ZAMA

Introduction to FHE and the TFHE scheme

Ilaria Chillotti

Workshop on Foundations and
Applications of Lattice-based Cryptography
ICMS, Edinburgh

July 26, 2022

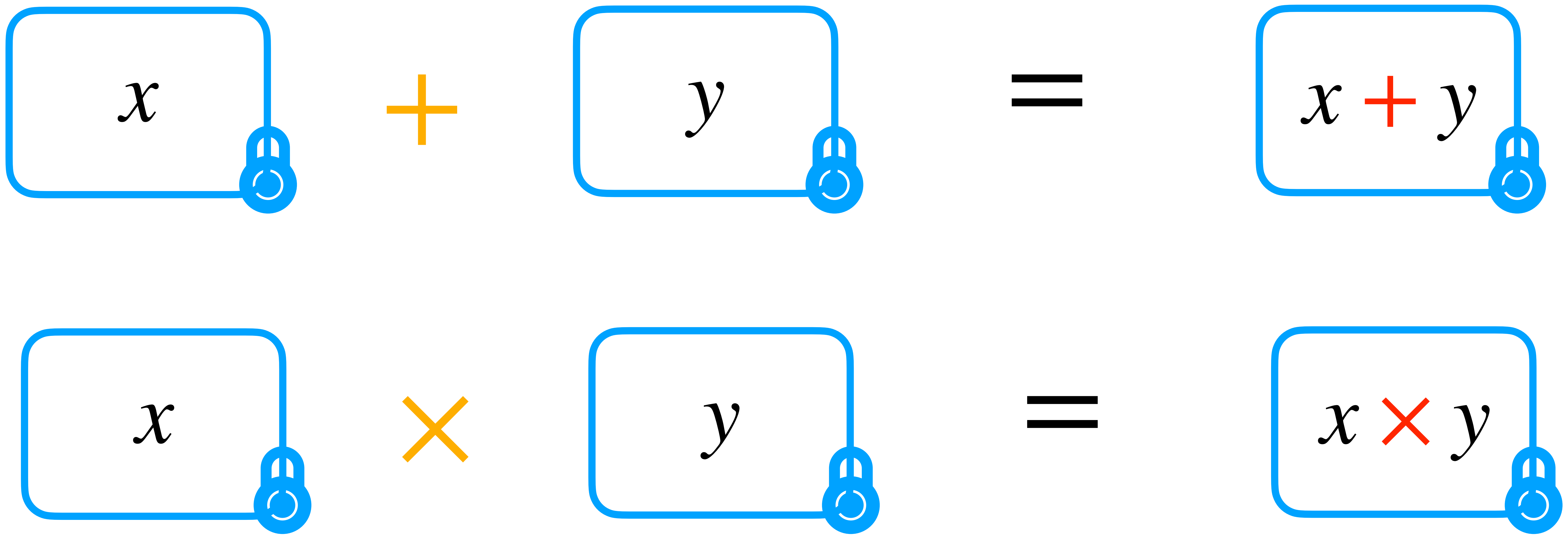
Overview

- What is FHE?
- A little bit of history
- FHE schemes based on LWE
- TFHE ciphertexts and operations
- TFHE Bootstrapping
- Implementations and applications

Overview

- **What is FHE?**
- A little bit of history
- FHE schemes based on LWE
- TFHE ciphertexts and operations
- TFHE Bootstrapping
- Implementations and applications

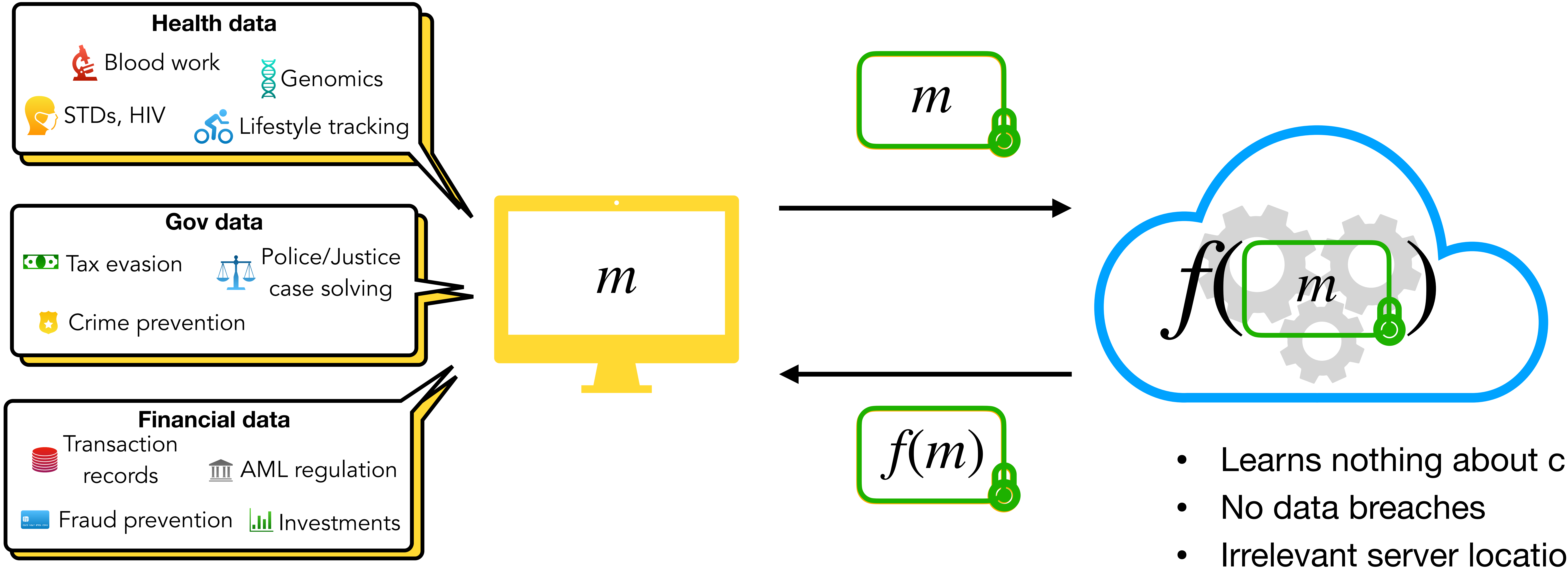
What is FHE?



FHE = Computations over encrypted messages

- Possibly any function ("Fully")
- Bit, integer, real messages
- Secret key and public key encryption

Where FHE Could Be Used IRL?



- Learns nothing about client data
- No data breaches
- Irrelevant server location

Overview

- What is FHE?
- **A little bit of history**
- FHE schemes based on LWE
- TFHE ciphertexts and operations
- TFHE Bootstrapping
- Implementations and applications

A little bit of history 📚

1978 - Rivest, Adleman and Dertouzos: talk about privacy homomorphisms

???

What happened in the meantime?

2009 - Gentry: first fully homomorphic encrypton scheme

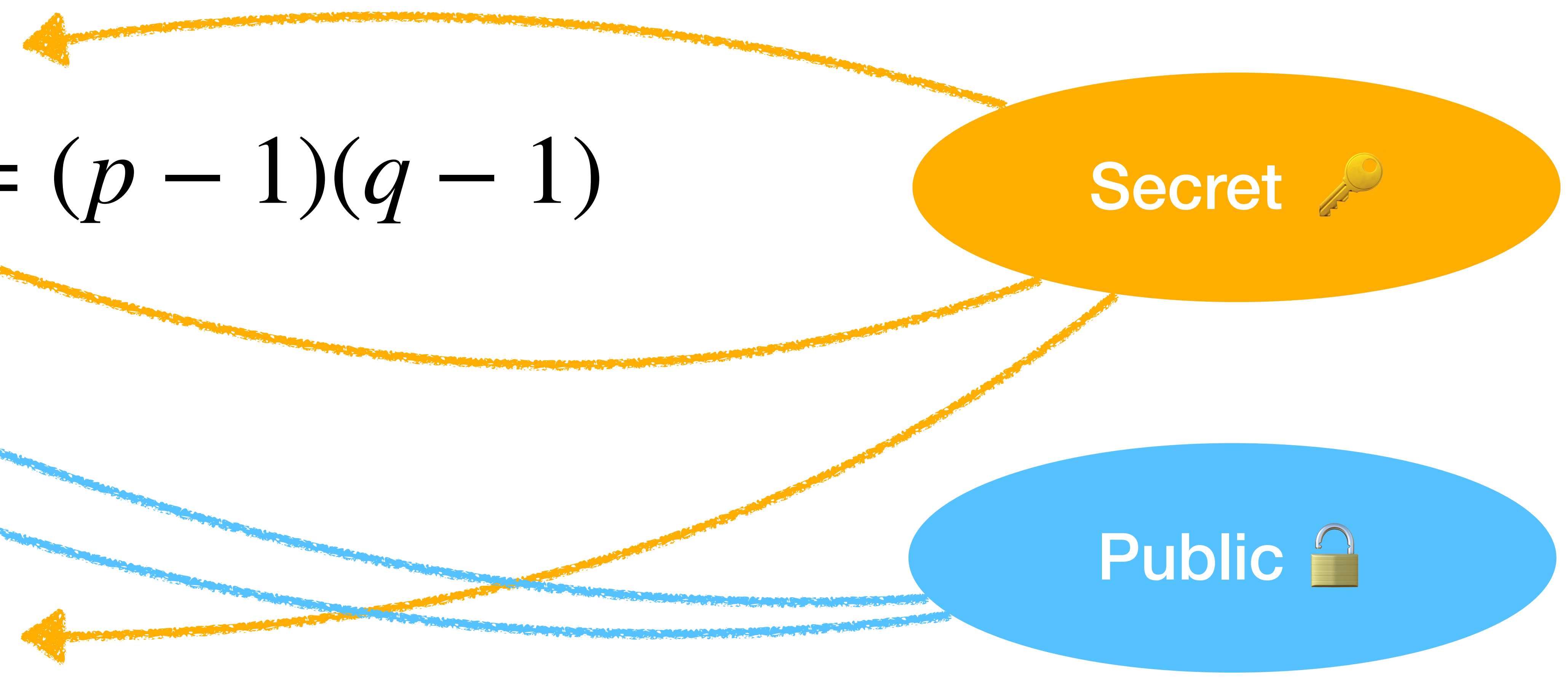
Partially homomorphic 🔏

An example: RSA

Rivest, Shamir, Adleman - 1977

Security factoring problem

- Select two large primes: $p \neq q$
- Compute: $n = p \cdot q$ and $\varphi(n) = (p-1)(q-1)$
- Chose: e such that
  - $1 < e < \varphi(n)$
  - e and $\varphi(n)$ coprimes
- Compute: $d = e^{-1} \mod \varphi(n)$



Encryption: $m \longmapsto c = m^e \mod n$

Decryption: $c \longmapsto m = c^d \mod n$

Multiplicative Homomorphic

$$\left.\begin{aligned} c_1 &= m_1^e \mod n \\ c_2 &= m_2^e \mod n \end{aligned}\right\} \quad c_1 \cdot c_2 = (m_1 \cdot m_2)^e \mod n$$

A little bit of history 📚

1978 - Rivest, Adleman and Dertouzos: talk about privacy homomorphisms

Partially Homomorphic: RSA, ElGamal, Paillier, Goldwasser-Micali, …
Somewhat Homomorphic: Boneh, Goh and Nissim (2005), …
Leveled Homomorphic: …

2009 - Gentry: first fully homomorphic encrypton scheme

A world full of noise 🌡️

An example: DGHV

2010 - Van Dijk, Gentry, Halevi, Vaikuntanathan

Security
Approximate GCD problem

- $m \in \{0,1\}$ message
- $p \in \mathbb{Z}$ large odd secret
- $q \in \mathbb{Z}$ way larger than p
- $e \in \mathbb{Z}$ way smaller than p, called *noise*

Encryption: $m \longmapsto c = pq + 2e + m$

Decryption: $c \longmapsto m = (c \mod p) \mod 2$

A world full of noise 🌡️

2010 - Van Dijk, Gentry, Halevi, Vaikuntanathan

An example: DGHV

$$c_1 = pq_1 + 2e_1 + m_1 \qquad c_2 = pq_2 + 2e_2 + m_2$$

Homomorphic addition (XOR)

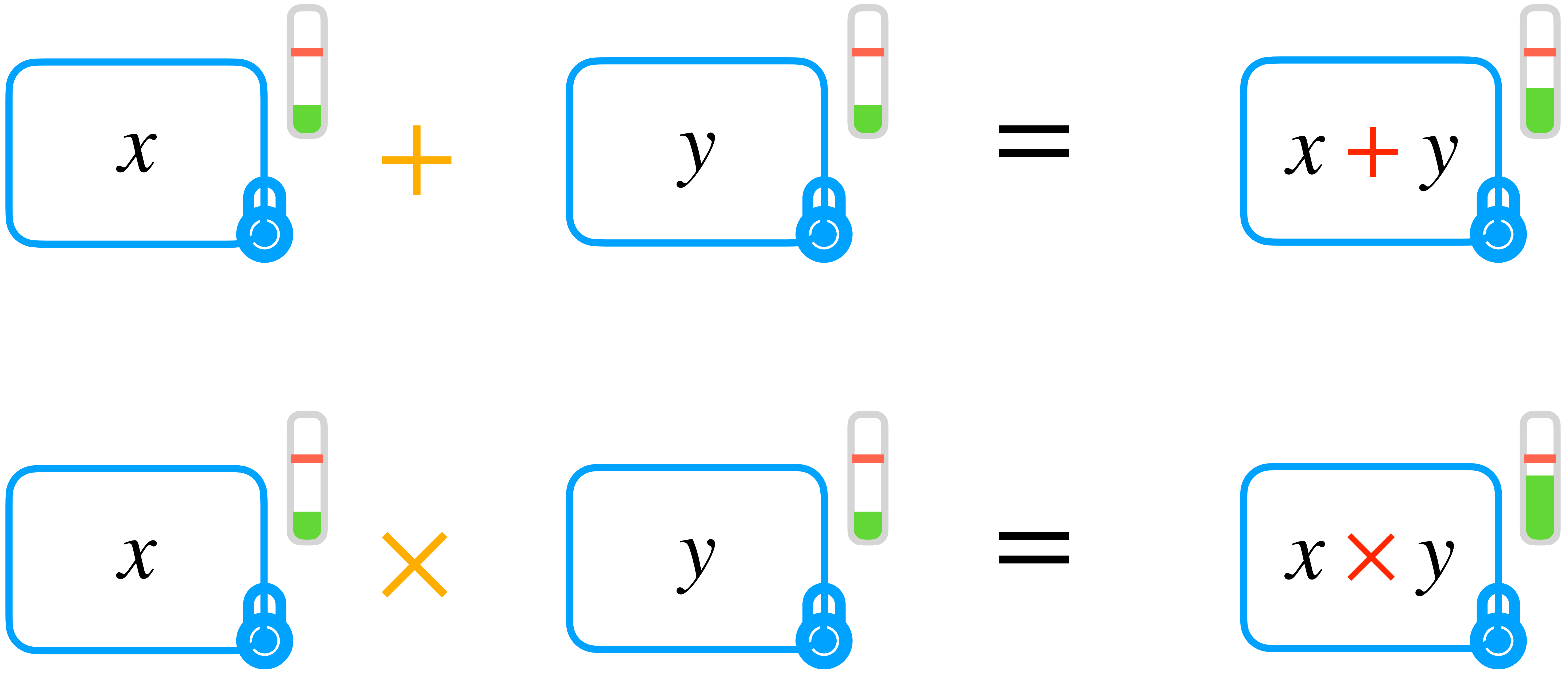
$$c_1 + c_2 = p \cdot (q_1 + q_2) + 2 \cdot (e_1 + e_2) + m_1 + m_2$$

Homomorphic multiplication (AND)

$$c_1 \cdot c_2 = p \cdot (pq_1q_2 + \ldots) + 2 \cdot (2e_1e_2 + \ldots) + m_1m_2$$

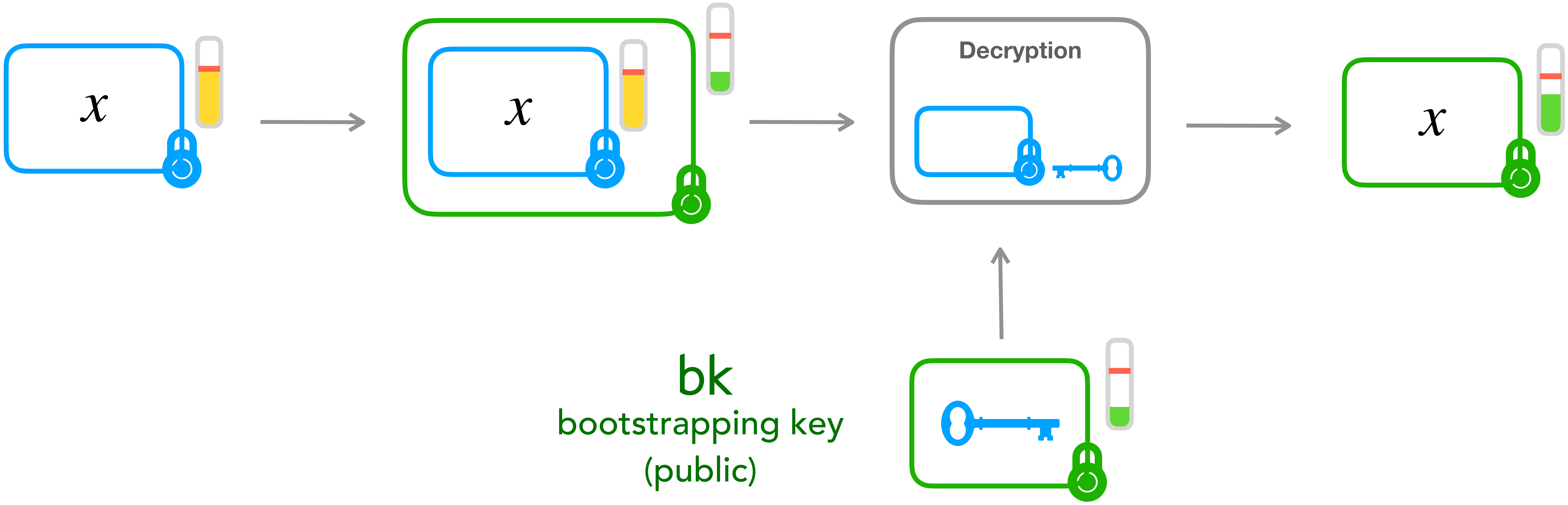
Noise grows too much 🌡️ ⇒ decryption incorrect 🚨

Noise 🌡️

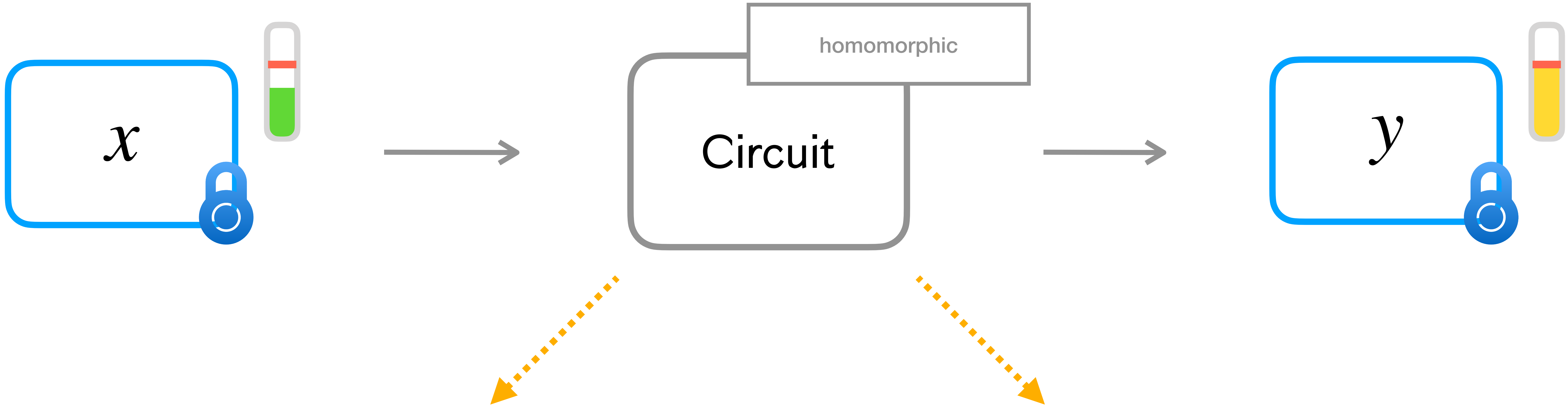


Noise grows too much 🌡️ ⇒ decryption incorrect 🚨

Bootstrapping [Gen09]



To bootstrap or not to bootstrap?



Your circuit is **small** and **known**

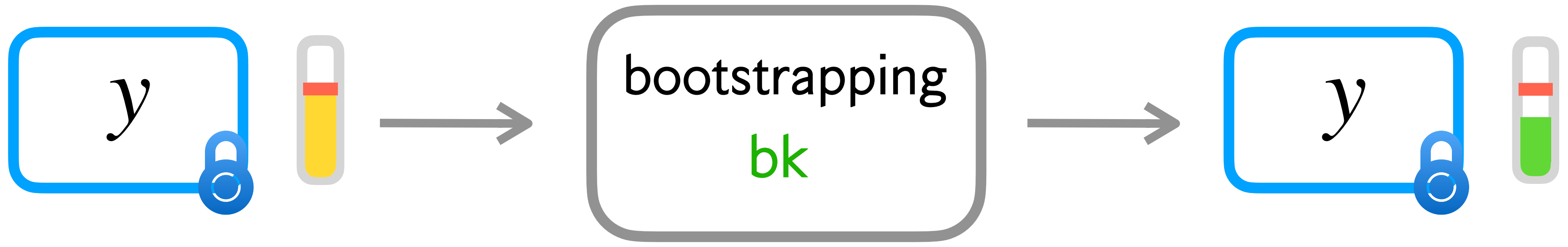
Leveled approach

- The largest the circuit, the largest the crypto parameters, the slowest the evaluation
- Circuit depth must be known in advance

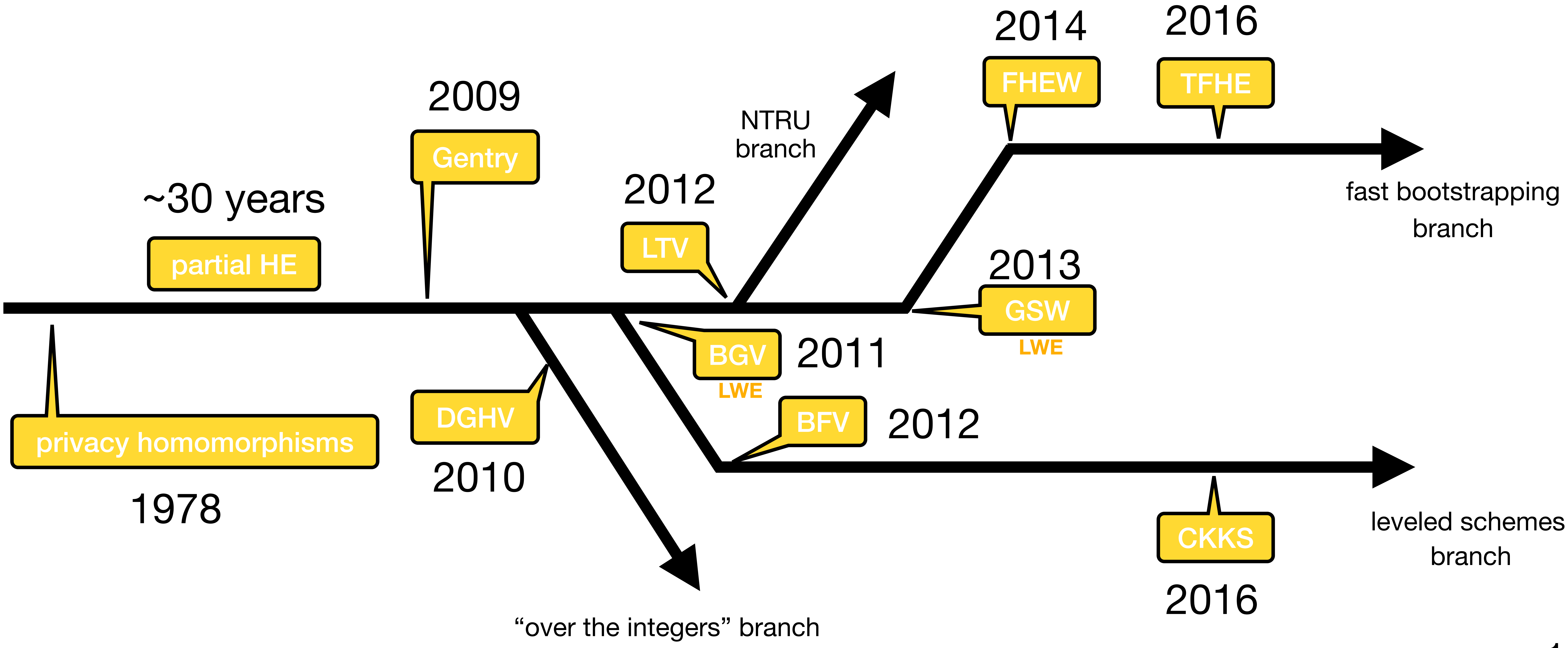
Your circuit is **deep** or **unknown**

Bootstrapped approach

- No depth limitations
- Bootstrap when needed



A timeline of ~40 years



Overview

- What is FHE?
- A little bit of history
- **FHE schemes based on LWE**
- TFHE ciphertexts and operations
- TFHE Bootstrapping
- Implementations and applications

Learning With Errors (LWE)

2005 - Regev: hard problem on lattices

RLWE - "LWE over the Rings"
2009 - Stehlé, Steinfeld, Tanaka, Xagawa
2010 - Lyubashevsky, Peikert, Regev

- Set a secret $(s_0, \ldots, s_{n-1}) \in \mathbb{Z}^n$
- Choose random elements $(a_0, \ldots, a_{n-1}) \in \mathbb{Z}_q^n$
- Choose a little random element $e \in \mathbb{Z}_q$ (Gaussian)
- Compute $b = \sum_{i=0}^{n-1} a_i \cdot s_i + e \in \mathbb{Z}_q$

$\}$ Call $(a_0, \ldots, a_{n-1}, b) \in \mathbb{Z}_q^{n+1}$ **LWE sample**

Decisional Problem

Given many **LWE samples:** $(a_0, \ldots, a_{n-1}, b) \in \mathbb{Z}_q^{n+1}$

Given many **random samples:** $(a_0, \ldots, a_{n-1}, u) \in \mathbb{Z}_q^{n+1}$

Hard to distinguish them!

Computational Problem

Given many **LWE samples:** $(a_0, \ldots, a_{n-1}, b) \in \mathbb{Z}_q^{n+1}$

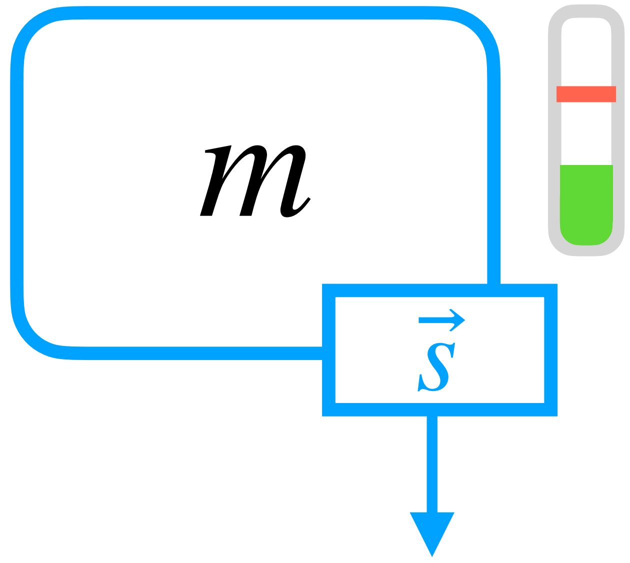
Hard to retrieve the secret

$(s_0, \ldots, s_{n-1}) \in \mathbb{Z}^n$**!**

LWE encryption (in the MSB)

Examples:
B/FV, CKKS, TFHE

Message $m \in \mathbb{Z}_p \longrightarrow$ Ciphertext in $\mathbb{Z}_q^{n+1}$



$(s_0, \dots, s_{n-1}) \in \{0,1\}^n$

$$= (\underbrace{\overbrace{a_0, \dots, a_{n-1}}^{\vec{a}}}_{a_i \in \mathbb{Z}_q}, \overbrace{b}^{b})$$

uniform random

where

$$b = \sum_{i=0}^{n-1} a_i \cdot s_i + \underbrace{e}_{\text{Gaussian}} + \Delta m \in \mathbb{Z}_q$$

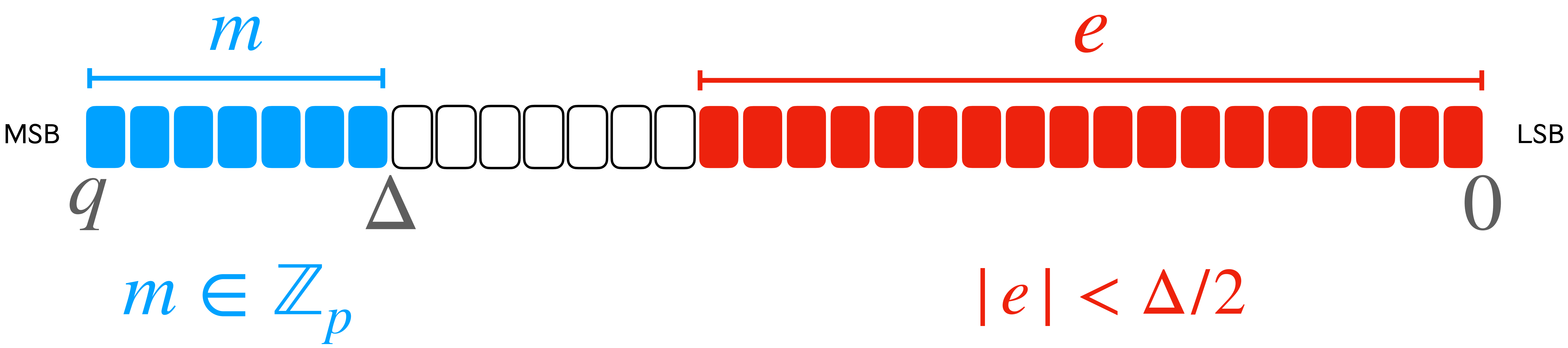
Decryption

1 $$\boxed{b} - \boxed{\vec{a}} \cdot \vec{s} = \Delta m + e$$

2 $$\left\lfloor \frac{\Delta m + e}{\Delta} \right\rceil \longrightarrow m$$

LWE encryption (in the MSB)

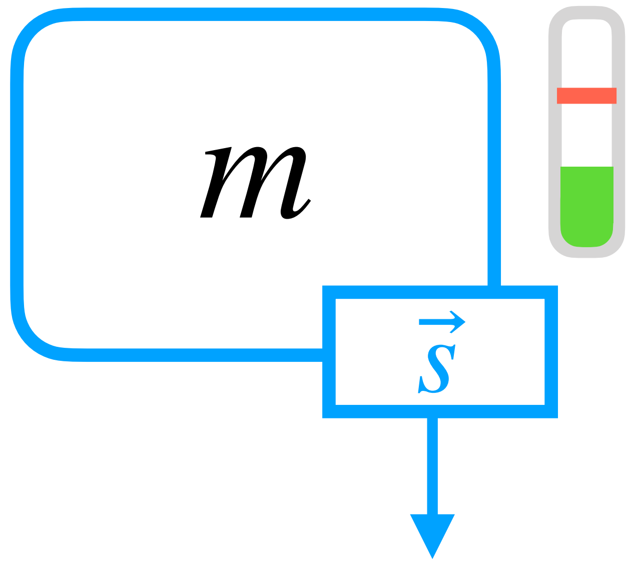
Why this works? $\left\lfloor \dfrac{\Delta m + e}{\Delta} \right\rceil \longrightarrow m$



LWE encryption (in the LSB)

Examples: BGV

Message $m \in \mathbb{Z}_p \longrightarrow$ Ciphertext in $\mathbb{Z}_q^{n+1}$



$(s_0, \dots, s_{n-1}) \in \{0,1\}^n$

$$= (\underbrace{\overbrace{a_0, \dots, a_{n-1}}^{\boxed{\vec{a}}}}_{a_i \in \mathbb{Z}_q}, \overbrace{b}^{\boxed{b}})$$

uniform random

where

$$b = \sum_{i=0}^{n-1} a_i \cdot s_i + p \cdot \underset{\substack{\downarrow \\ \text{Gaussian}}}{e} + m \in \mathbb{Z}_q$$

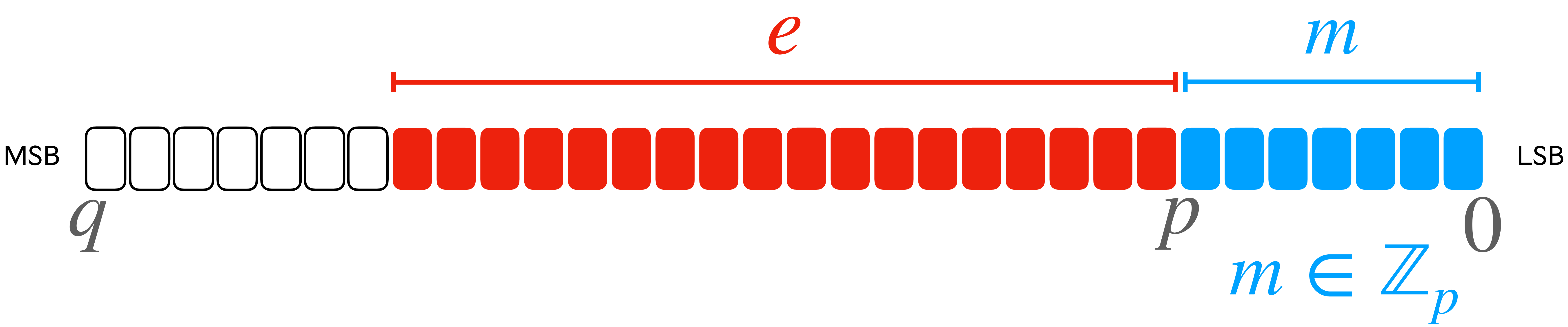
Decryption

1. $\boxed{b} - \boxed{\vec{a}} \cdot \vec{s} = p \cdot e + m$

2. $p \cdot e + m \mod p \longrightarrow m$

LWE encryption (in the LSB)

Why this works? $p \cdot e + m \mod p \longrightarrow m$

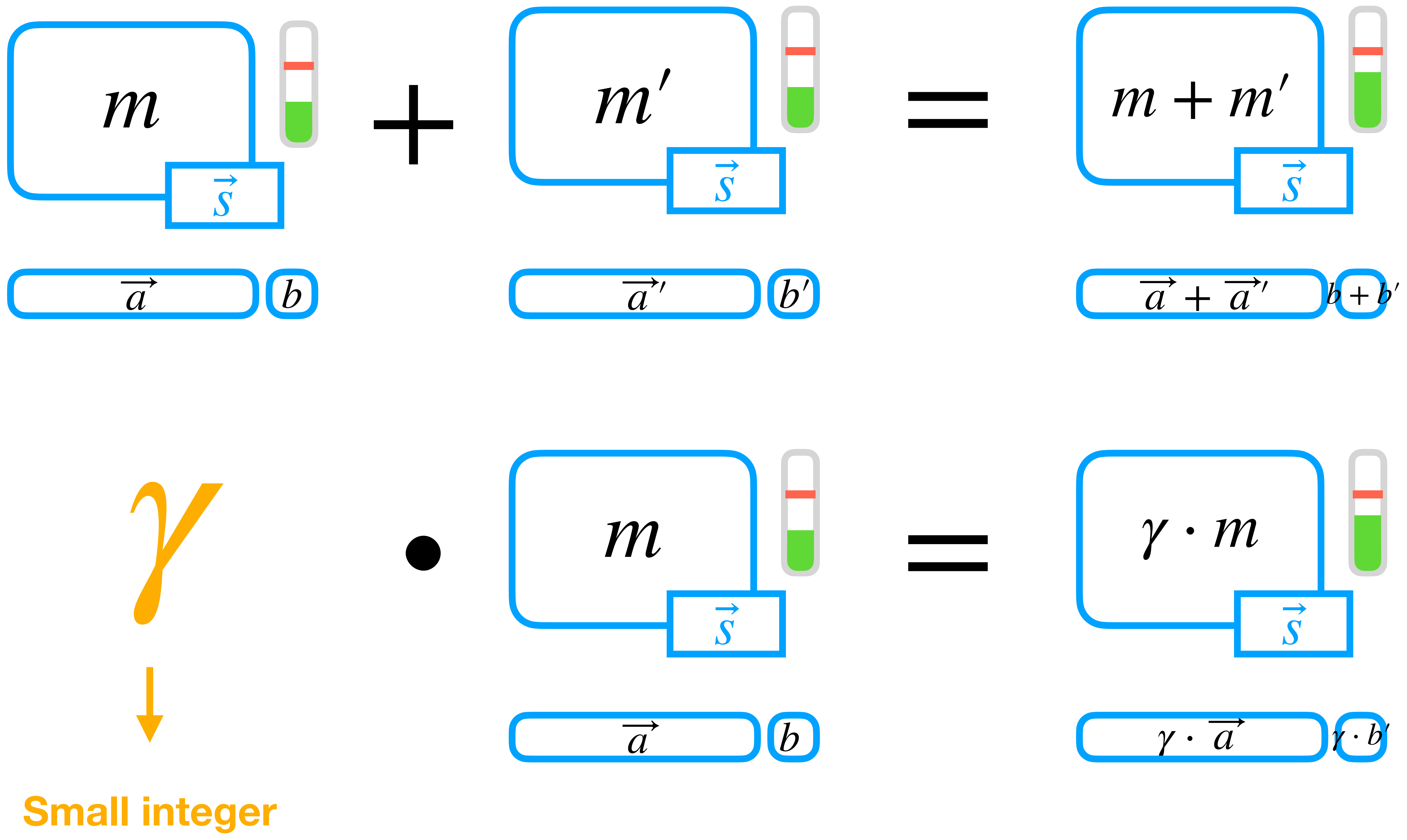


We will focus on MSB schemes

LWE homomorphic properties

Addition

Small constant multiplication

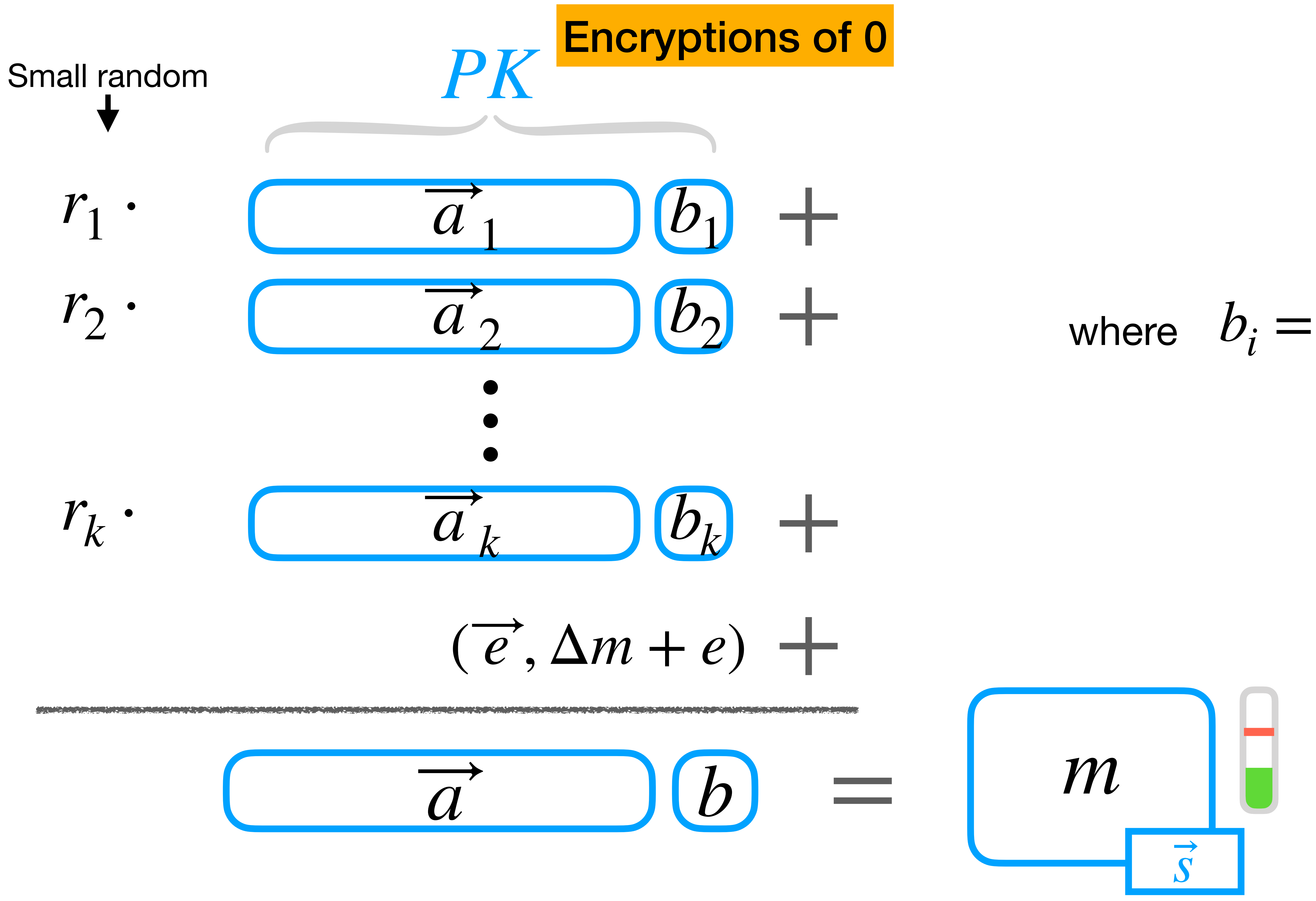


Small integer

LWE public key encryption

Message $m \in \mathbb{Z}_p \longrightarrow$ Ciphertext in $\mathbb{Z}_q^{n+1}$

$\vec{s} = (s_0, \ldots, s_{n-1}) \in \{0,1\}^n$



Small random

PK

Encryptions of 0

$r_1 \cdot \quad \vec{a}_1 \quad b_1 \quad +$

$r_2 \cdot \quad \vec{a}_2 \quad b_2 \quad +$

$\vdots$

$r_k \cdot \quad \vec{a}_k \quad b_k \quad +$

$(\vec{e}, \Delta m + e) \quad +$

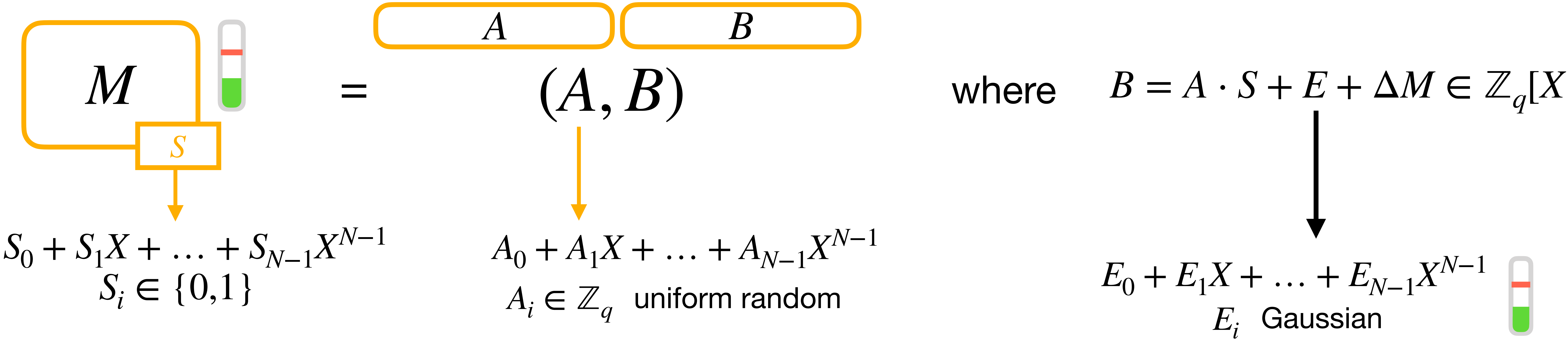
$\vec{a} \quad b \quad = \quad m \quad \vec{s}$

where $b_i = \vec{a}_i \cdot \vec{s} + e_i \in \mathbb{Z}_q$

Gaussian

RLWE encryption (in the MSB)

Message $M \in \mathbb{Z}_p[X]/(X^N+1) \longrightarrow$ **Ciphertext in** $\left(\mathbb{Z}_q[X]/(X^N+1)\right)^2$



M S $=$ A B (A, B) where $B = A \cdot S + E + \Delta M \in \mathbb{Z}_q[X]/(X^N+1)$

$S_0 + S_1X + \ldots + S_{N-1}X^{N-1}$
$S_i \in \{0,1\}$

$A_0 + A_1X + \ldots + A_{N-1}X^{N-1}$
$A_i \in \mathbb{Z}_q$ uniform random

$E_0 + E_1X + \ldots + E_{N-1}X^{N-1}$
E_i Gaussian

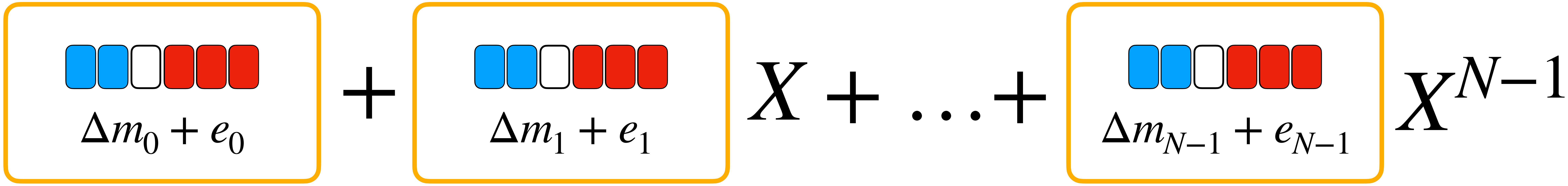
Decryption

1 $B - A \cdot S = \Delta M + E$

2 $\left\lfloor \dfrac{\Delta M + E}{\Delta} \right\rceil \longrightarrow M$

RLWE encryption (in the MSB)

Why this works? $\left\lfloor \dfrac{\Delta M + E}{\Delta} \right\rceil \longrightarrow M$

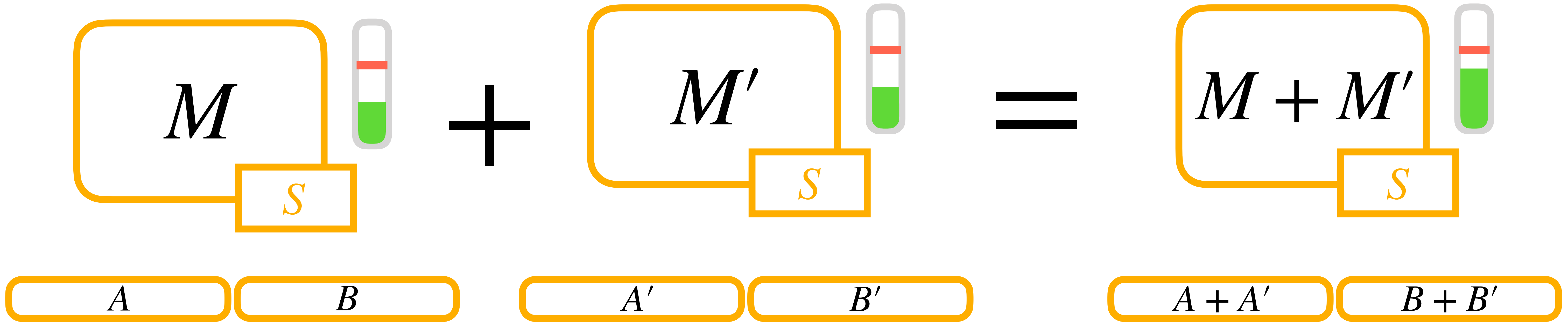


$$\boxed{\Delta m_0 + e_0} + \boxed{\Delta m_1 + e_1}\, X + \ldots + \boxed{\Delta m_{N-1} + e_{N-1}}\, X^{N-1}$$

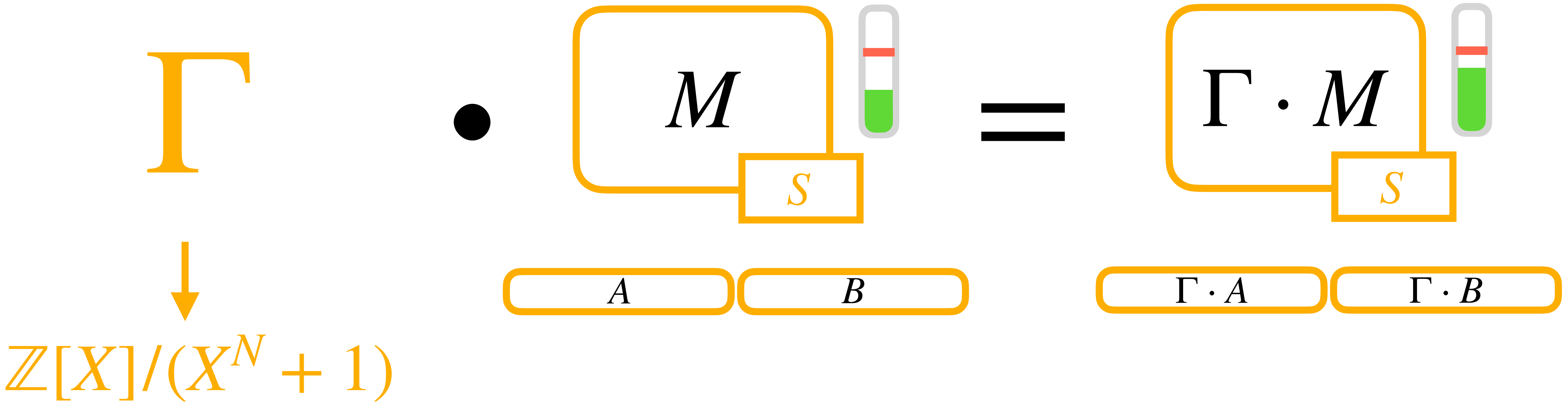
$|e_i| < \Delta/2$

RLWE homomorphic properties

Addition



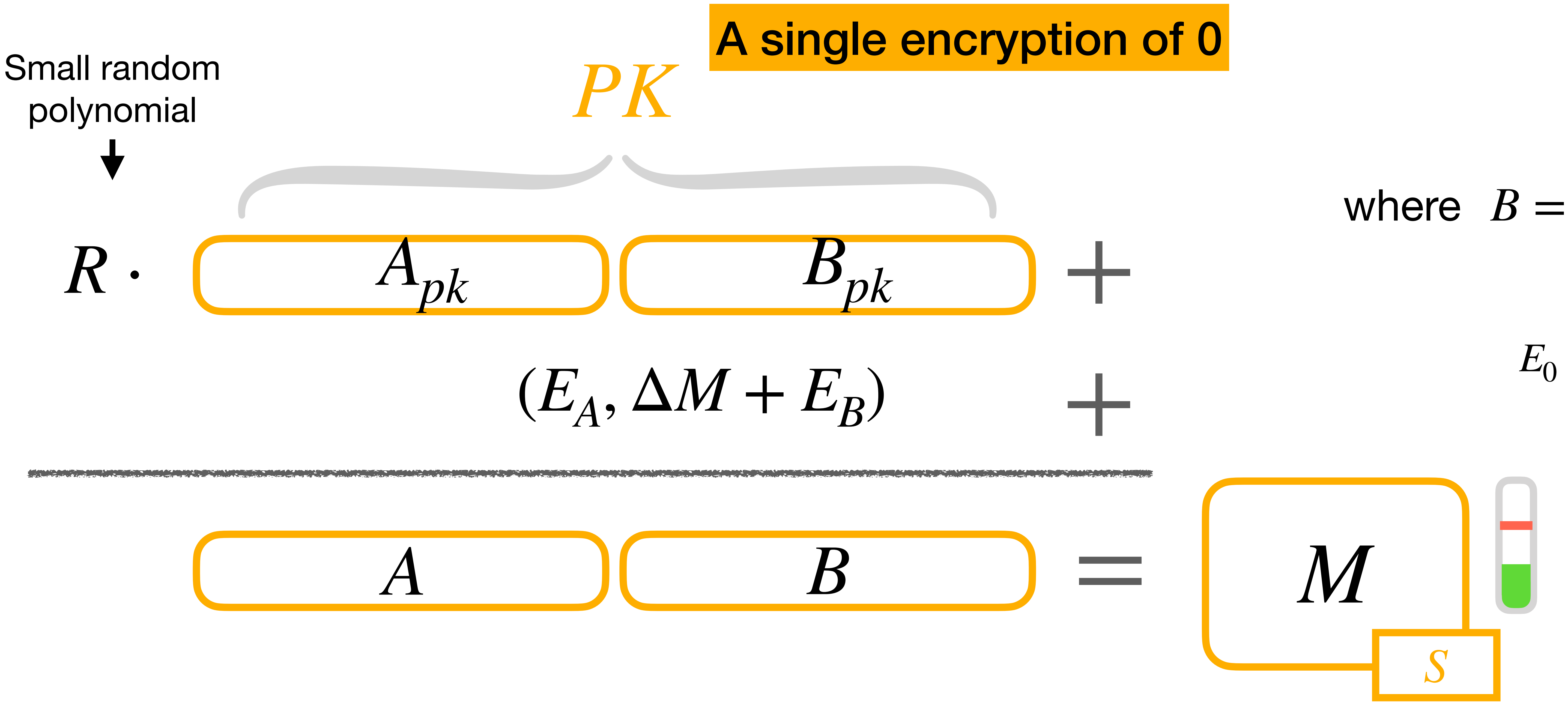
Small constant polynomial multiplication



RLWE public key encryption

Message $M \in \mathbb{Z}_p[X]/(X^N+1) \longrightarrow$ **Ciphertext in** $\left(\mathbb{Z}_q[X]/(X^N+1)\right)^2$

$$S = S_0 + S_1X + \ldots + S_{N-1}X^{N-1}$$
$$S_i \in \{0,1\}$$



A single encryption of 0

Small random polynomial

$$R \cdot \overbrace{\boxed{A_{pk}}\;\boxed{B_{pk}}}^{PK} +$$
$$(E_A, \Delta M + E_B) \quad +$$
$$\boxed{A}\;\boxed{B} = \boxed{M}_{\boxed{S}}$$

where $B = A \cdot S + E \in \mathbb{Z}_q[X]/(X^N+1)$

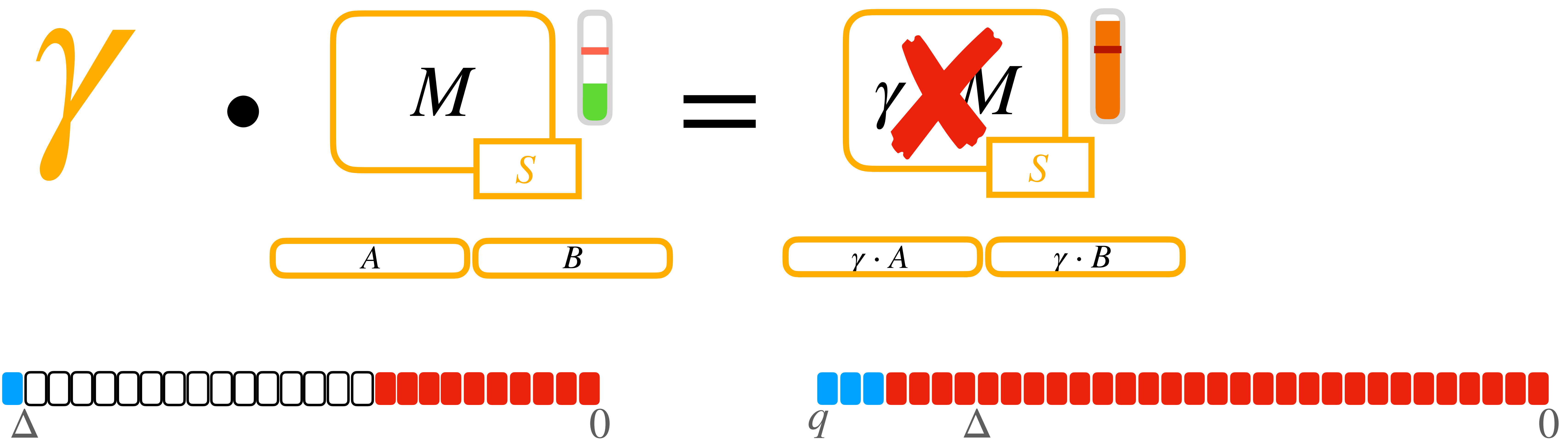
$$E_0 + E_1X + \ldots + E_{N-1}X^{N-1}$$

E_i Gaussian

What if we want to multiply for a large constant?

RLWE homomorphic properties

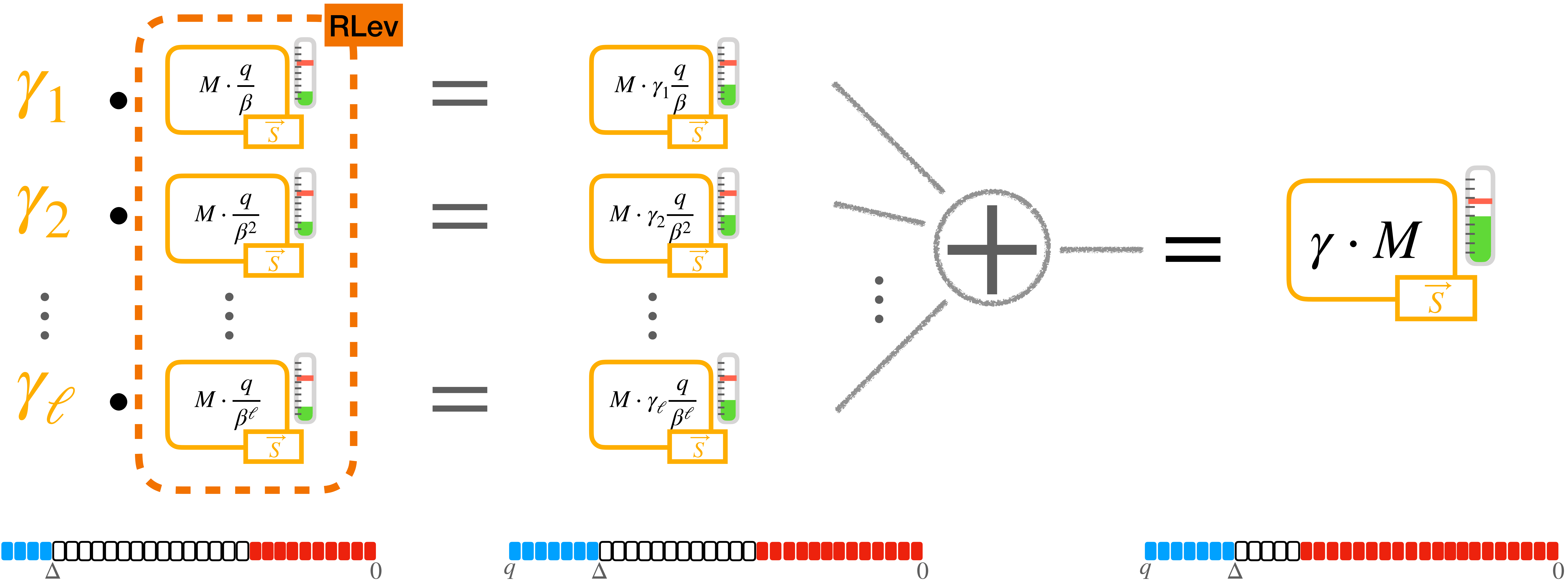
For simplicity $\Gamma = \gamma \in \mathbb{Z}$ large (order of q)



RLWE homomorphic properties

Decompose with respect to a small base (e.g., $\beta = 2$)

$$\gamma = \gamma_1 \frac{q}{\beta} + \gamma_1 \frac{q}{\beta^2} + \ldots + \gamma_\ell \frac{q}{\beta^\ell}$$



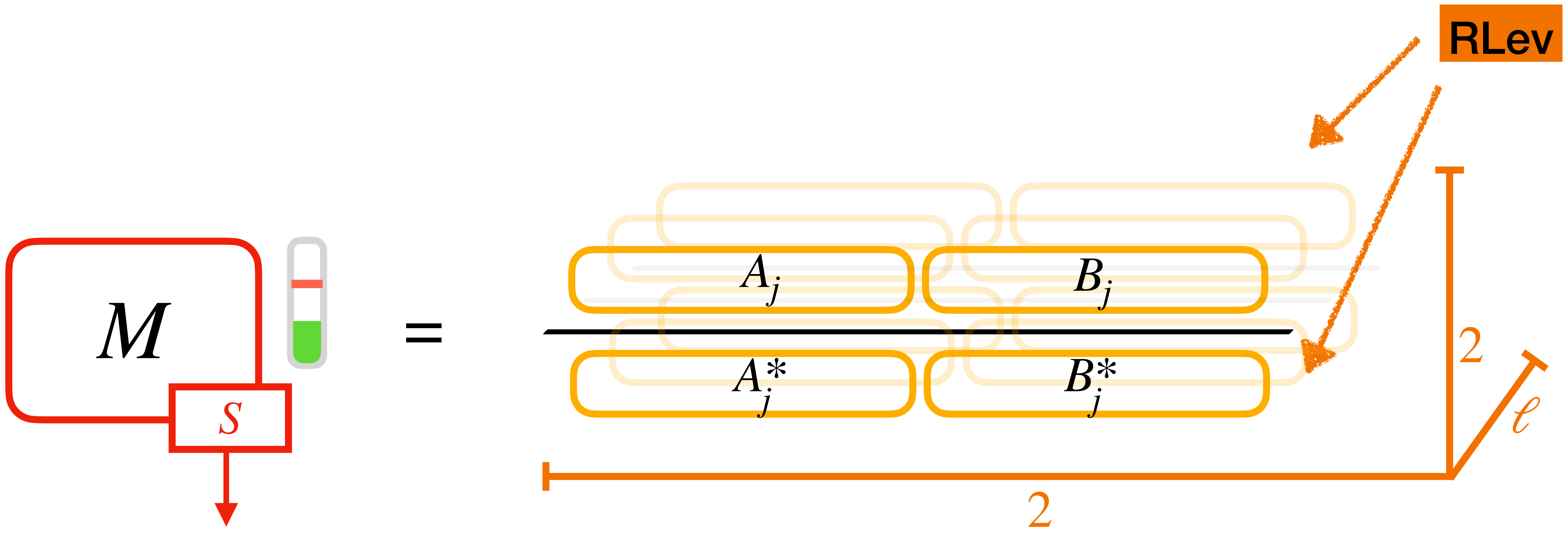
RLev
γ_1
γ_2
γ_ℓ
$M \cdot \frac{q}{\beta}$
$M \cdot \frac{q}{\beta^2}$
$M \cdot \frac{q}{\beta^\ell}$
$\vec{s}$
$M \cdot \gamma_1 \frac{q}{\beta}$
$M \cdot \gamma_2 \frac{q}{\beta^2}$
$M \cdot \gamma_\ell \frac{q}{\beta^\ell}$
$\gamma \cdot M$
q
Δ
0

Two ways of doing multiplication between ciphertexts

- GSW -

RGSW

Message $M \in \mathbb{Z}_p[X]/(X^N+1) \longrightarrow$ **Ciphertext in** $\left(\mathbb{Z}_q[X]/(X^N+1)\right)^{2\ell \times 2}$



$S_0 + S_1X + \ldots + S_{N-1}X^{N-1}$

$S_i \in \{0,1\}$

RLWE

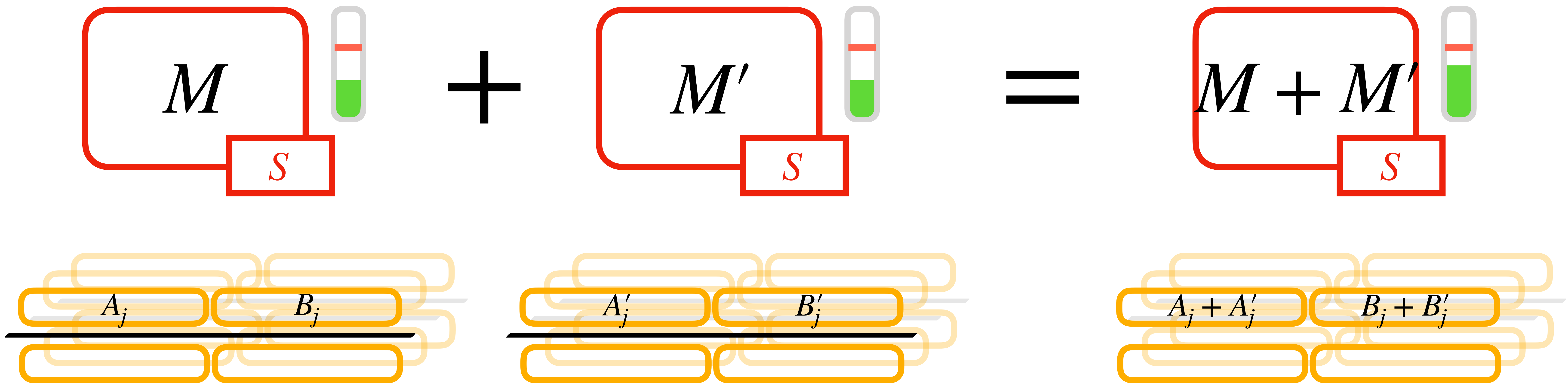
$$B_j = A_j \cdot S + E_j - M \cdot S \cdot \frac{q}{\beta^j}$$

$$B_j^* = A_j^* \cdot S + E_j^* + M \cdot \frac{q}{\beta^j}$$

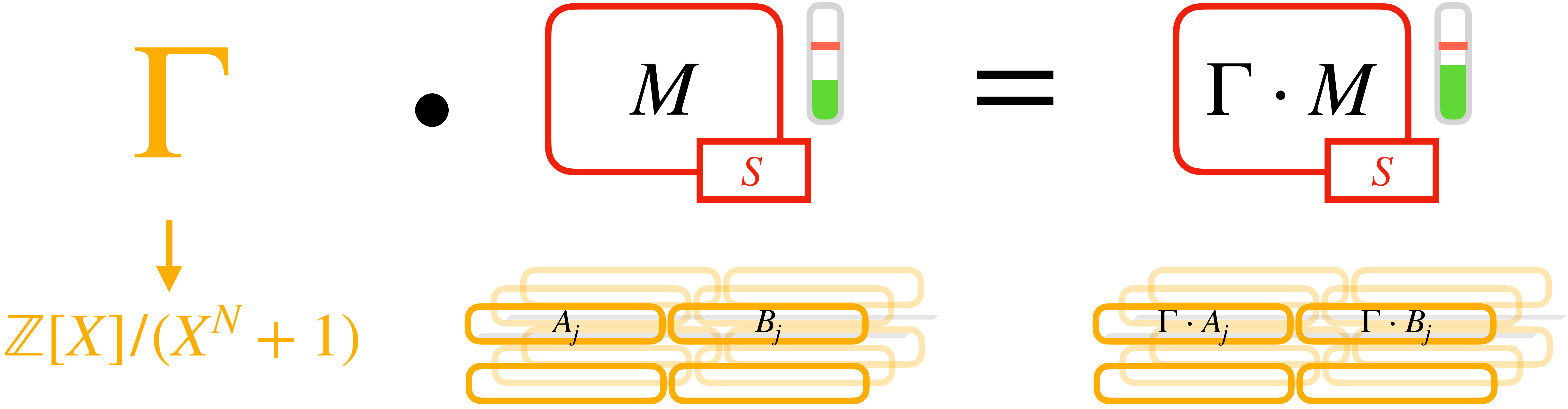
$$j = 1,\ldots,\ell$$

RGSW

Addition

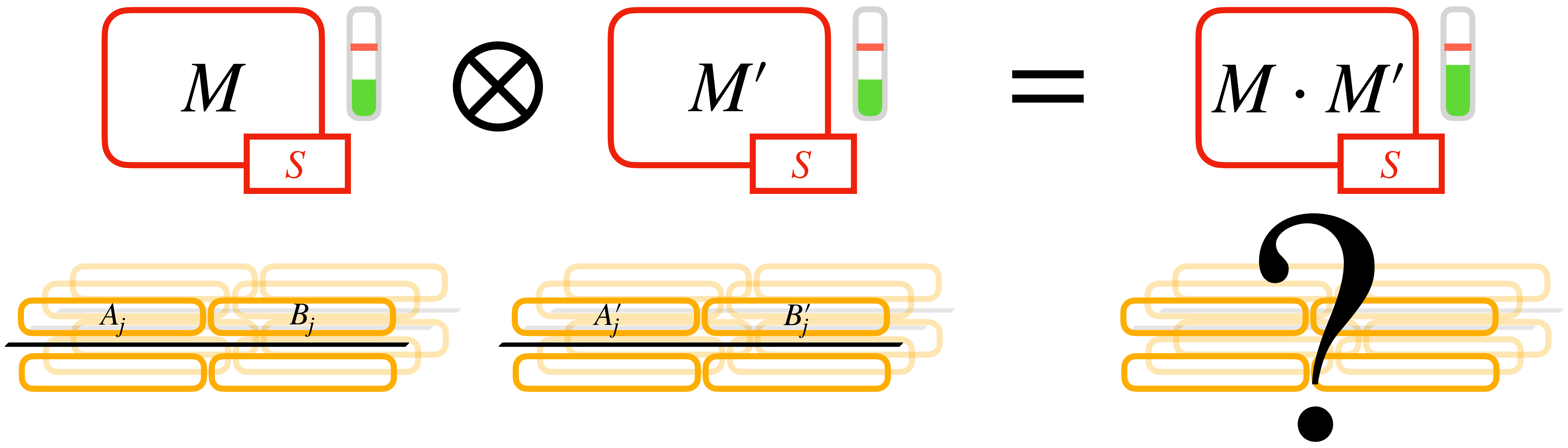


Small constant polynomial multiplication

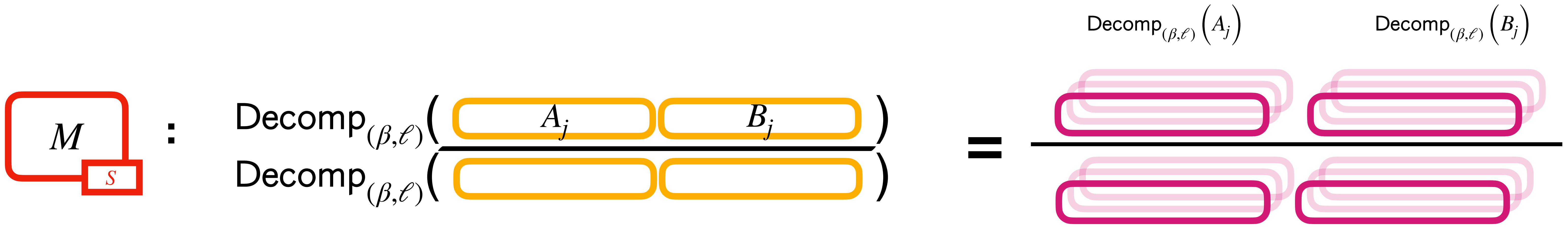


RGSW

Multplication

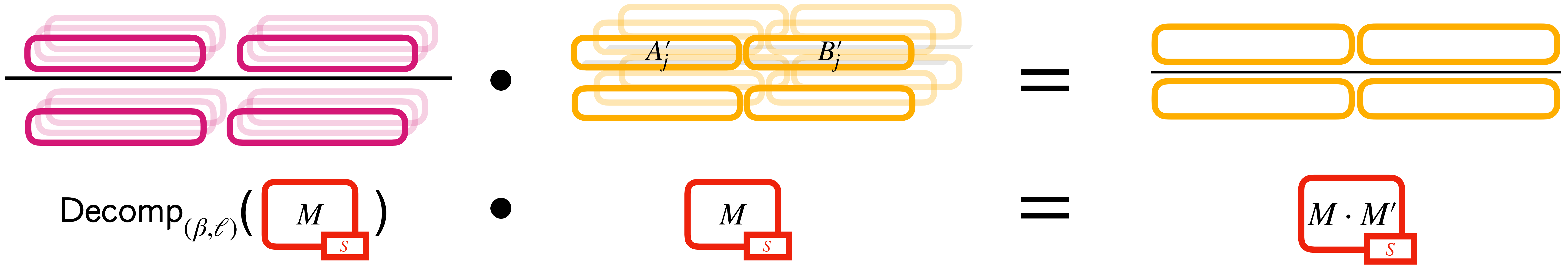


1 - Decompose



M
S
:
$\text{Decomp}_{(\beta,\ell)}($ A_j B_j $)$
$\text{Decomp}_{(\beta,\ell)}($ $)$
=
$\text{Decomp}_{(\beta,\ell)}(A_j)$
$\text{Decomp}_{(\beta,\ell)}(B_j)$

2 - Matrix dot-product:

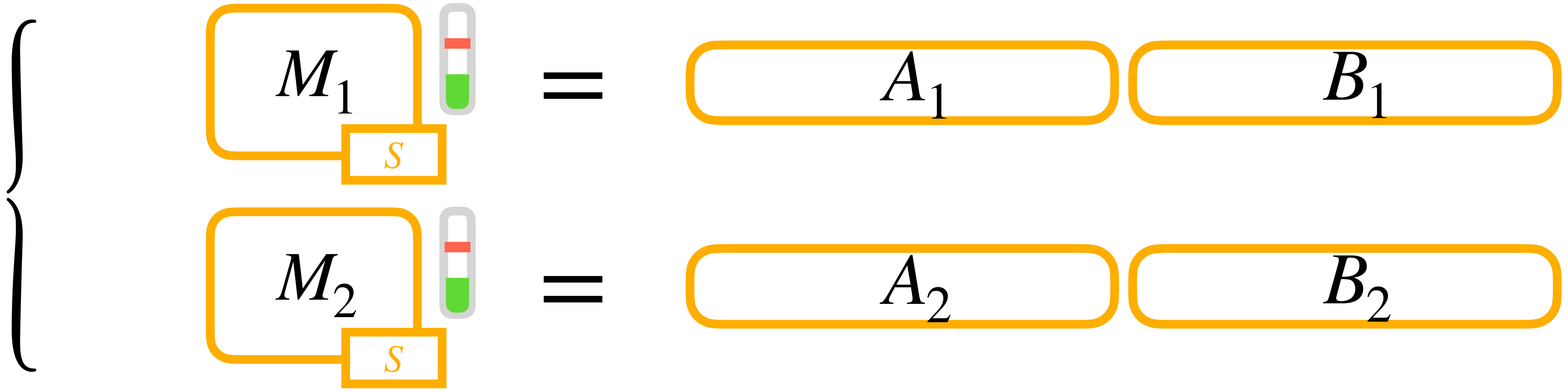


Two ways of doing multiplication between ciphertexts

- BGV -

RLWE multiplication (BGV style)

Input: two RLWE ciphertexts



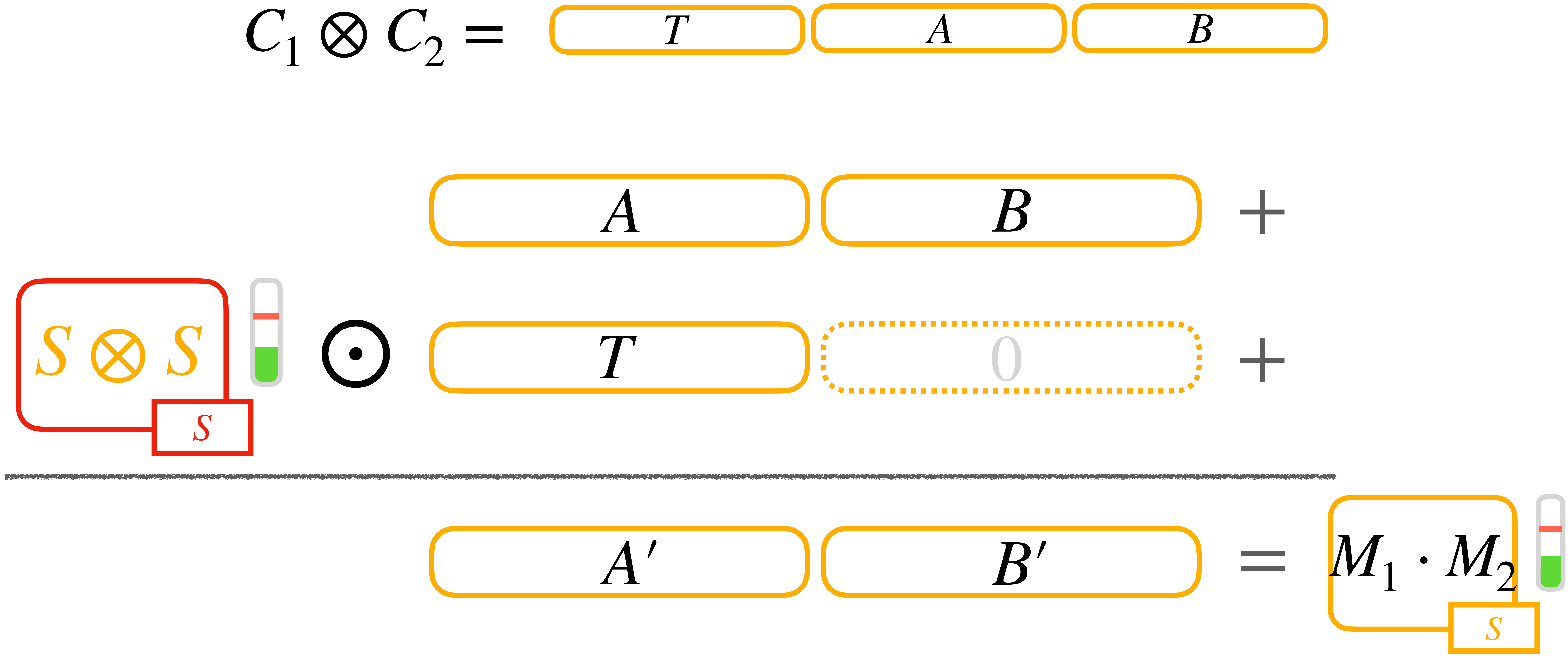
① **Tensor product:** $C_1 \otimes C_2 = \boxed{\quad T \quad}\boxed{\quad A \quad}\boxed{\quad B \quad}$

$$T = \left[\left\lfloor \frac{A_1 \cdot A_2}{\Delta} \right\rceil\right]_q \qquad A = \left[\left\lfloor \frac{A_1 \cdot B_2 + A_2 \cdot B_1}{\Delta} \right\rceil\right]_q \qquad B = \left[\left\lfloor \frac{B_1 \cdot B_2}{\Delta} \right\rceil\right]_q$$

Encrypted under the secret key $S \otimes S$

RLWE multiplication (BGV style)

2 **Relinearization:** switching the key



$C_1 \otimes C_2 =$ T A B

A B $+$

$S \otimes S$ S $\odot$ T 0 $+$

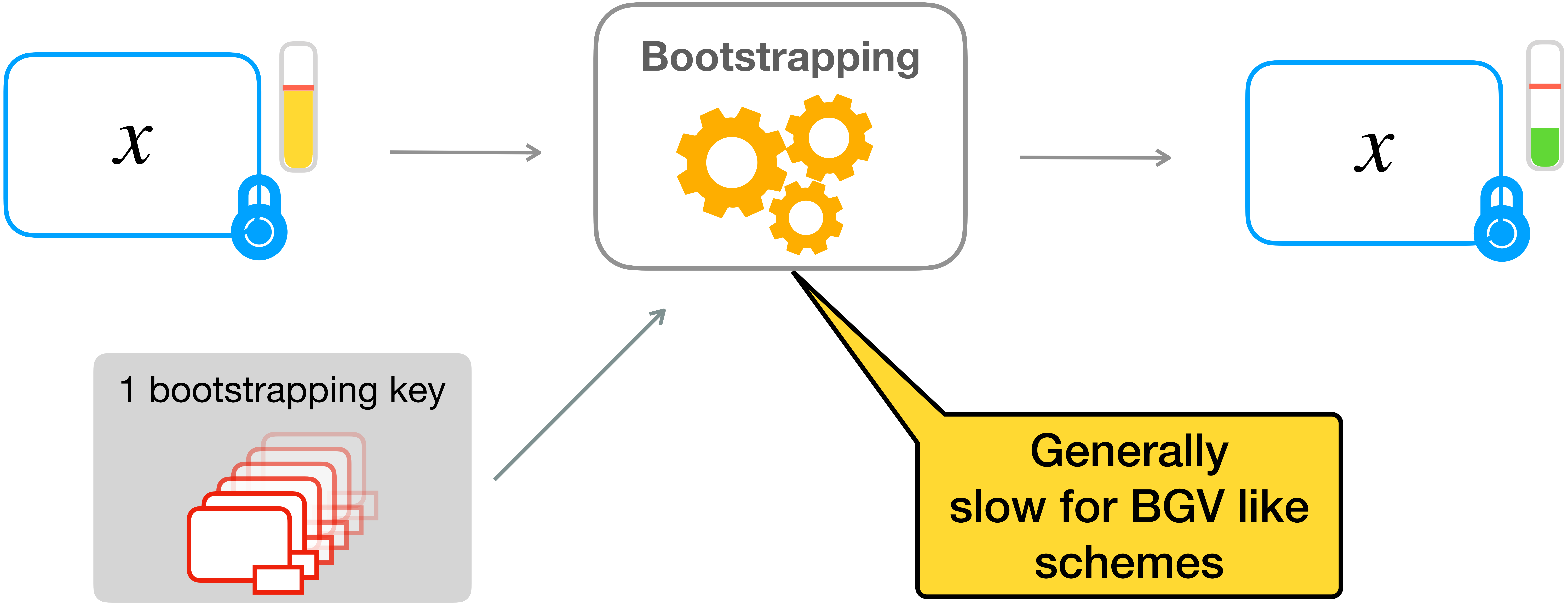
A' B' $=$ $M_1 \cdot M_2$ S

How to deal with noise?

🌡️

Bootstrapping

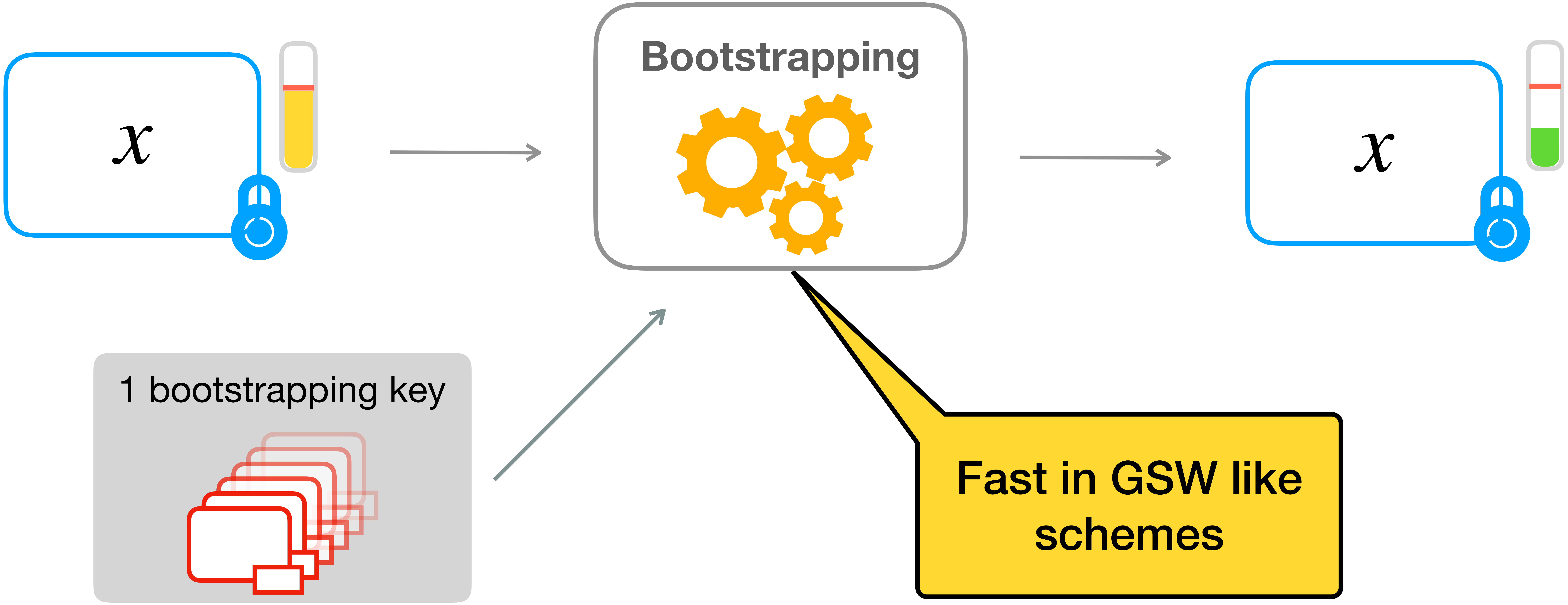
2009 - Gentry



Generally slow for BGV like schemes

Bootstrapping

2009 - Gentry

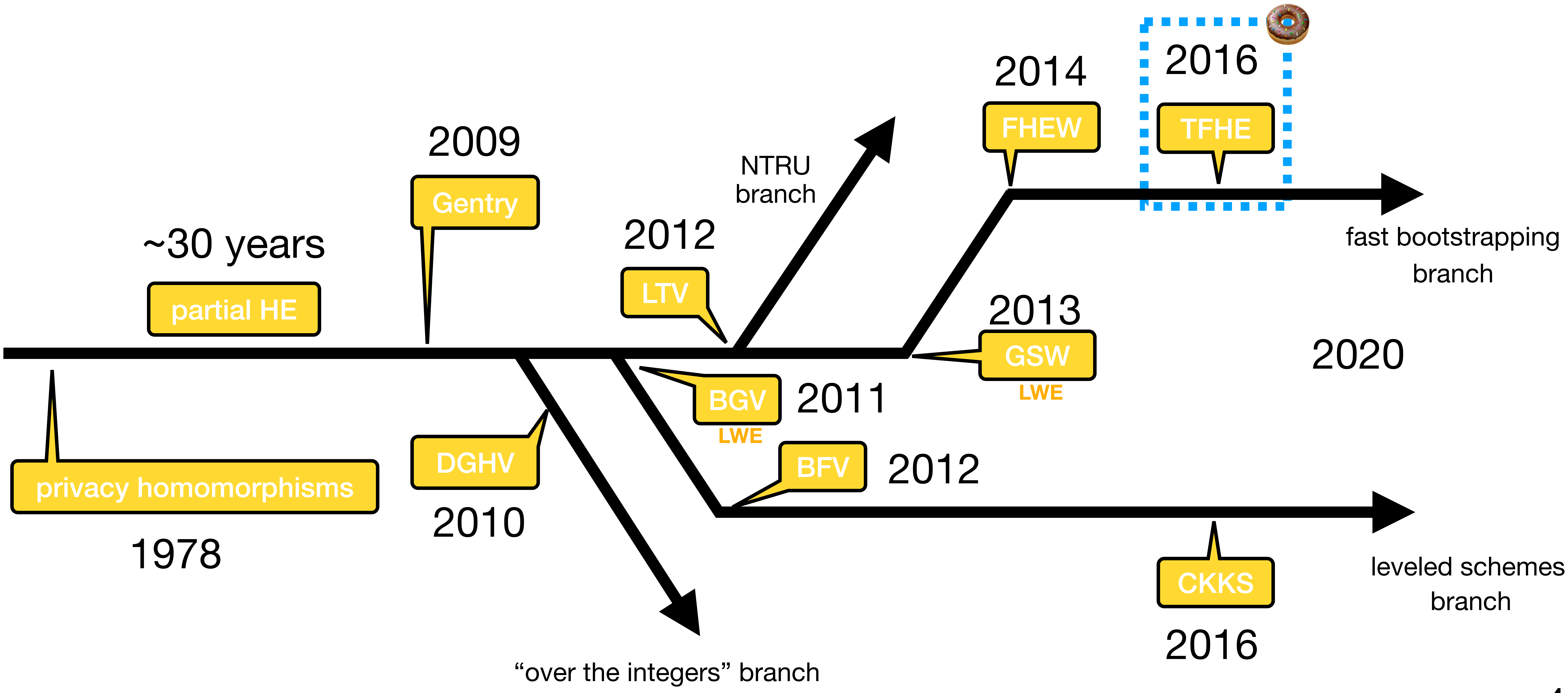


Fast in GSW like schemes

Overview

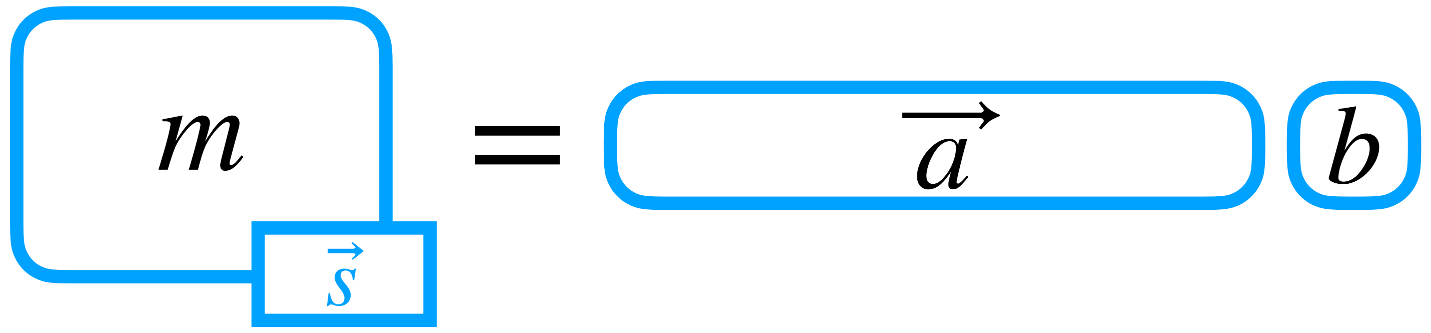
- What is FHE?
- A little bit of history
- FHE schemes based on LWE
- **TFHE ciphertexts and operations**
- TFHE Bootstrapping
- Implementations and applications

A timeline of ~40 years



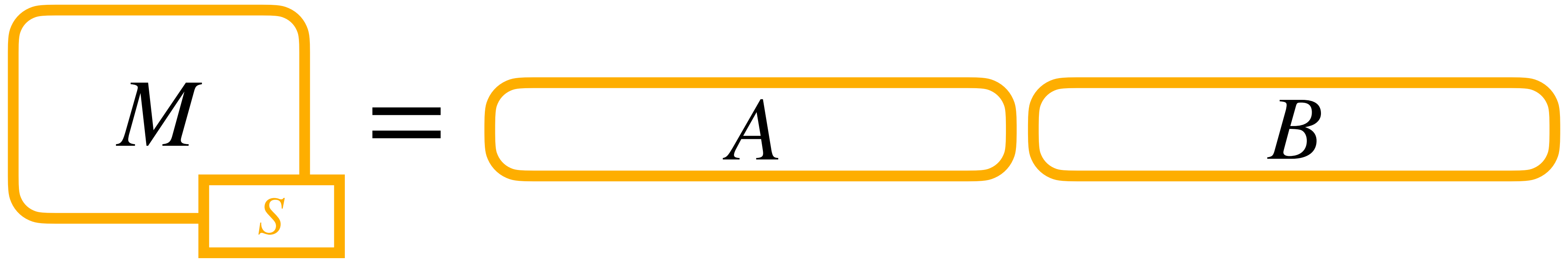
Ciphertexts: Summary

LWE



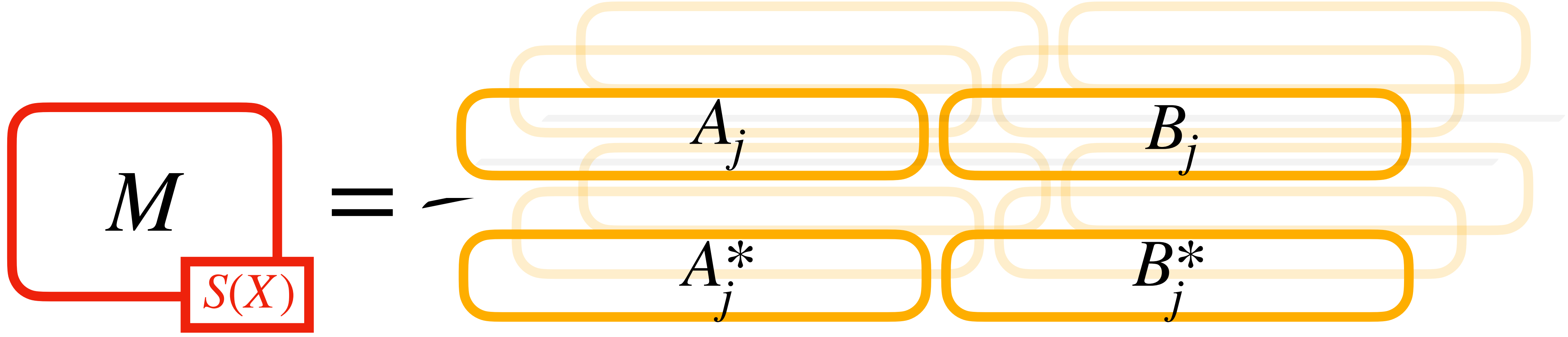
- Addition
- Constant multiplication

RLWE



- Addition
- Constant multiplication

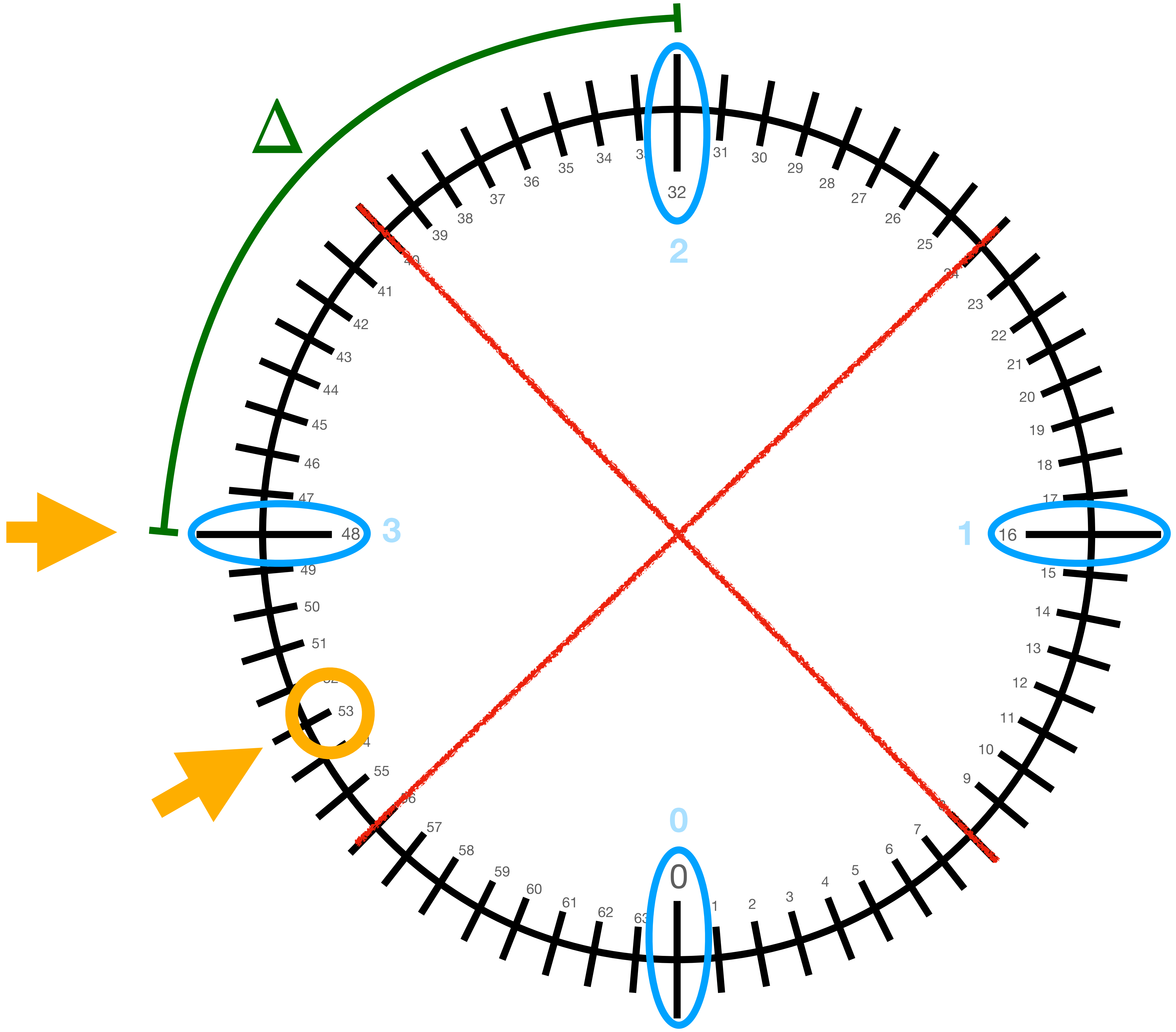
RGSW



- Addition
- Constant multiplication
- Multiplication

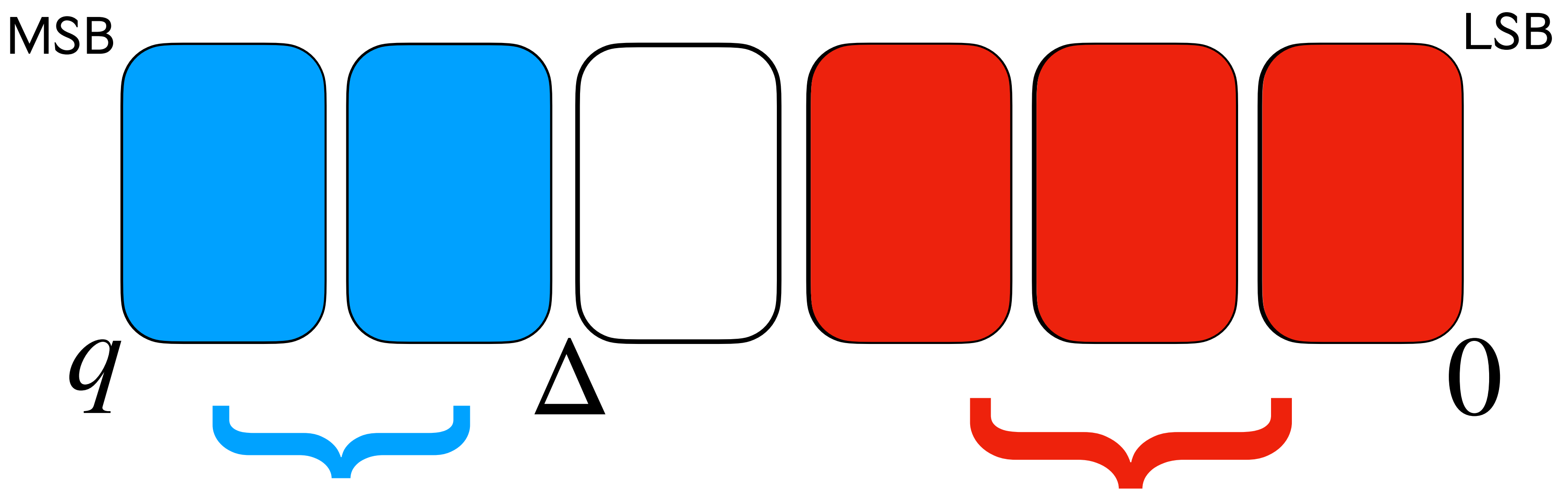
LWE

Encoding 🍩



$$\begin{cases} q = 64 = 2^6 \\ p = 4 = 2^2 \\ \Delta = \dfrac{q}{p} = 16 = 2^4 \end{cases}$$

In practice: $q = 2^{32}$ or $q = 2^{64}$



$\mathscr{M} = \{0,1,2,3\}$

$\text{Encode}(m) = \Delta m$

$|e| < \dfrac{\Delta}{2} = 8 = 2^3$

Example: $m = 3$

$\Delta m = 48$

$e = 5$

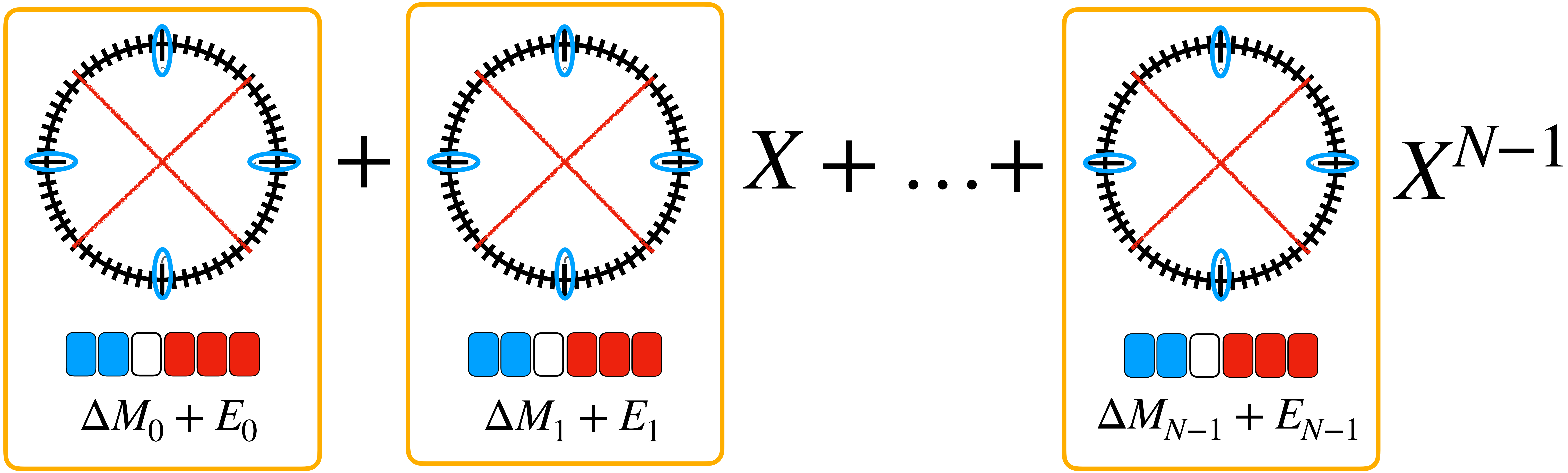
1 1 0 1 0 1

$\Delta m + e = 53$

RLWE

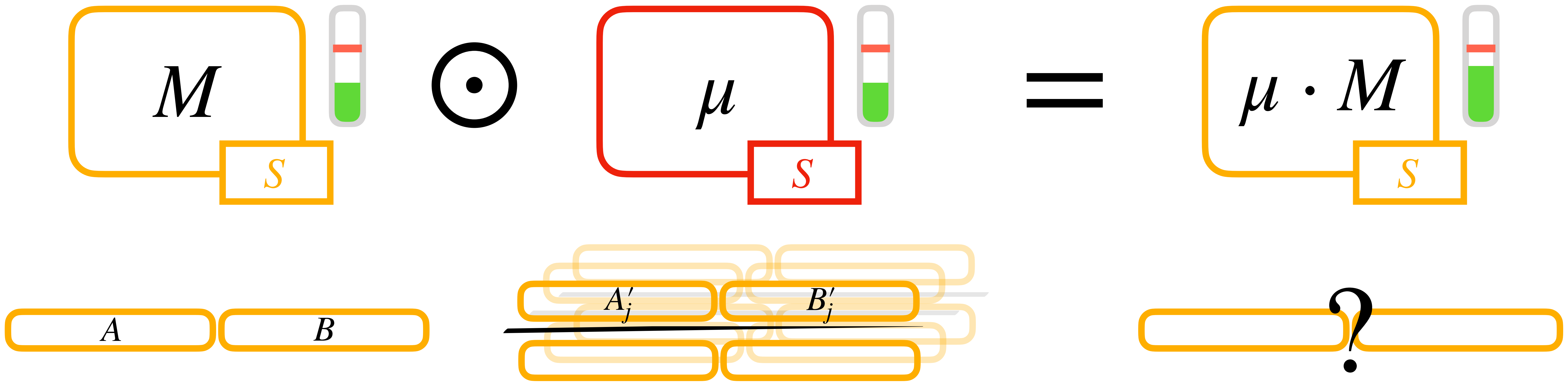
Encoding

$$\Delta M + E \text{ with } \begin{cases} M = M_0 + M_1X + \ldots + M_{N-1}X^{N-1} \\ E = E_0 + E_1X + \ldots + E_{N-1}X^{N-1} \end{cases}$$

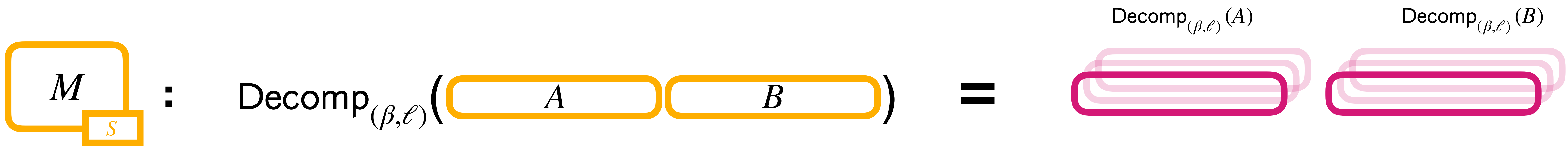


External Product

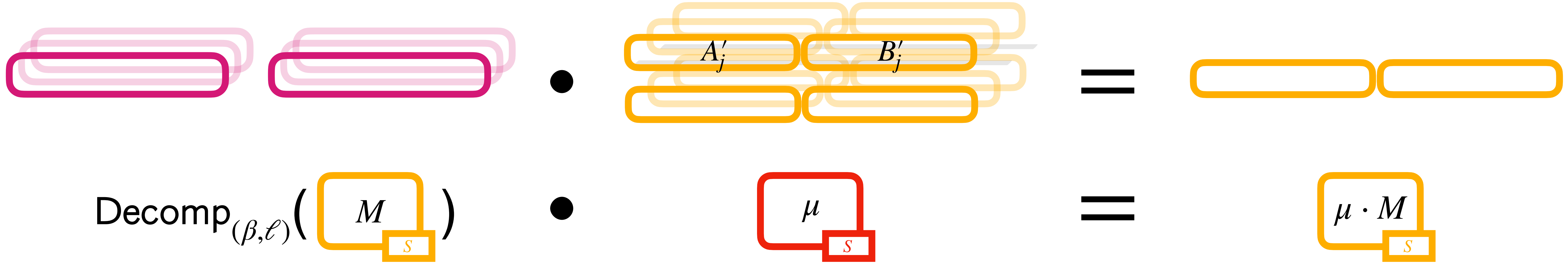
RLWE x RGSW



1 - Decompose

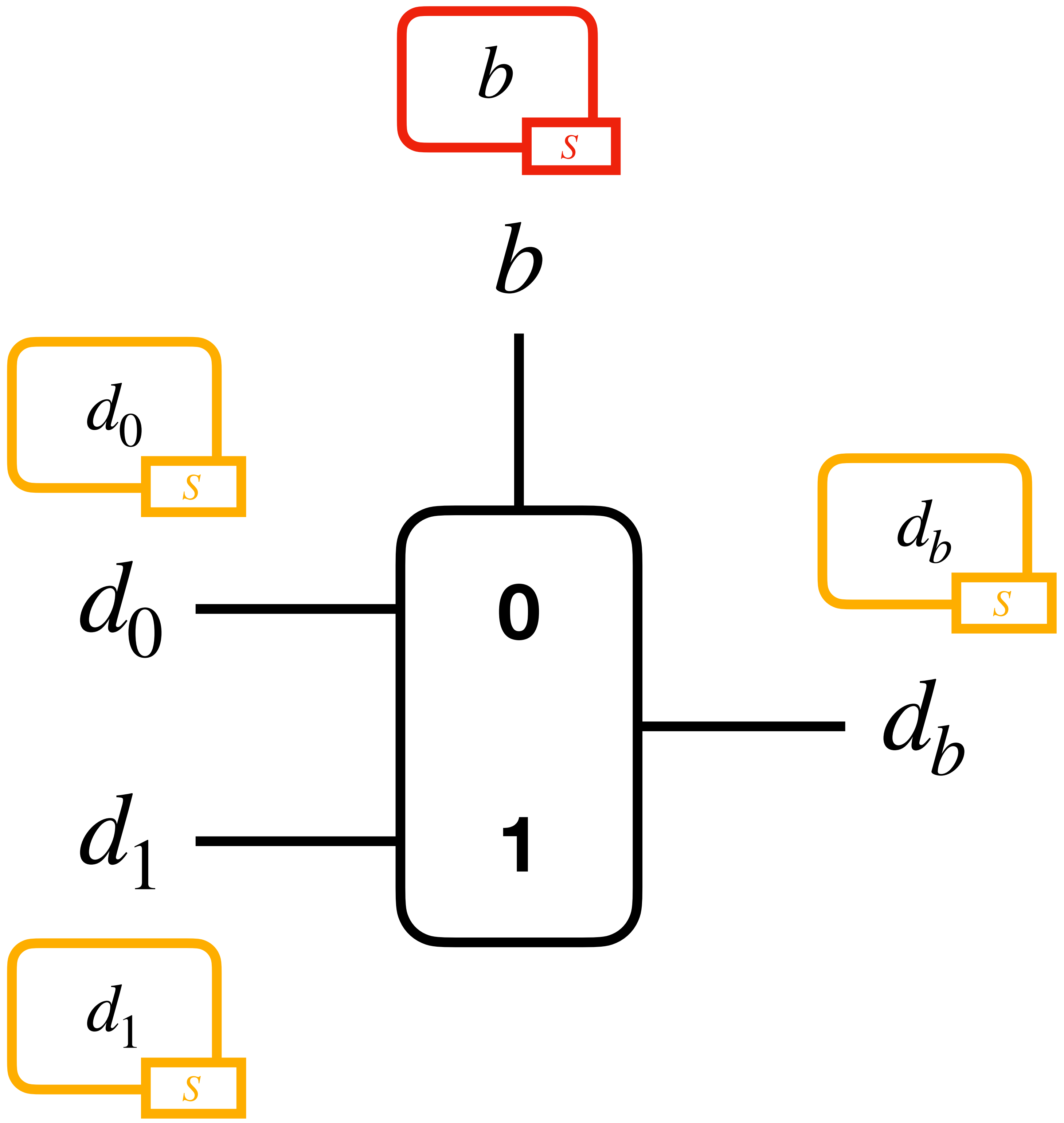

M
S
:
$\text{Decomp}_{(\beta,\ell)}($ A B $)$
$=$
$\text{Decomp}_{(\beta,\ell)}(A)$
$\text{Decomp}_{(\beta,\ell)}(B)$

2 - Vector-matrix dot-product:

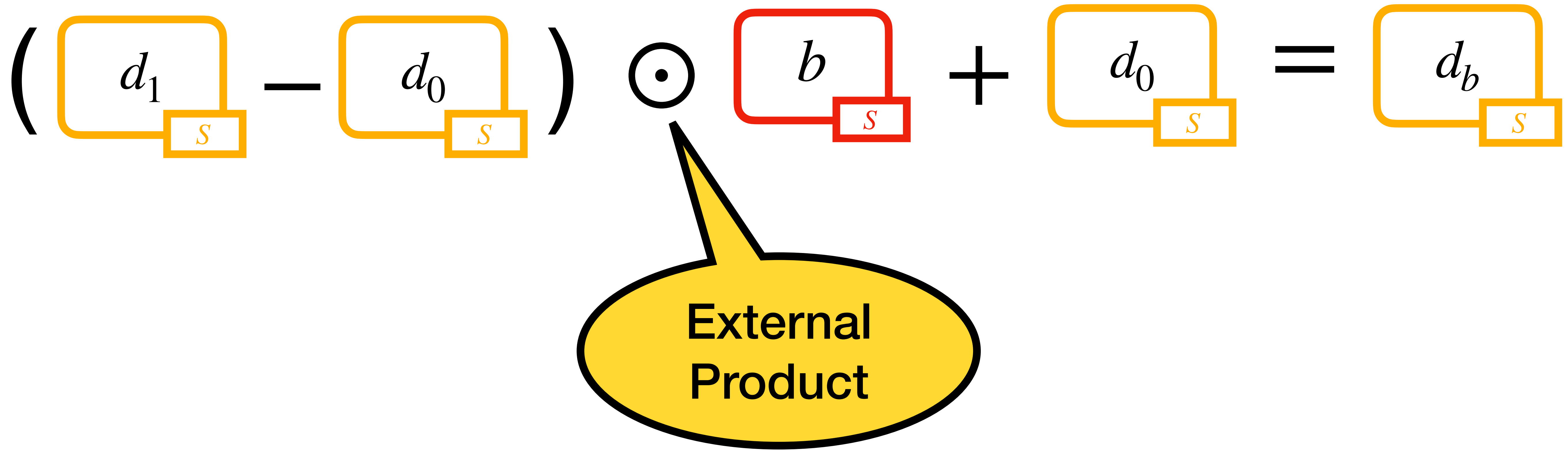


CMux

Controlled Mux



$$(d_1 - d_0) \cdot b + d_0 = d_b$$



Rotation

Rotate a polynomial M of p positions

$\cdot X^{-p}$

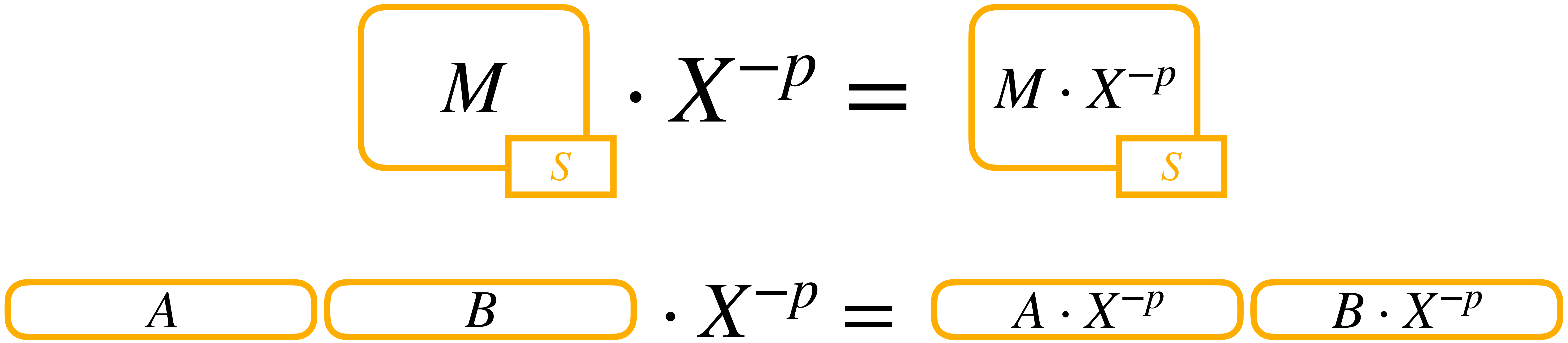
$$M(X) = M_0 + M_1 X + \ldots + M_p X^p + \ldots + M_{N-1} X^{N-1}$$

$$M(X) \cdot X^{-p} = M_p + M_{p+1} X + \ldots + M_{N-1} X^{N-p-1} - M_0 X^{N-p} - \ldots - M_{p-1} X^{N-1}$$

$\text{mod } X^N + 1$

$X^N = -1$

Rotate an encrypted polynomial M of p positions



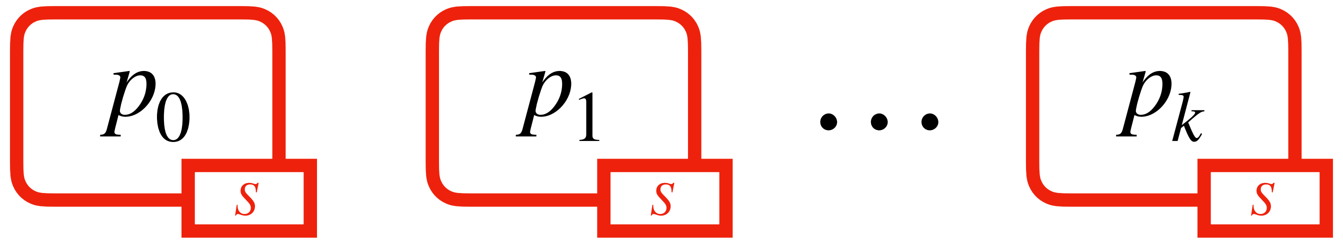
$$\boxed{M}_{S} \cdot X^{-p} = \boxed{M \cdot X^{-p}}_{S}$$

$$\boxed{A}\ \boxed{B} \cdot X^{-p} = \boxed{A \cdot X^{-p}}\ \boxed{B \cdot X^{-p}}$$

Blind Rotation

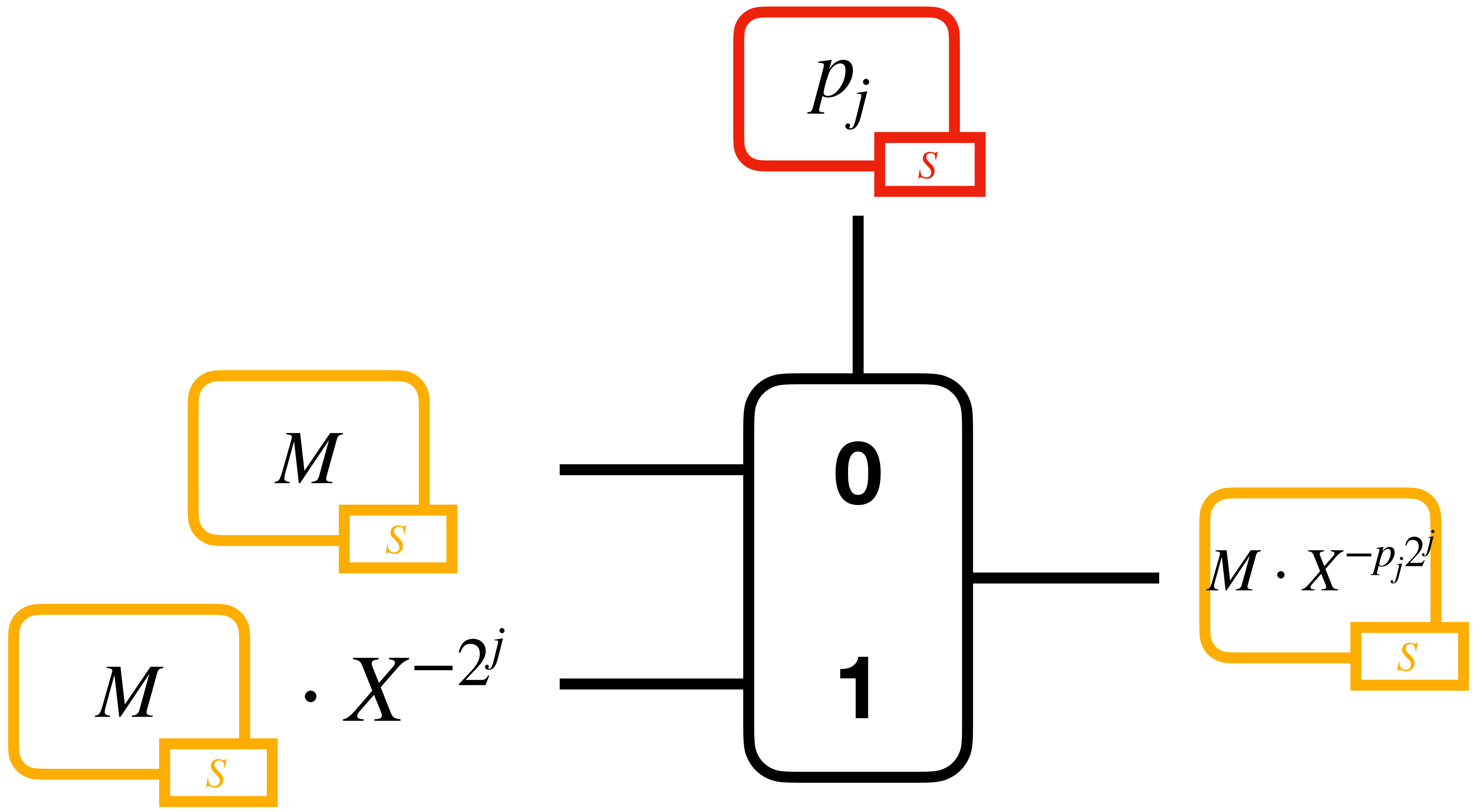
Rotate an **encrypted** polynomial M of p **encrypted positions**

$$p = p_0 \cdot 2^0 + \ldots + \underset{\text{Secret}}{p_j}\underset{\text{Known Constant}}{2^j} + \ldots + p_k \cdot 2^k$$



$$M \cdot X^{-p} = M \cdot X^{-p_0 \cdot 2^0 - \ldots - p_j 2^j - \ldots - p_k \cdot 2^k}$$
$$= M \cdot X^{-p_0 \cdot 2^0} \cdot \ldots \cdot X^{-p_j 2^j} \cdot \ldots \cdot X^{-p_k \cdot 2^k}$$

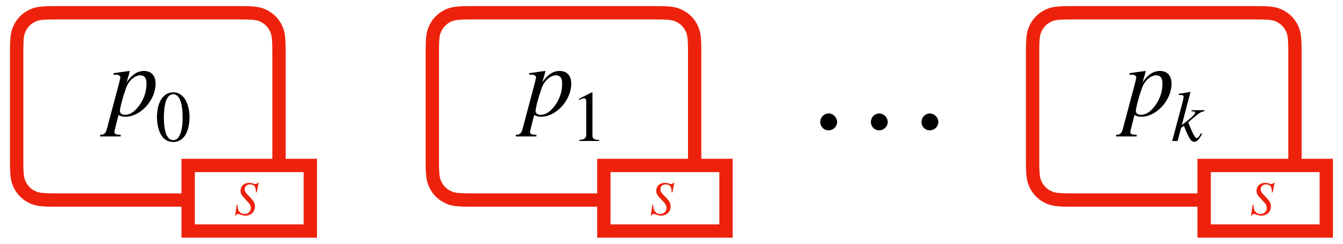
$$M \cdot X^{-p_j 2^j} = \begin{cases} M & \text{if } p_j = 0 \\ M \cdot X^{-2^j} & \text{if } p_j = 1 \end{cases}$$

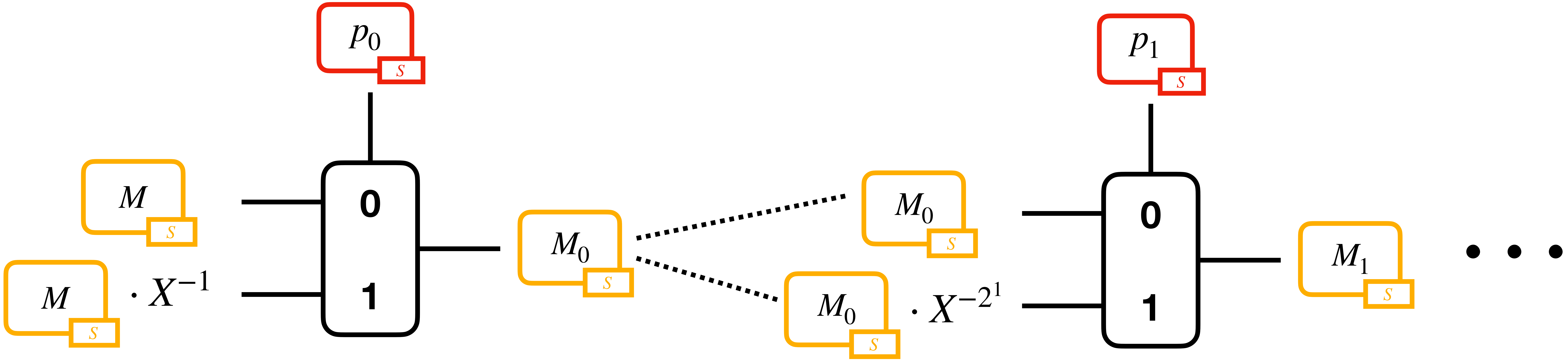


Blind Rotation

Rotate an **encrypted** polynomial M of p **encrypted** positions

$$p = p_0 \cdot 2^0 + \ldots + p_k \cdot 2^k$$



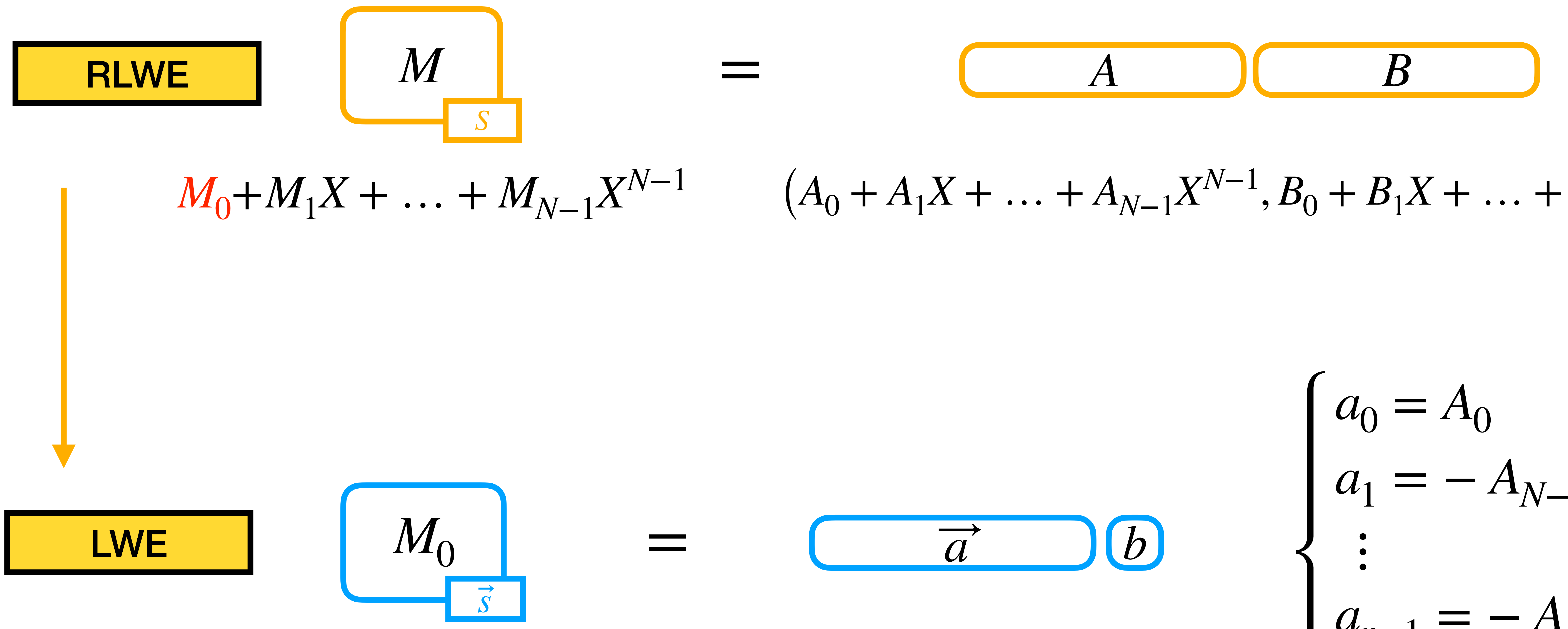


$$M_0 = M \cdot X^{-p_0}$$

$$M_1 = M_0 \cdot X^{-p_1 2}$$

Sample Extraction

$S = S_0 + S_1 X + \ldots + S_{N-1} X^{N-1}$



RLWE

$\boxed{M}_{S} = \boxed{A}\ \boxed{B}$

$M_0 + M_1 X + \ldots + M_{N-1} X^{N-1}$

$\left(A_0 + A_1 X + \ldots + A_{N-1} X^{N-1}, B_0 + B_1 X + \ldots + B_{N-1} X^{N-1}\right)$

LWE

$\boxed{M_0}_{\vec{s}} = \boxed{\vec{a}}\ \boxed{b}$

$$\begin{cases} a_0 = A_0 \\ a_1 = -A_{N-1} \\ \vdots \\ a_{n-1} = -A_1 \\ b = B_0 \end{cases}$$

$$\begin{cases} \vec{s} = (s_0 = S_0, \ldots, s_{n-1} = S_{N-1}) \\ n = N \end{cases}$$

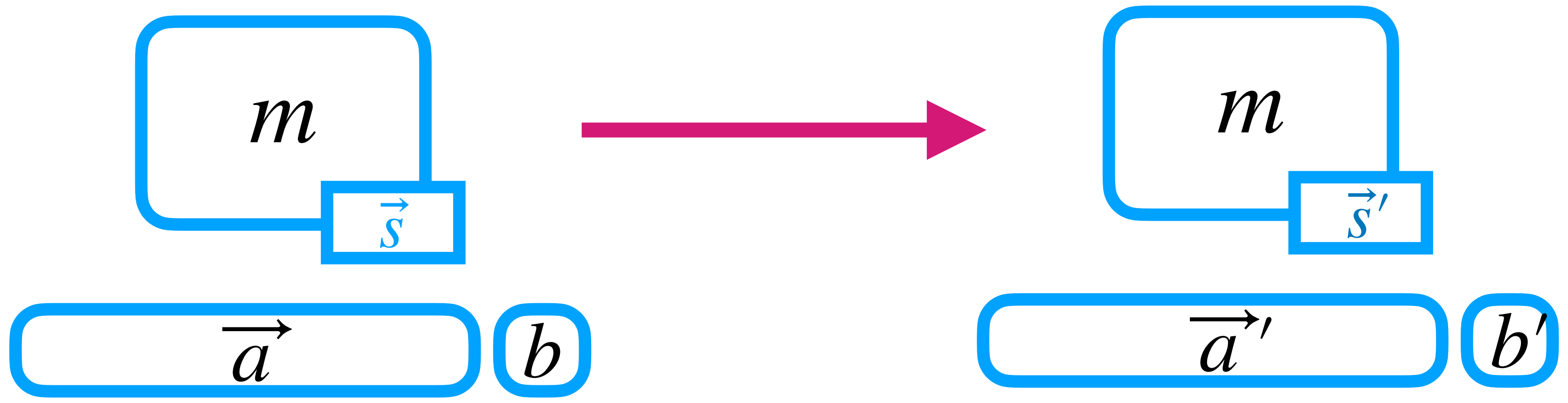
All the other coefficients can be extracted in a similar way

Key Switching

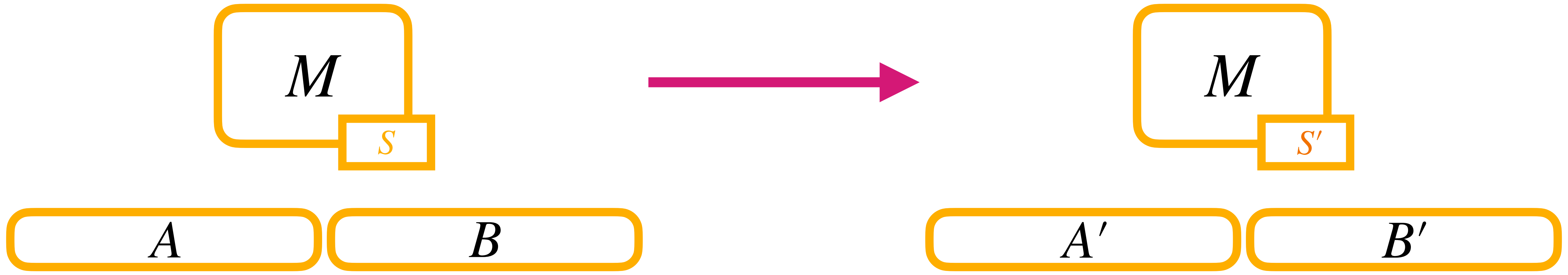
EVERY KEY SWITCHING...

- Needs a **key-switching key**
- Used to switch the key
- Used to switch the parameters
- Can be used to **evaluate** a very regular **function** (public or private)
- **Increases the noise**

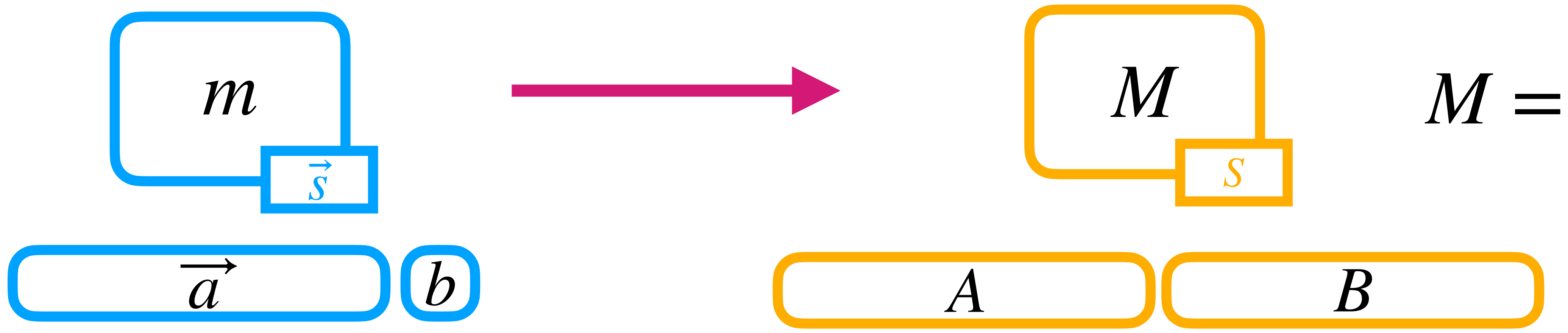
LWE to LWE



RLWE to RLWE

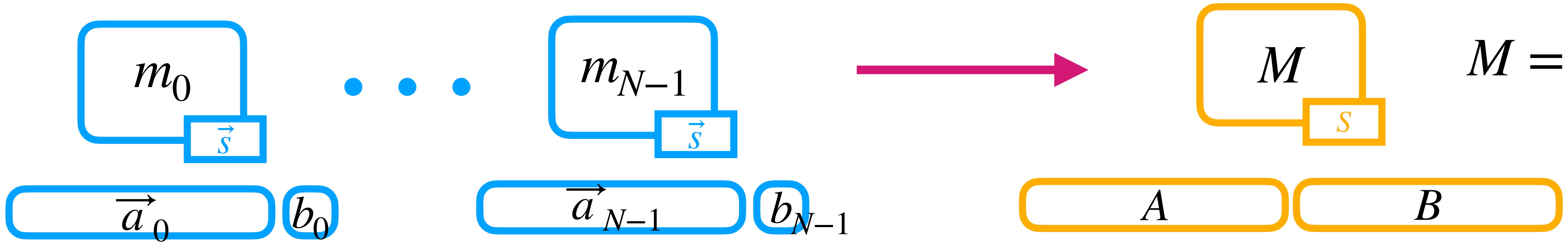


LWE to RLWE



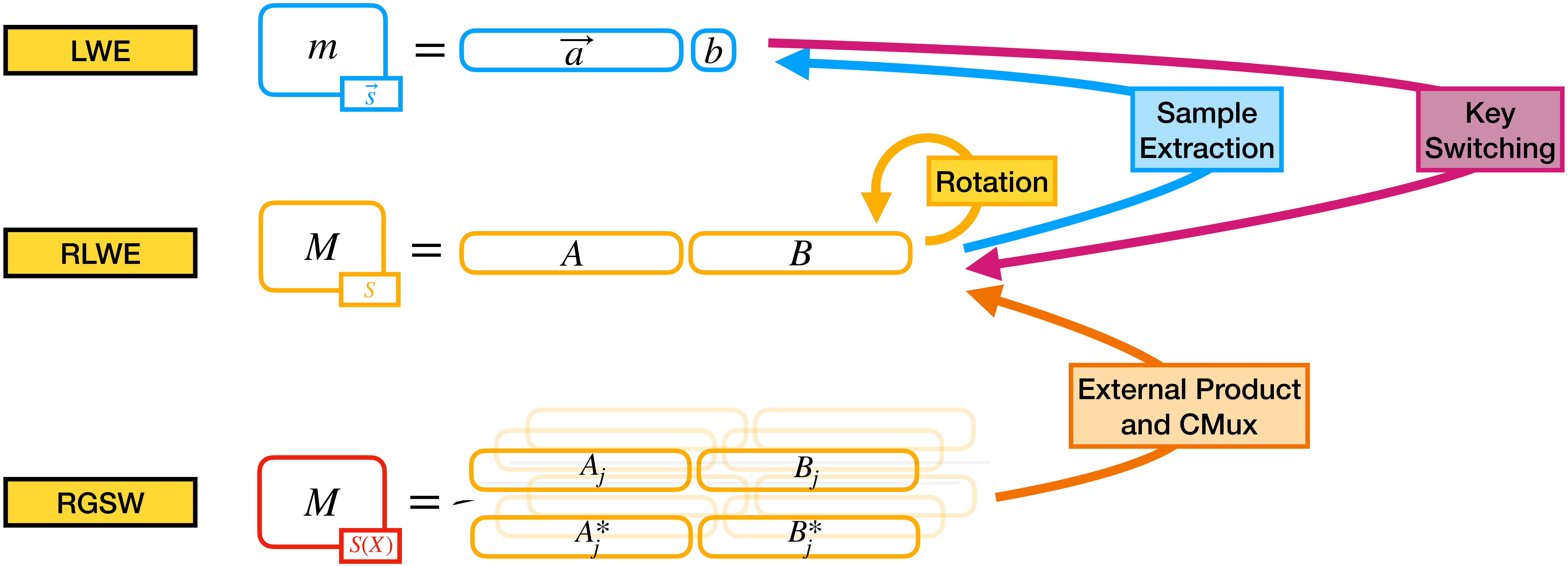
$M = m \cdot X^j$

many-LWE to 1-RLWE



$M = \sum_{i=0}^{N-1} m_i \cdot X^i$

Building Blocks: Summary



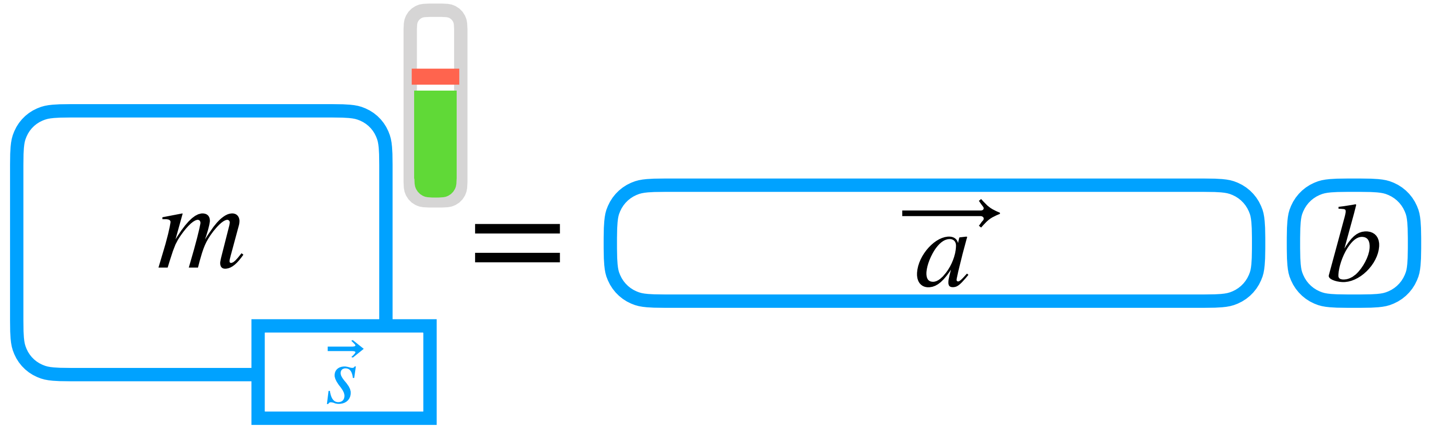
Overview

- What is FHE?
- A little bit of history
- FHE schemes based on LWE
- TFHE ciphertexts and operations
- **TFHE Bootstrapping**
- Implementations and applications

Bootstrapping

Original goal: reduce the noise when it grows too much

In TFHE, we can **bootstrap LWE ciphertexts**



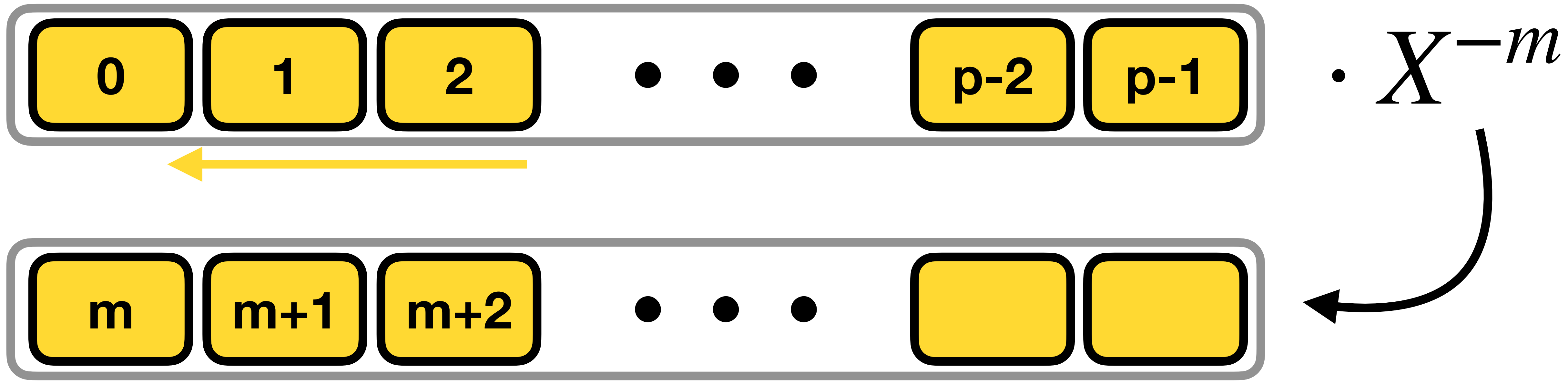
To bootstrap, we need to **evaluate the decryption**:

$$\begin{cases} \text{1} \quad b - \sum_{i=0}^{n-1} a_i \cdot s_i = \Delta m + e \\ \text{2} \quad \left\lceil \frac{\Delta m + e}{\Delta} \right\rfloor = m \end{cases}$$

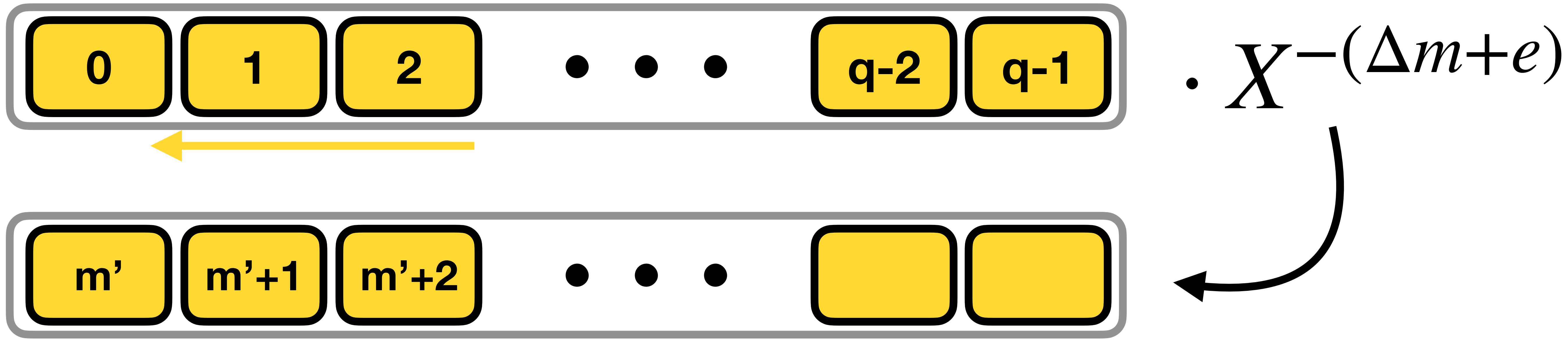
Bootstrapping

Let's start from **step 2** (the rounding of $\Delta m + e$)

$m \in \{0,1,\ldots,p-1\}$

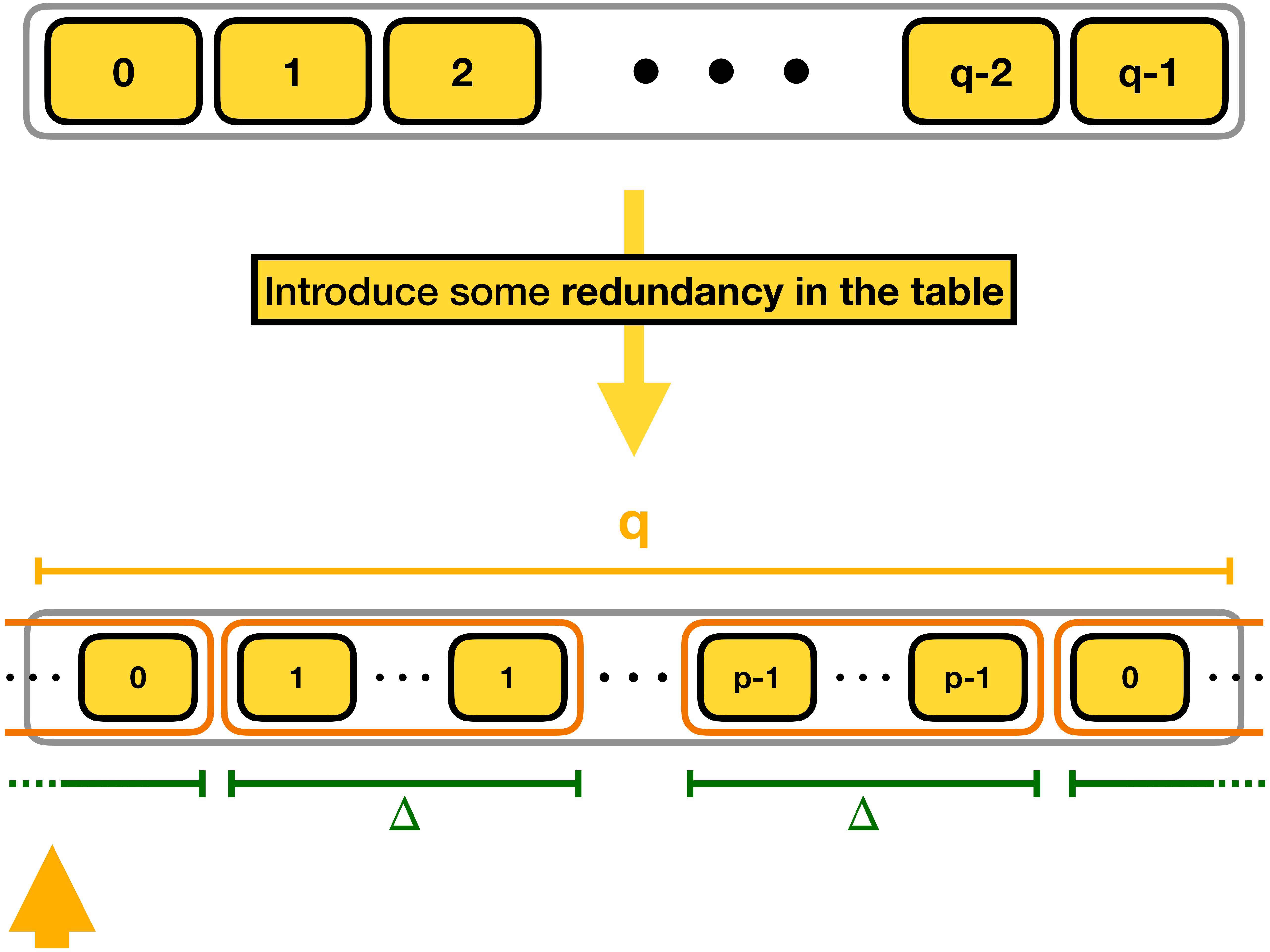


$m' = \Delta m + e \in \{0,1,\ldots,q-1\}$

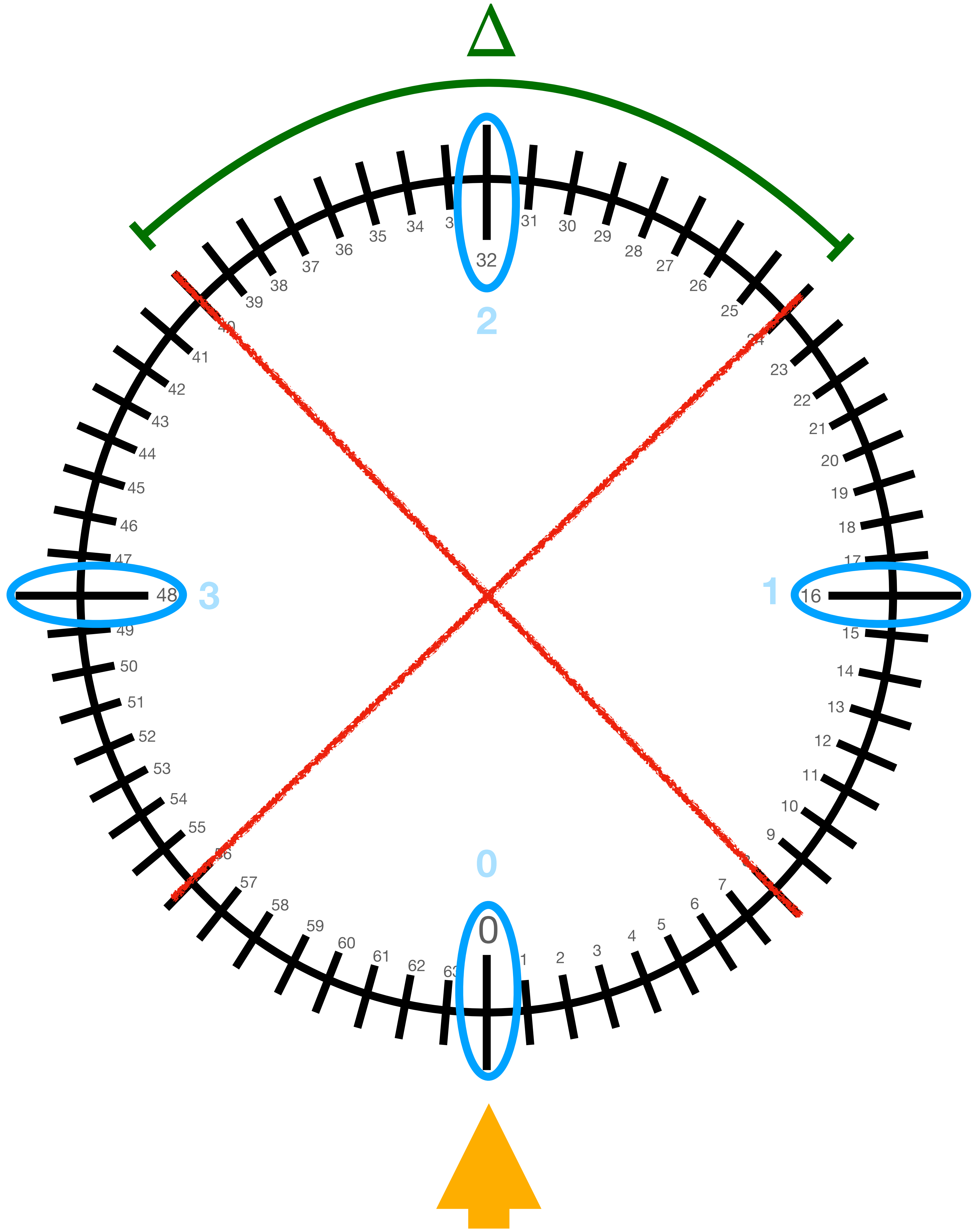


Bootstrapping

Let’s start from **step 2** (the rounding of $\Delta m + e$)



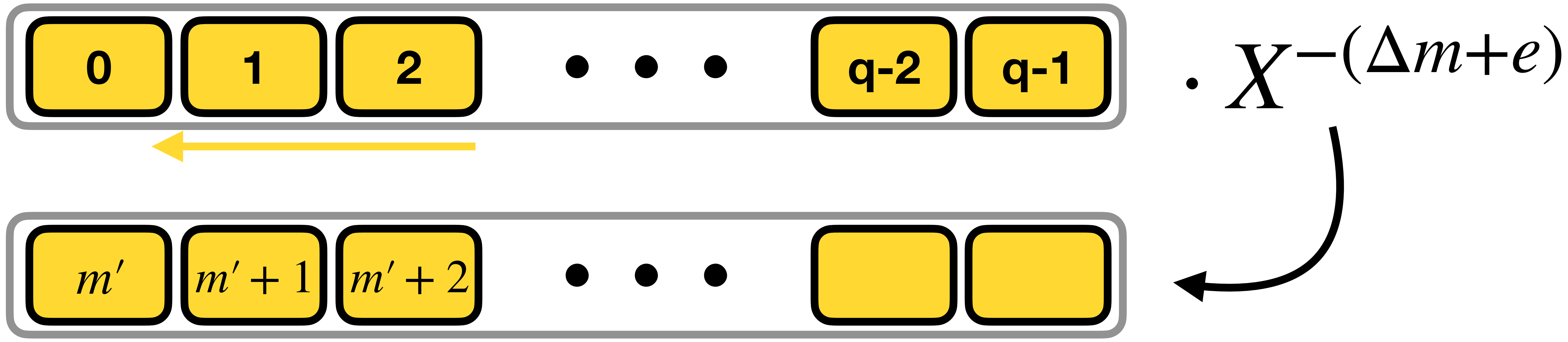
Introduce some **redundancy in the table**



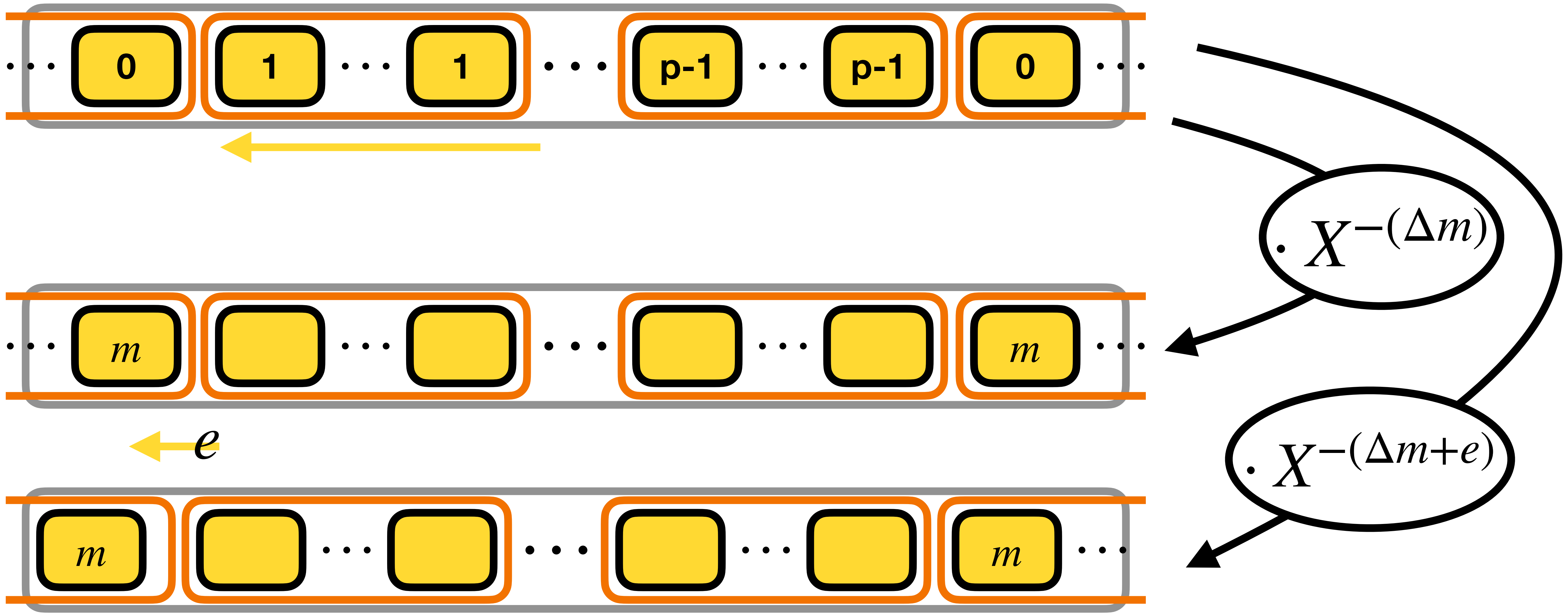
Bootstrapping

Let's start from **step 2** (the rounding of $\Delta m + e$)

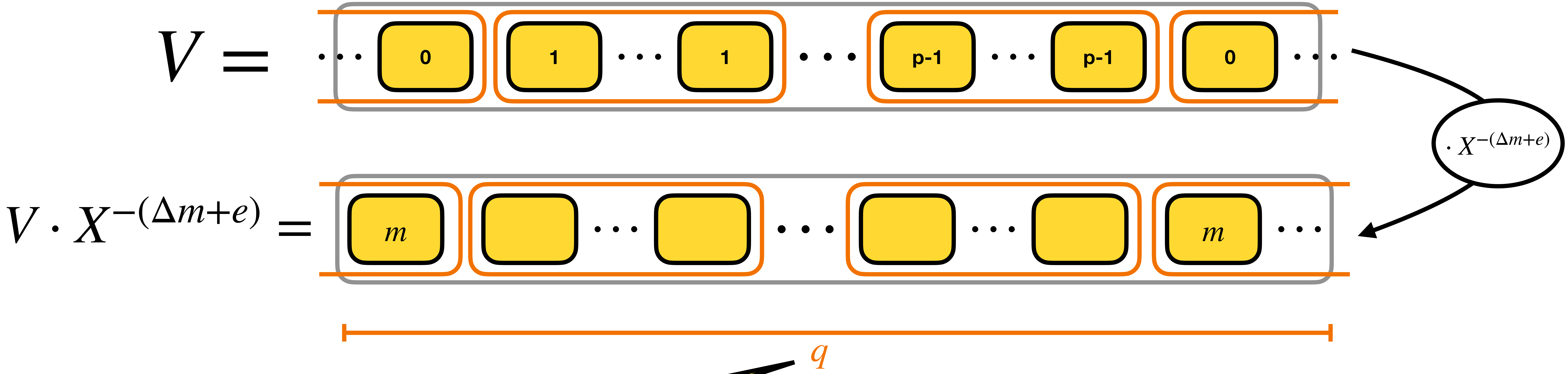
$$m' = \Delta m + e \in \{0,1,\ldots,q-1\}$$



$$\Delta m + e \to \left\lfloor \frac{\Delta m + e}{\Delta} \right\rceil = m$$



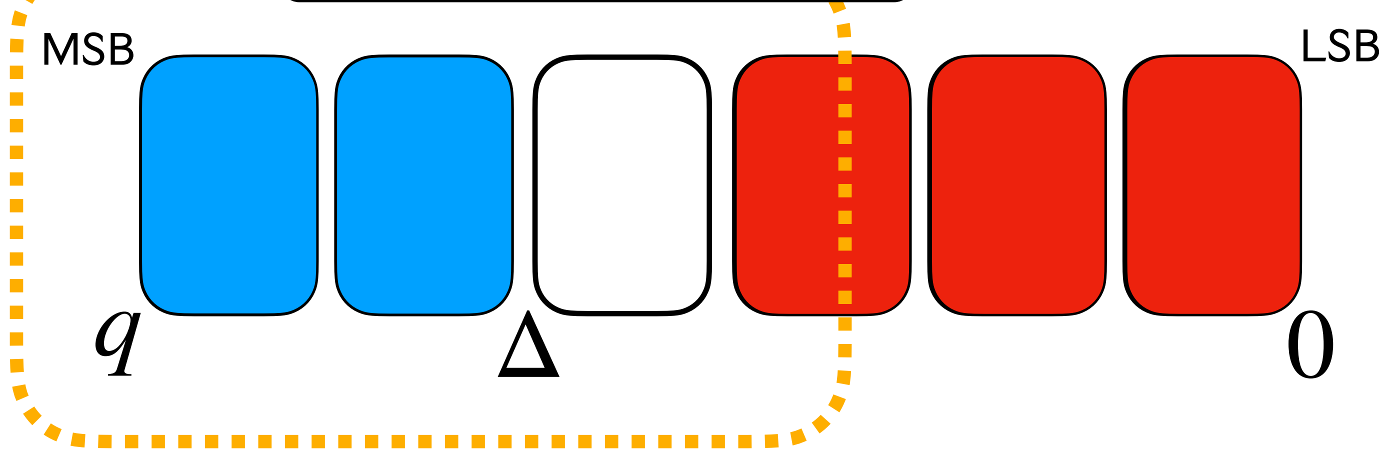
Bootstrapping



Using polynomials with $q = 2^{32}$ or $q = 2^{64}$ coefficients would be impractical!!!

Use instead polynomials with $N = 2^{10}$ (or a bit more) coefficients!

Use only the MSB

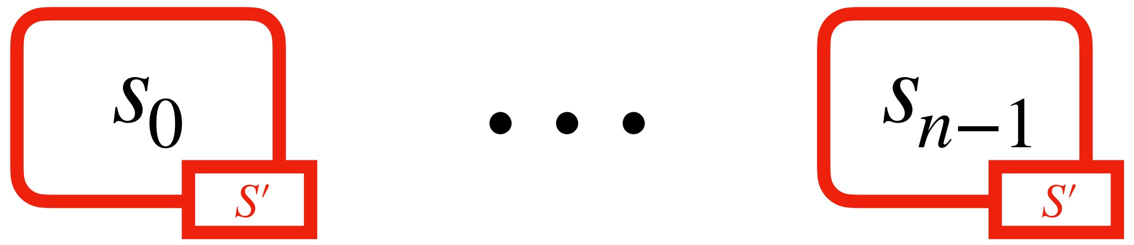


Bootstrapping

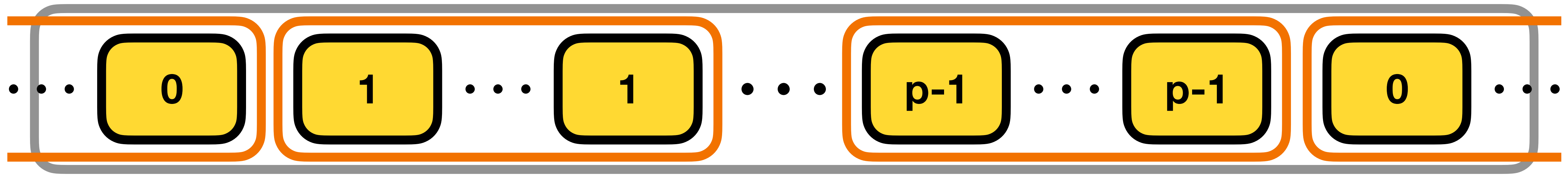
How to compute $V \cdot X^{-(\Delta m+e)}$?

$$-(\Delta m + e) = -b + \sum_{i=0}^{n-1} a_i \cdot s_i$$

$$= \boxed{-b} + \boxed{a_0 \cdot s_0} + \ldots + \boxed{a_{n-1} \cdot s_{n-1}}$$



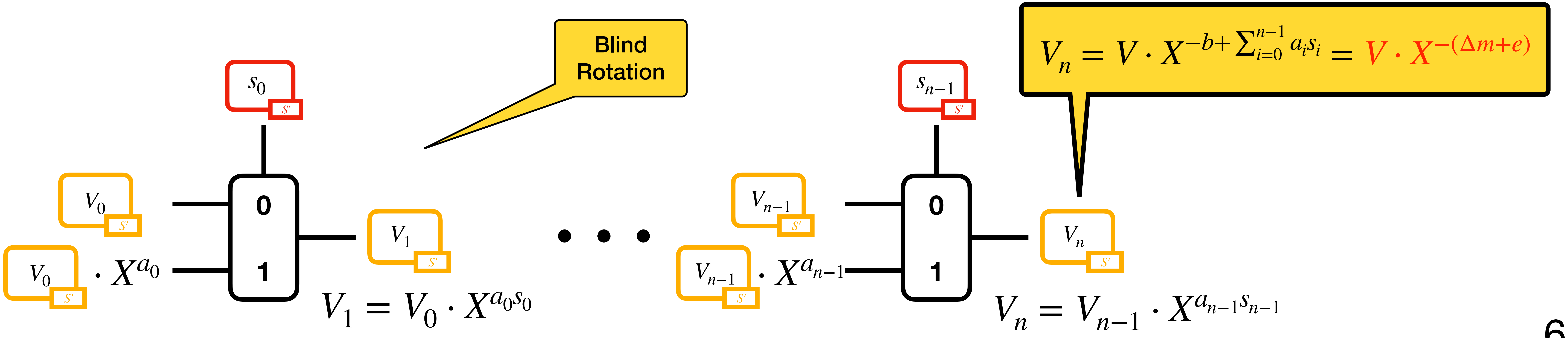
1 $V =$



2 $V_0 = V \cdot X^{-b}$

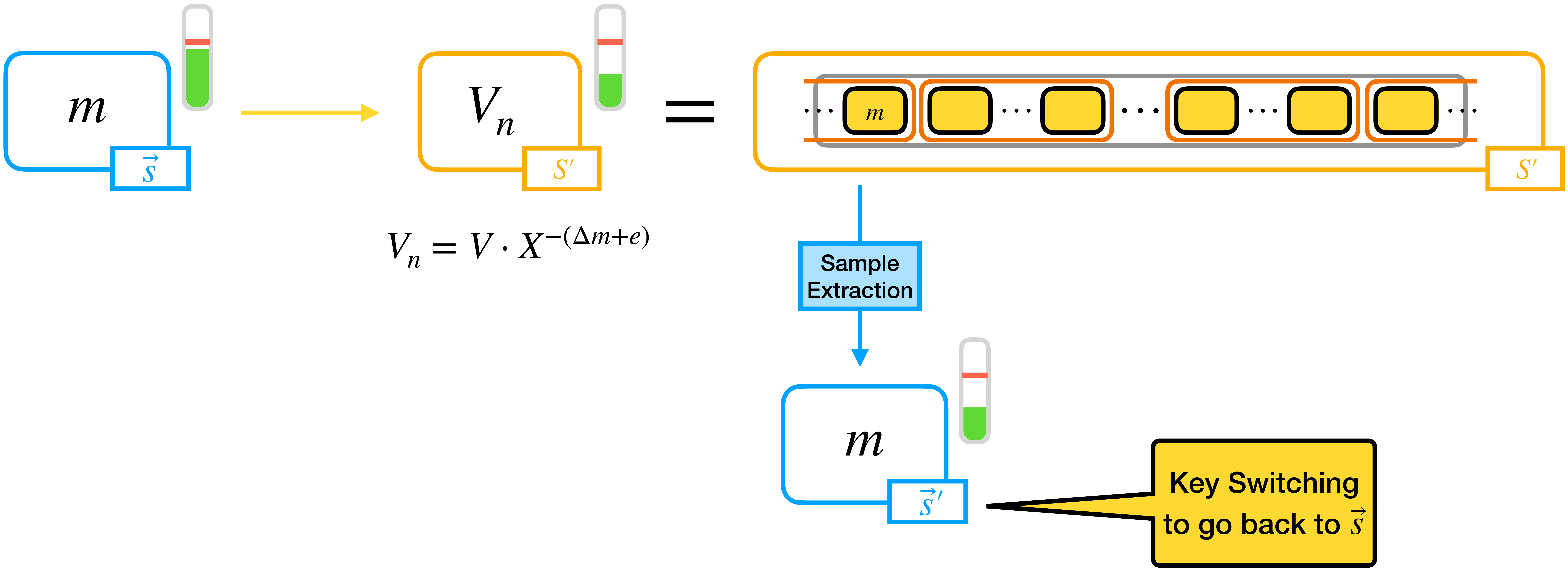
Rotation

3



Blind Rotation
$V_n = V \cdot X^{-b+\sum_{i=0}^{n-1} a_i s_i} = V \cdot X^{-(\Delta m+e)}$
s_0
S'
V_0
S'
V_0
S'
$\cdot X^{a_0}$
0
1
V_1
S'
$V_1 = V_0 \cdot X^{a_0 s_0}$
…
s_{n-1}
S'
V_{n-1}
S'
V_{n-1}
S'
$\cdot X^{a_{n-1}}$
0
1
V_n
S'
$V_n = V_{n-1} \cdot X^{a_{n-1} s_{n-1}}$

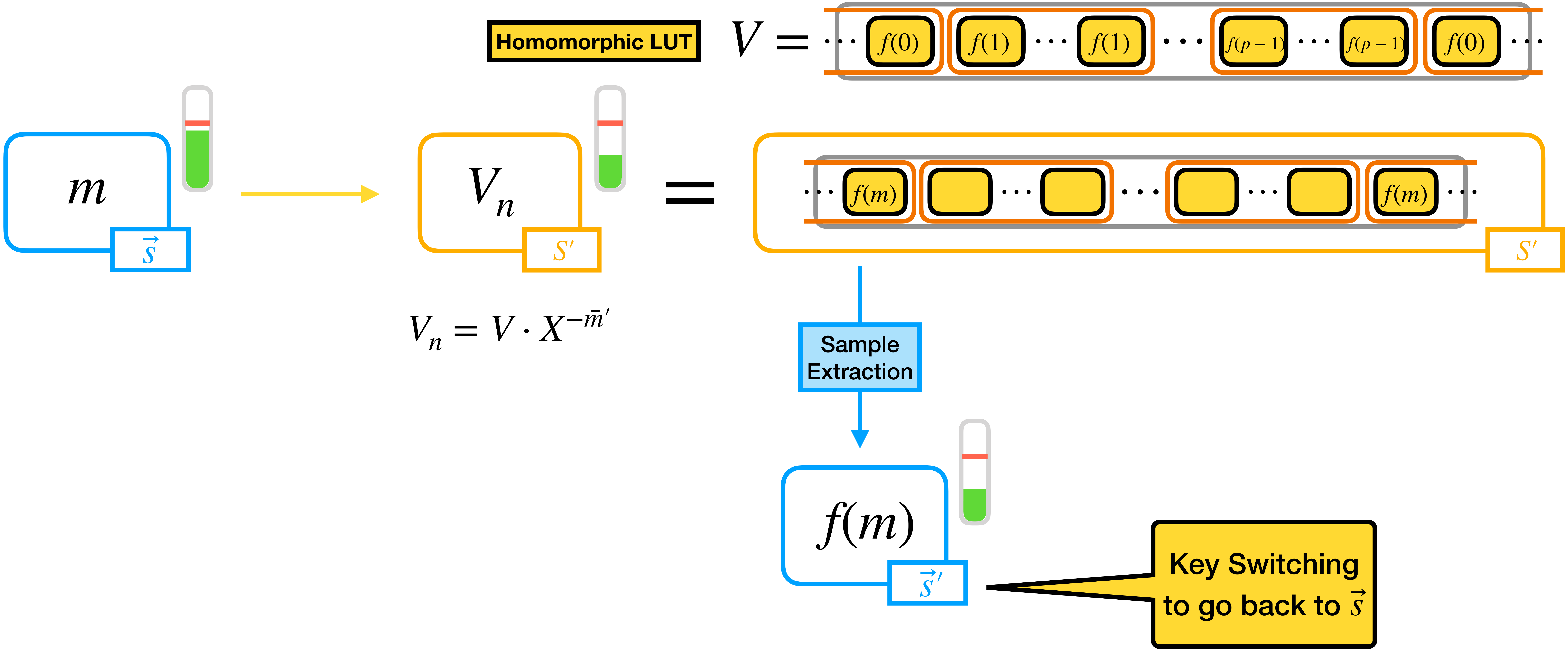
Bootstrapping



$$V_n = V \cdot X^{-(\Delta m + e)}$$

Bootstrapping

TFHE bootstrapping is "programmable": evaluates a function while reducing the noise



Homomorphic LUT

$$V = \cdots f(0)\; f(1) \cdots f(1) \cdots f(p-1) \cdots f(p-1)\; f(0) \cdots$$

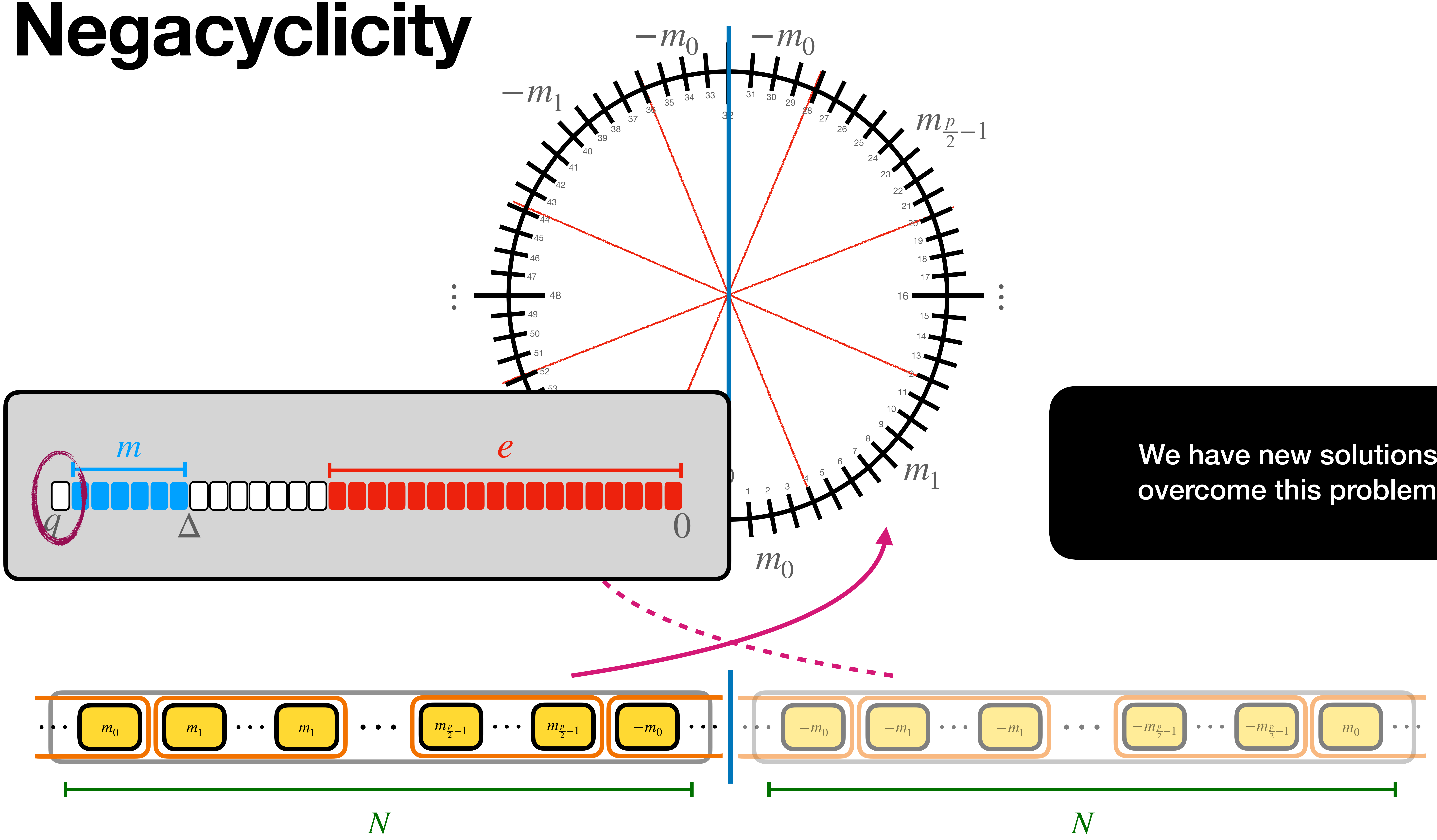
$$V_n = V \cdot X^{-\bar{m}'}$$

Sample Extraction

Key Switching to go back to $\vec{s}$

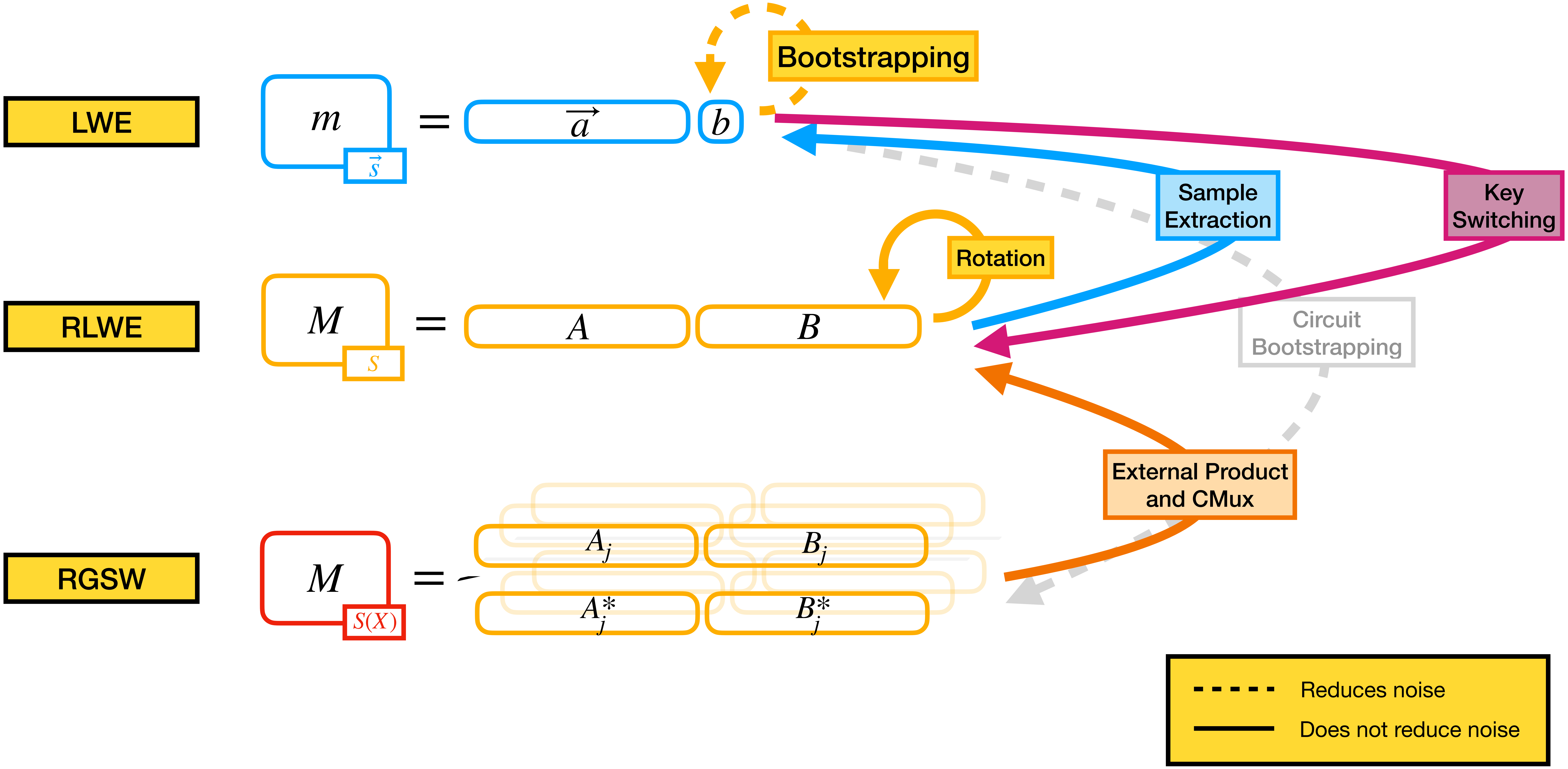
I lied a little bit… 🤥

Negacyclicity



We have new solutions to overcome this problem 😉

Bootstrapping: Summary



Other features in TFHE

- How to do **Gate Bootstrapping** 0️⃣1️⃣
- **Leveled evaluation of LUT** with vertical and horizontal packing 📊
- Evaluate deterministic (weighted) **finite automata** 🔗
- Homomorphic counter **TBSR** 🧮
- **Circuit bootstrapping** 🔋
- **New WoP-PBS** ⚙️
- And more …

Overview

- What is FHE?
- A little bit of history
- FHE schemes based on LWE
- TFHE ciphertexts and operations
- TFHE Bootstrapping
- **Implementations and applications**

Some open source implementations

HEAAN

TFHE

FHEW

LATTIGO

Microsoft SEAL

HELib

Concrete

PALISADE

There exists also some GPU implementations

Open source implementations

TFHE: bootstrapped binary circuits
Experimental TFHE: circuit bootstrapping (binary)

Concrete: (programmable) bootstrapping,
binary-integer-real encodings
noise tracking…

More than a library

Some applications

MPC

💁‍♀️💁🏾💁🏻‍♂️

E-voting

💻🗳️

Multi-key TFHE

🔑🗝️

Blockchain

🧱⛓️

Statistics over sensitive data

📊📈

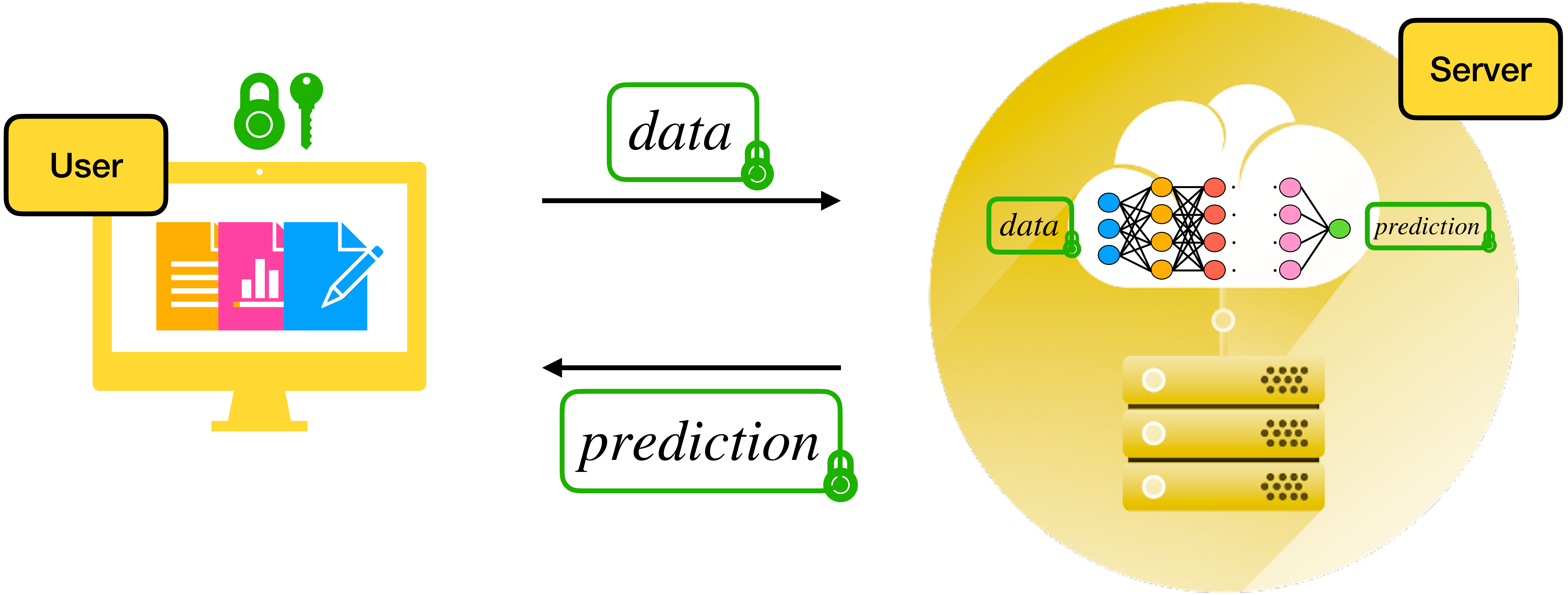
Machine Learning

🧠🤖

Machine Learning

- Inference over encrypted data -

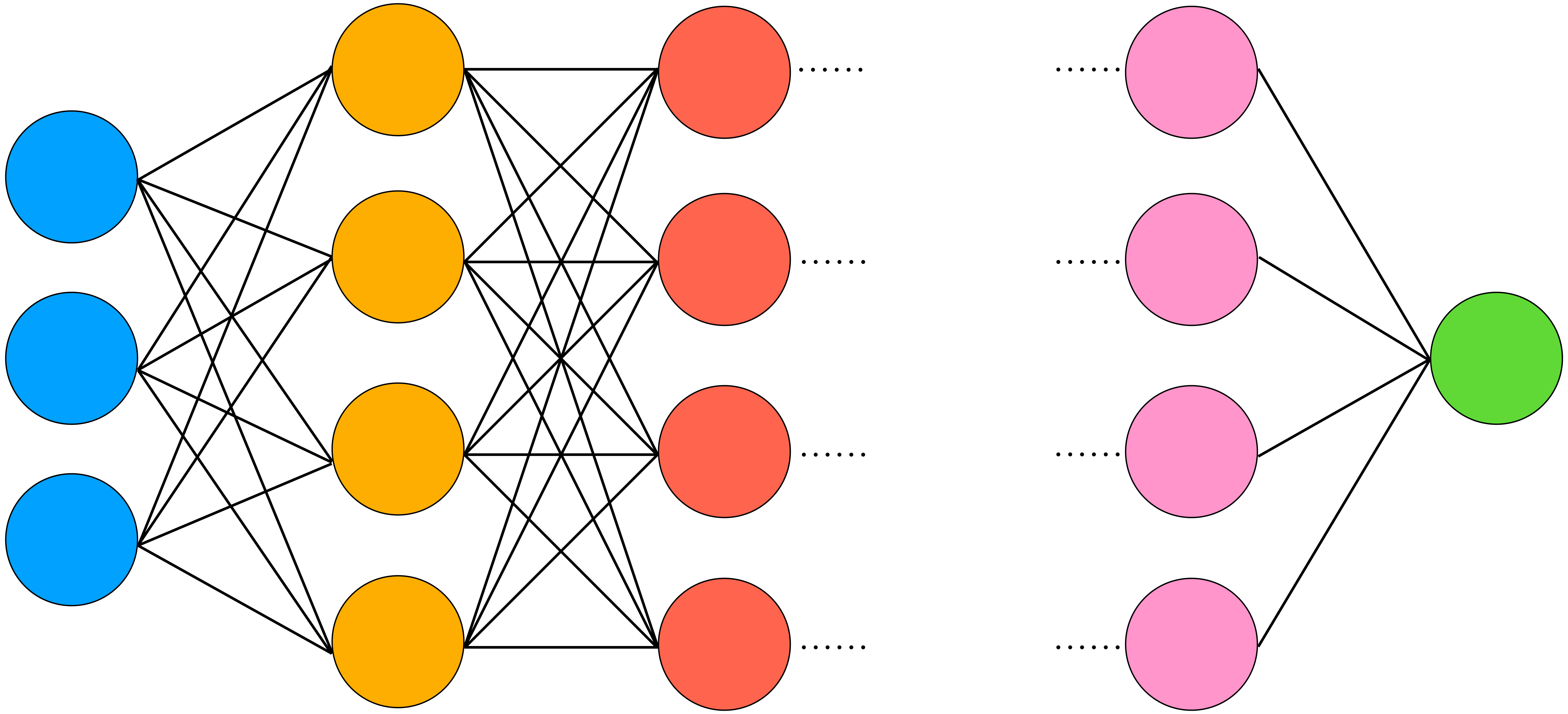
Empowering machine learning with FHE



Data stays encrypted during all the process!
The server learns nothing

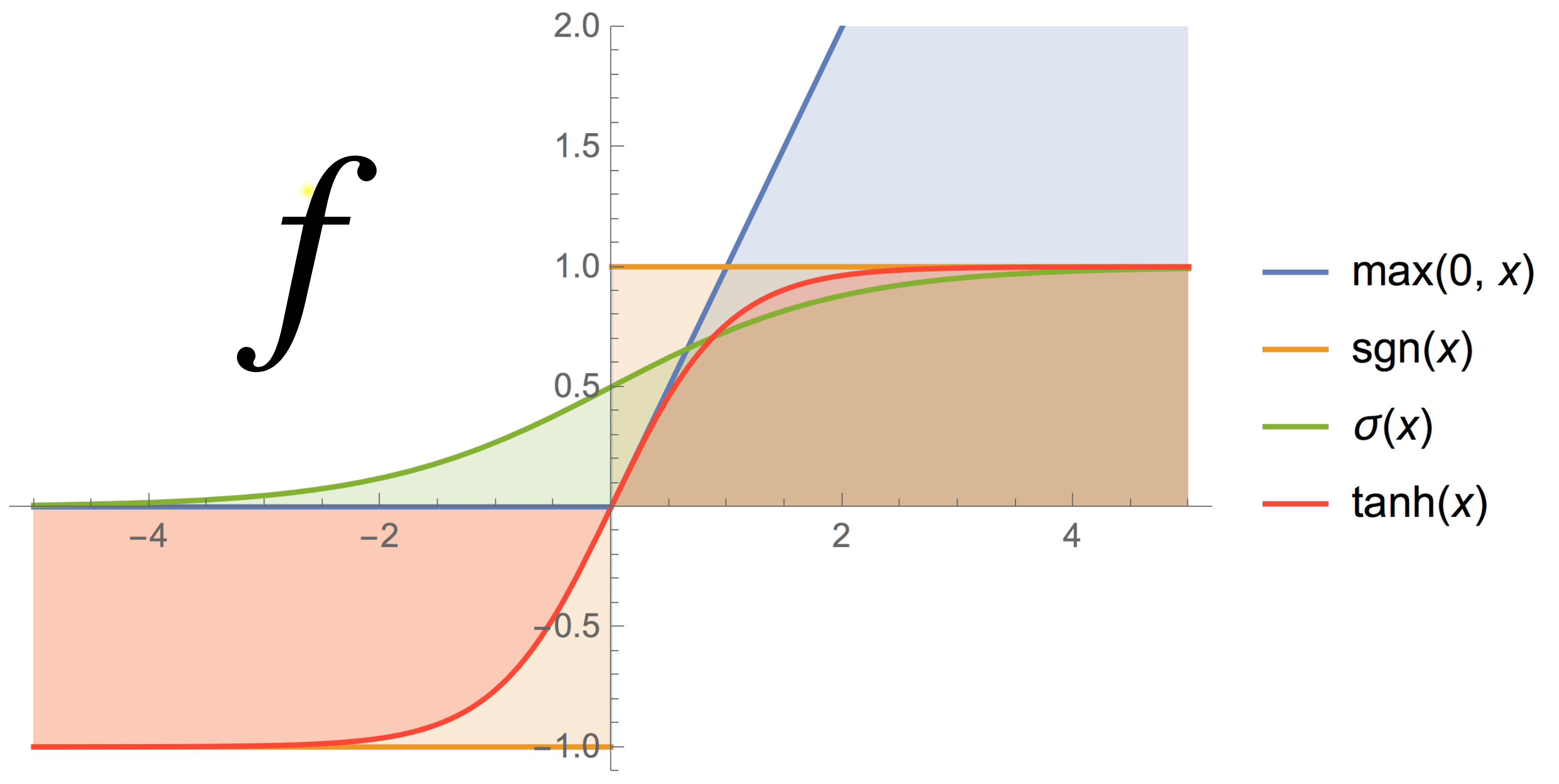
Machine learning applications 🧠

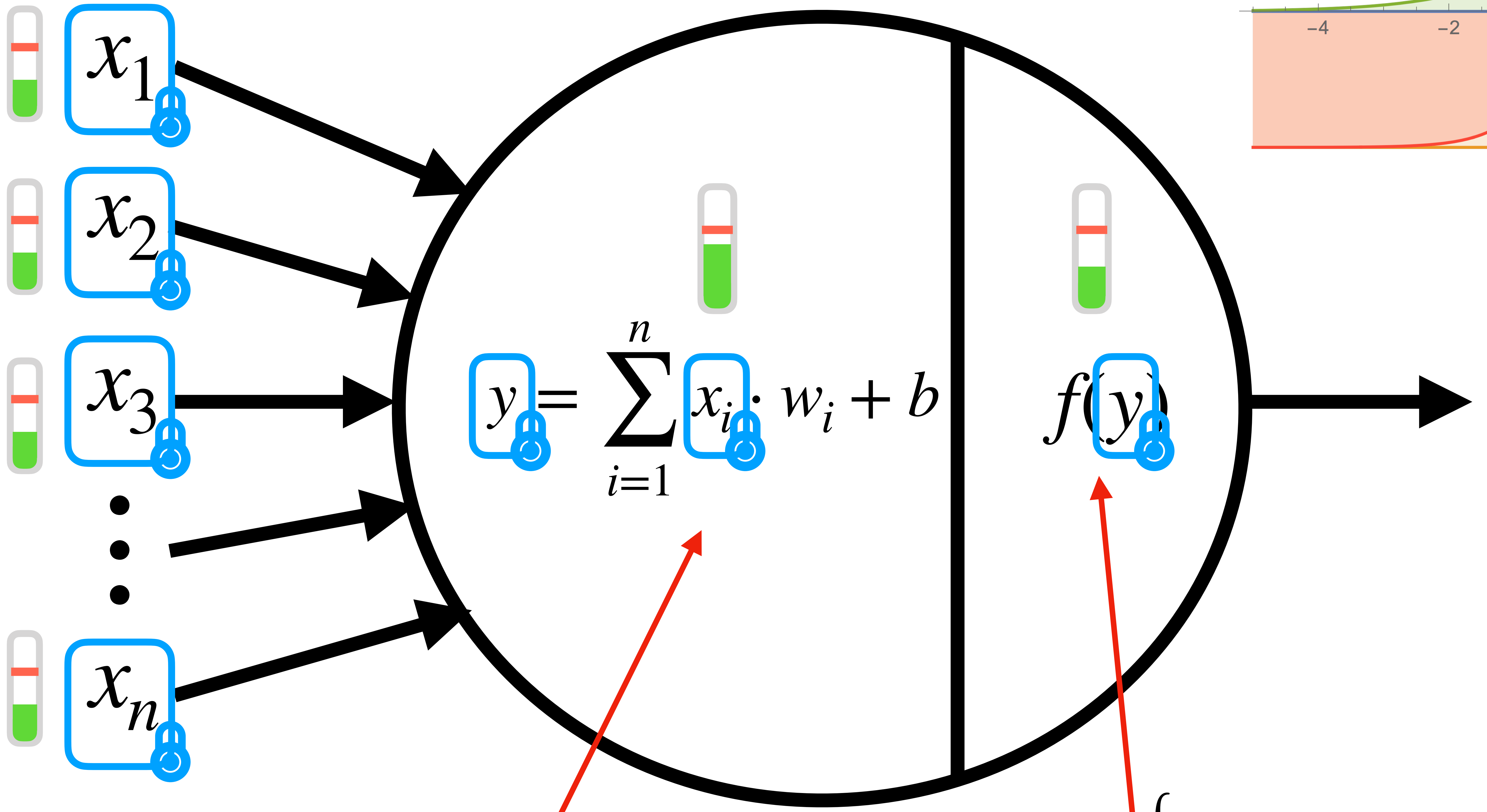
Neural network



Many type of layers: dense, convolution, activation, pooling, etc.
In FHE: different operations with different costs.

Artificial neuron 🧠





No depth limitations: Inference of deep NN

Homomorphic Addition (discretized weights)

BGV-like: approximate with polynomial

TFHE-like: programmable bootstrapping

Let’s be Concrete

https://concrete.zama.ai/

Some experiments: NN-20

[CJP21] “Programmable Bootstrapping Enables Efficient Homomorphic Inference of Deep Neural Networks”
I. Chillotti, M. Joye and P. Paillier, CSCML 2021

MNIST dataset

	Accuracy	CPU	AWS	AWS2	
in the clear NN-20	97.5%	0.17 ms	0.19 ms		
NN-20 homomorphic	97.5%	30.04 s	6.19 s	2.10 s	80 bits of security
	97.1%	115.52 s	21.17 s	7.53 s	128 bits of security

~ 100 active neurons per layer

- CPU: PC with 2.6 GHz 6-Core Intel ® Core™ i7 processor,
- AWS: a 3.00 GHz Intel ® Xeon ® Platinum 8275CL processor with 96 vCPUs hosted on AWS
- AWS2: as above but with 8 NVIDIA ® A100 Tensor Core GPUs

Some experiments: NN-50

[CJP21] "Programmable Bootstrapping Enables Efficient Homomorphic Inference of Deep Neural Networks"
I. Chillotti, M. Joye and P. Paillier, CSCML 2021

MNIST dataset

	Accuracy	CPU	AWS	AWS2	
in the clear NN-50	95.4%	0.20 ms	0.30 ms		
NN-50 homomorphic	95.1%	71.71 s	13.00 s	5.27 s	80 bits of security
	94.7%	233.55 s	43.91 s	18.89 s	128 bits of security

~ 100 active neurons per layer

- CPU: PC with 2.6 GHz 6-Core Intel ® Core™ i7 processor,
- AWS: a 3.00 GHz Intel ® Xeon ® Platinum 8275CL processor with 96 vCPUs hosted on AWS
- AWS2: as above but with 8 NVIDIA ® A100 Tensor Core GPUs

Conclusion

What we learned?

What's next in FHE?

Bibliography

[Reg05] O. Regev. On lattices, learning with errors, random linear codes, and cryptography. STOC 2005.

[SSTX09] D. Stehlé, R. Steinfeld, K. Tanaka, K. Xagawa. Efficient public key encryption based on ideal lattices. ASIACRYPT 2009.

[LPR10] V. Lyubashevsky, C. Peikert, O. Regev. On ideal lattices and learning with errors over rings. EUROCRYPT 2010.

[Gen09] C. Gentry. Fully homomorphic encryption using ideal lattices. STOC 2009.

[RAD78] R. L. Rivest, L. Adleman, M. L. Dertouzos. On data banks and privacy homomorphisms. Foundations of secure computation 1978.

[DGHV10] M. van Dijk, C. Gentry, S. Halevi, V. Vaikuntanathan. Fully homomorphic encryption over the integers. EUROCRYPT 2010.

[BGV12] Z. Brakerski, C. Gentry, V. Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. ITCS 2012.

[Bra12] Z. Brakerski. Fully homomorphic encryption without modulus switching from classical gapsvp. CRYPTO 2012.

[FV12] J. Fan, F. Vercauteren. Somewhat practical fully homomorphic encryption. IACR Cryptology ePrint Archive, 2012.

[CKKS17] J. H. Cheon, A. Kim, M. Kim, Y. Song. Homomorphic encryption for arithmetic of approximate numbers. ASIACRYPT 2017.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. CRYPTO 2013.

[DM15] L. Ducas, D. Micciancio. FHEW: bootstrapping homomorphic encryption in less than a second. EUROCRYPT 2015.

[CGGI16] I. Chillotti, N. Gama, M. Georgieva, M. Izabachène. Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. ASIACRYPT 2016.

[CGGI17] I. Chillotti, N. Gama, M. Georgieva, M. Izabachène. Faster Packed Homomorphic Operations and Efficient Circuit Bootstrapping for TFHE. ASIACRYPT 2017.

[CGGI20] I. Chillotti, N. Gama, M. Georgieva, M. Izabachène. TFHE: Fast Fully Homomorphic Encryption over the Torus. Journal of Cryptology 2020.

Thank you 🙏

❓ Q&A