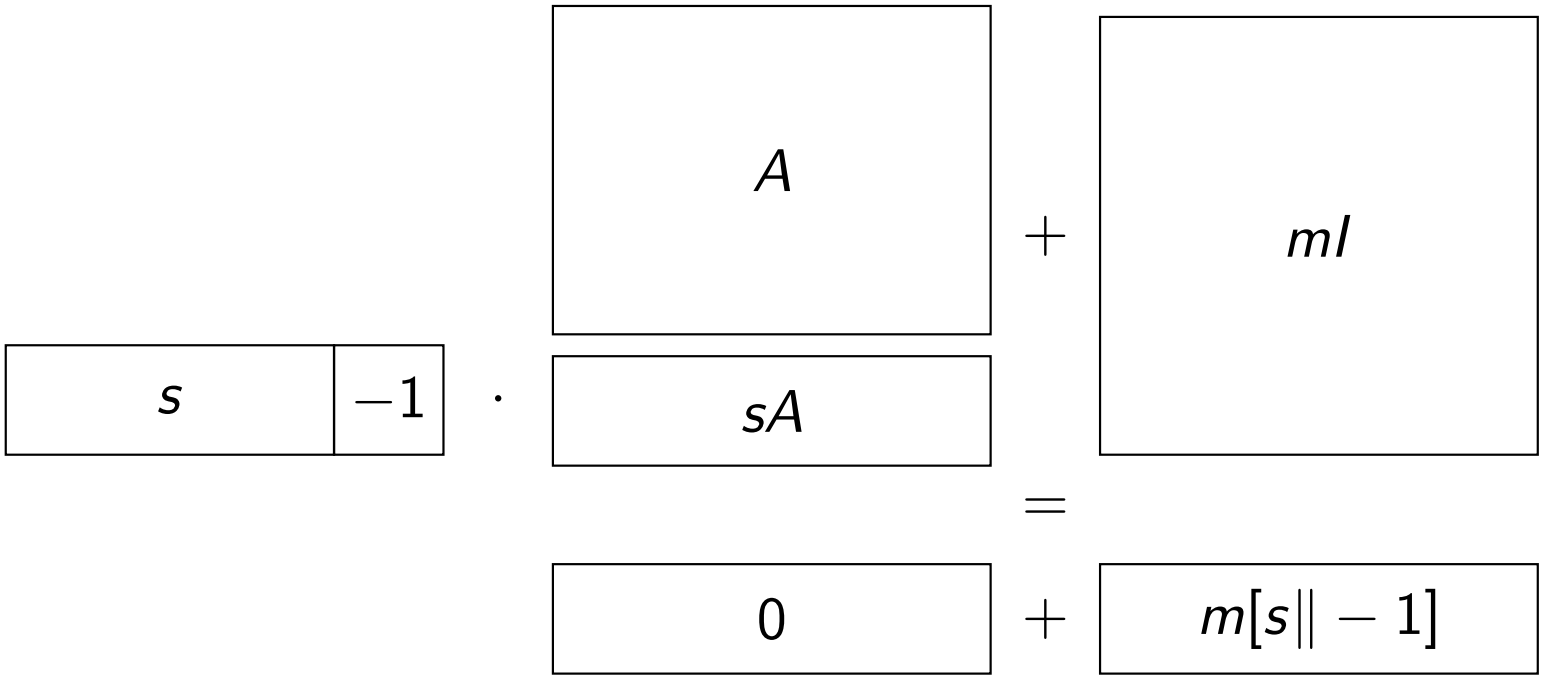
Idea of modern FHE

- Decryption: $[s\| -1]C = m[s\| -1]$



Idea of modern FHE

- Decryption: $[s\| -1]C = m[s\| -1]$



INSECURE!

- $[s\| -1]$ is the eigenvector of C with eigenvalue m.
- $m = 0$ or $m = 1$, therefore it is easy to compute the eigenvector of the matrix (it is equivalent with solving a linear equation).

Homomorphic properties

Suppose that $tC_i = m_i t$ for a vector t ($t = [s\| - 1]$. Then

Homomorphic addition

$$t(C_1 + C_2) = (m_1 + m_2)t$$

Homomorphic multiplication

$$t(C_1 \cdot C_2) = (tC_1)C_2 = m_1 tC_2 = m_1 m_2 t$$

- The encryption satisfies the homomorphism for both addition and multiplication
- Still not secure

Scheme with noise

- Parameters: A random $(n+1) \times n$ matrix over $\mathbb{Z}_q$.
- Private key: $s \in \mathbb{Z}_q^n$
- Encryption: message $m \in \{0,1\}$, e, is a small random vector, $C = \begin{bmatrix} A \\ sA + e \end{bmatrix} + mI$ (I is the identity matrix)
- Decryption: $[s\| - 1]C \approx m[s\| - 1]$

Scheme with noise

- Parameters: A random $(n+1) \times n$ matrix over $\mathbb{Z}_q$.
- Private key: $s \in \mathbb{Z}_q^n$
- Encryption: message $m \in \{0,1\}$, e, is a small random vector, $C = \begin{bmatrix} A \\ sA + e \end{bmatrix} + mI$ (I is the identity matrix)
- Decryption: $[s\| - 1]C \approx m[s\| - 1]$

Security

CPA-secure by LWE (Learning with errors)

Homomorphic properties

Suppose that $tC_i = m_i t + e_i$ for a vector t ($t = [s\| - 1]$. Then

Homomorphic addition

$$t(C_1 + C_2) = (m_1 + m_2)t + (e_1 + e_2)$$

Homomorphic multiplication

$$t(C_1 \cdot C_2) = (tC_1)C_2 = (m_1 t + e_1)C_2 = m_1 m_2 t + m_1 e_2 + e_1 C_2$$

Homomorphic properties

Suppose that $tC_i = m_i t + e_i$ for a vector t ($t = [s\| - 1]$. Then

Homomorphic addition

$$t(C_1 + C_2) = (m_1 + m_2)t + (e_1 + e_2)$$

Homomorphic multiplication

$$t(C_1 \cdot C_2) = (tC_1)C_2 = (m_1 t + e_1)C_2 = m_1 m_2 t + m_1 e_2 + e_1 C_2$$

- The noise grows slowly for addition: $e_1 + e_2$
- The noise grows more for multiplication: $m_1 e_2 + e_1 C_2$. If C_2 is small, the noise is small too.
- Still needs bootstrapping to reduce the noise after certain rounds.

Summary

- FHE is a very important cryptographic protocol for privacy.
- Constructions are based on SWHE scheme with bootstrapping.
- Early years it was very inefficient.
- Significant progress in recent years.
- More information: https://www.zama.ai/introduction-to-homomorphic-encryption

Commutative Encryption

Definition

An encryption E is commutative, if $E_{K_1}(E_{K_2}(m)) = E_{K_2}(E_{K_1}(m))$ for every K_1, K_2 keys.

- If encrypt a message twice with different keys, the order does not matter.
- OTP and stream ciphers are commutative:
  $m \oplus G(K_1, r_1) \oplus G(K_2, r_2) = m \oplus G(K_2, r_2) \oplus G(K_1, r_1)$
- AES is not commutative.

Mental Poker

Alice and Bob want to play poker online, but do not trust a third party (or online games).

Online poker

- Both Alice and Bob end up with 5 cards each that are randomly selected, and the other has no information about the cards.
- After revealing the cards, both of them can check that the other haven't cheated (they revealed their actual cards).

How do we achieve it?

Mental poker algorithm

Let E be a CPA secure commutative encryption. Each card is represented with a message $m_1, \ldots, m_{52}$ (for example, *two of clubs*, *ace of spades*, etc.).

1. Bob chooses an encryption key K_B and encrypts every card with it: $c_i = E_{K_B}(m_i)$.
2. Bob permutes (shuffles) the encrypted cards, then sends them to Alice.
3. Alice chooses 5 cards for Bob and sends them to Bob. Bob decrypts the cards.
4. Alice chooses 5 cards for herself, encrypts them with her key K_A, and sends the doubly encrypted cards to Bob.
5. Bob decrypts his encryption from Alice's cards and sends them back to Alice. Now Alice can decrypt her cards.
6. After the game, both Alice and Bob reveal their key so that the other can check that they played fairly.

If Alice or Bob needs to draw more cards, they repeat the steps 3-5.

Mental poker analysis

- Correctness: Alice correctly decrypts her cards because $D_{K_A}(D_{K_B}(E_{K_A}(E_{K_B}(m_i)))) = m_i$ by commutativity.
- Security: Both Alice and Bob see only the others cards while they are encrypted.
- Randomness: Alice chooses all the cards, but she has no information about the order, so she chooses randomly.
- Soundness: If Alice (the same for Bob) wants to change her card m_i to m_j, then Alice has to find a key K'_A s.t. $E_{K'_A}(m_i) = E_{K_A}(m_j)$ (because Bob sees $E_{K_A}(m_j)$). If Alice can do this, she can perform a CPA attack against E.