Loris Bergerat[1,2], Ilaria Chillotti, Damien Ligier, Jean-Baptiste Orfila[1],
Adeline Roux-Langlois[2] & Samuel Tap[1]

12. 09. 2024

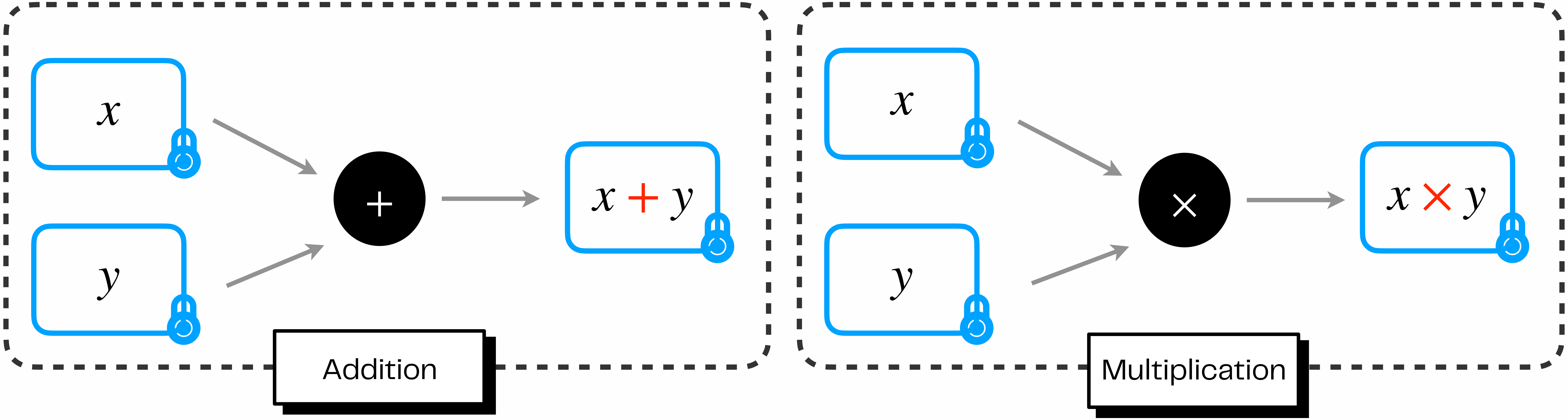
New Secret Keys for Enhanced Performance in (T)FHE

FHE.org Meetup

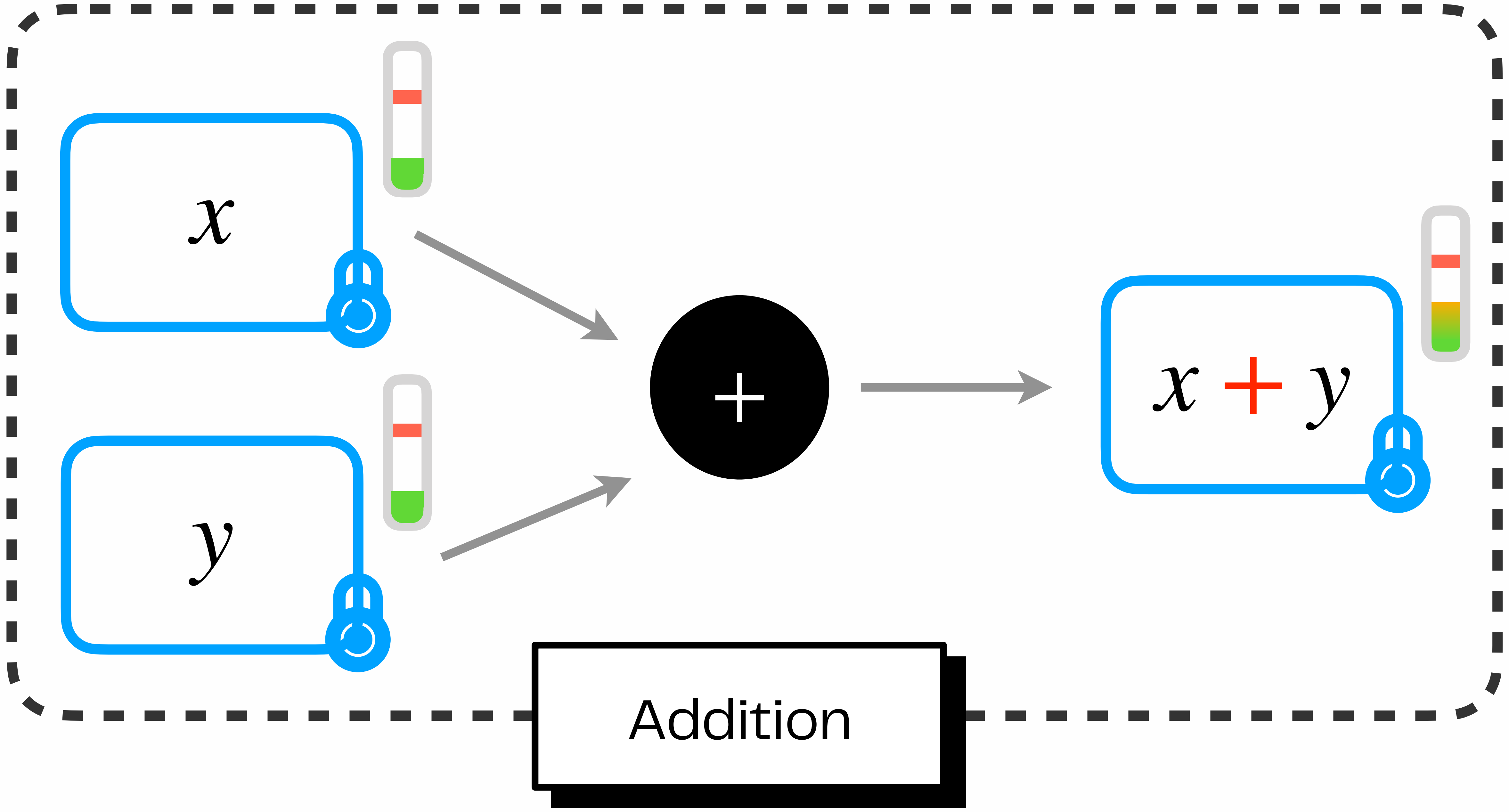
[1]Zama
[2]CNRS, GREYC,Caen

Introduction

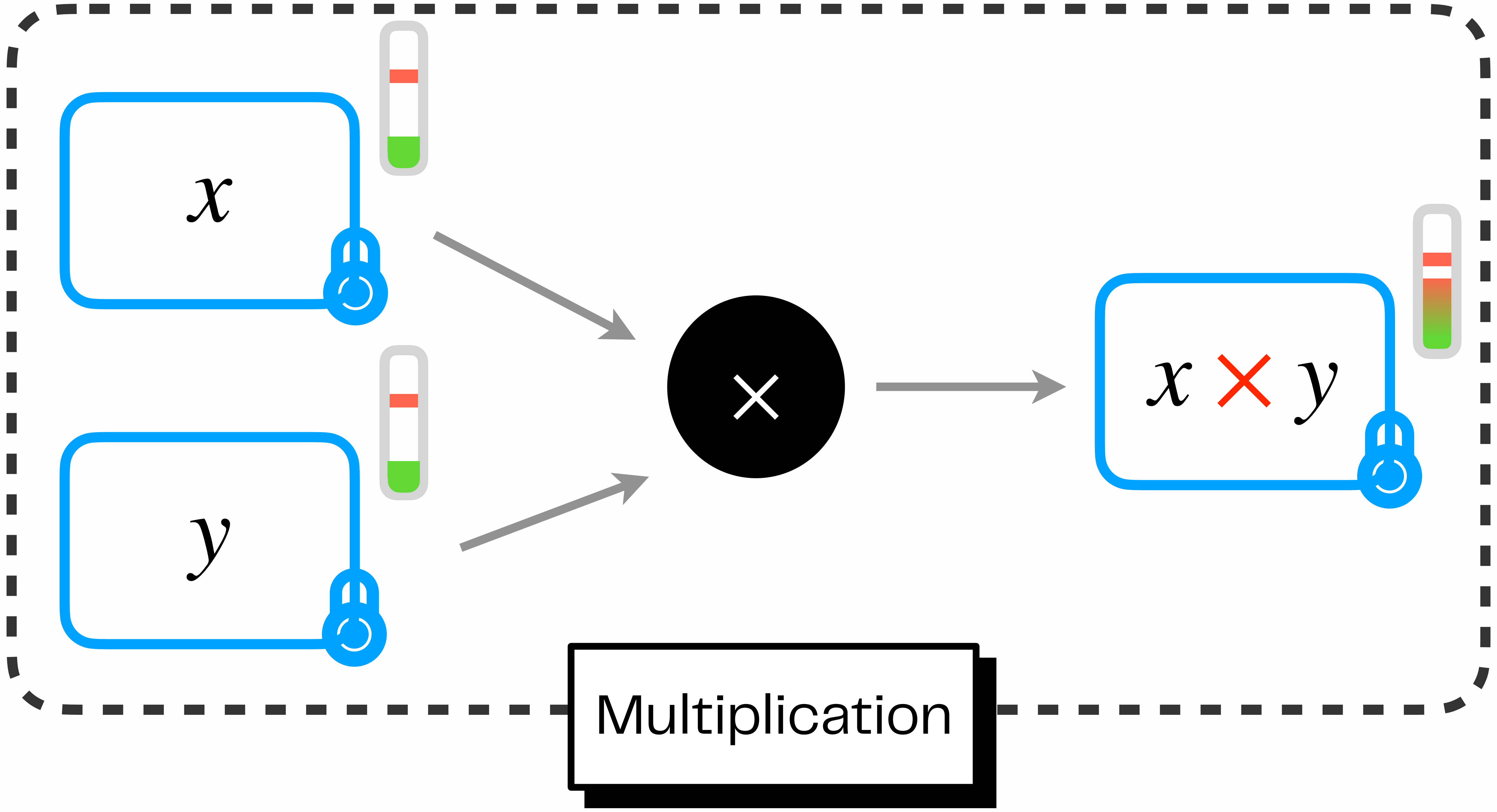
Fully Homomorphic Encryption (FHE)



Fully Homomorphic Encryption (FHE)

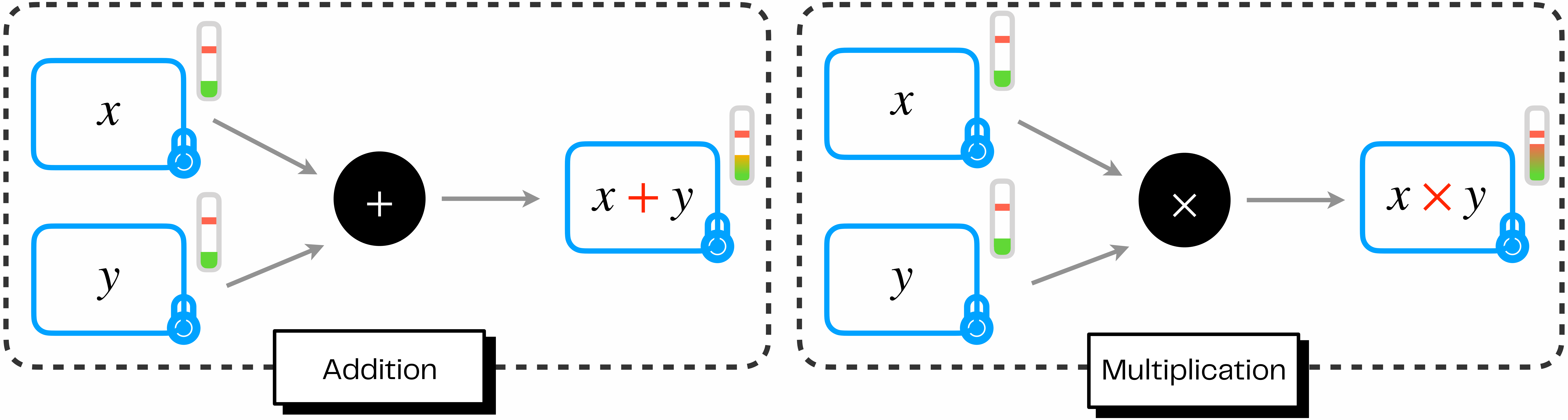


Addition



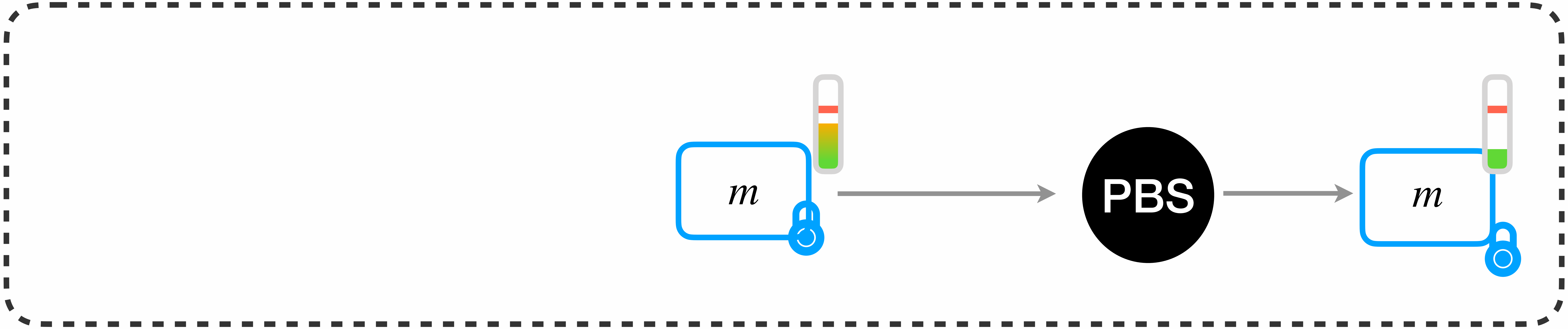
Multiplication

Fully Homomorphic Encryption (FHE)



too much noise $\Longrightarrow$ incorrect decryption

TFHE Bootstrapping Graph

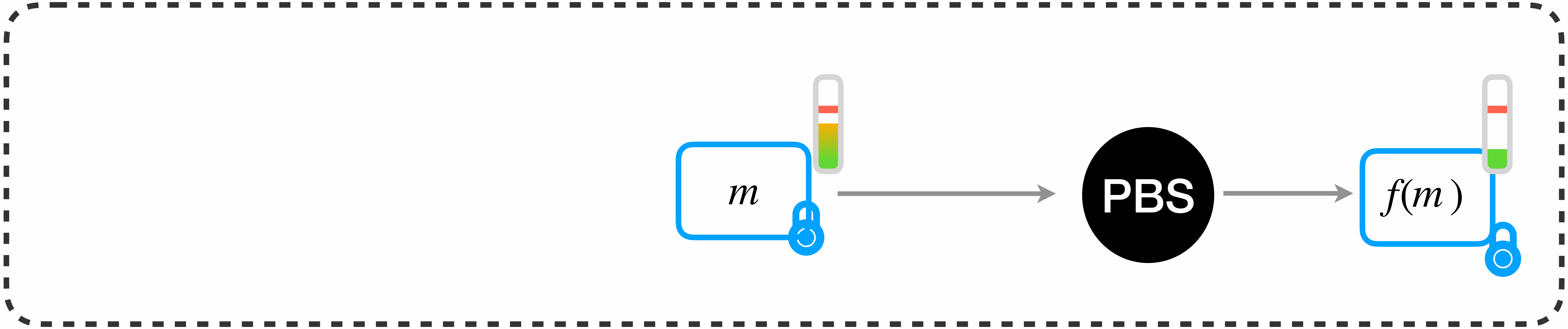


Bootstrapping

Reduce the noise

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC2009.
[CJP21] Ilaria Chillotti, Marc Joye, and Pascal Paillier. Programmable bootstrapping enables efficient homomorphic inference of deep neural networks. In CSCML 2021.

TFHE Bootstrapping Graph



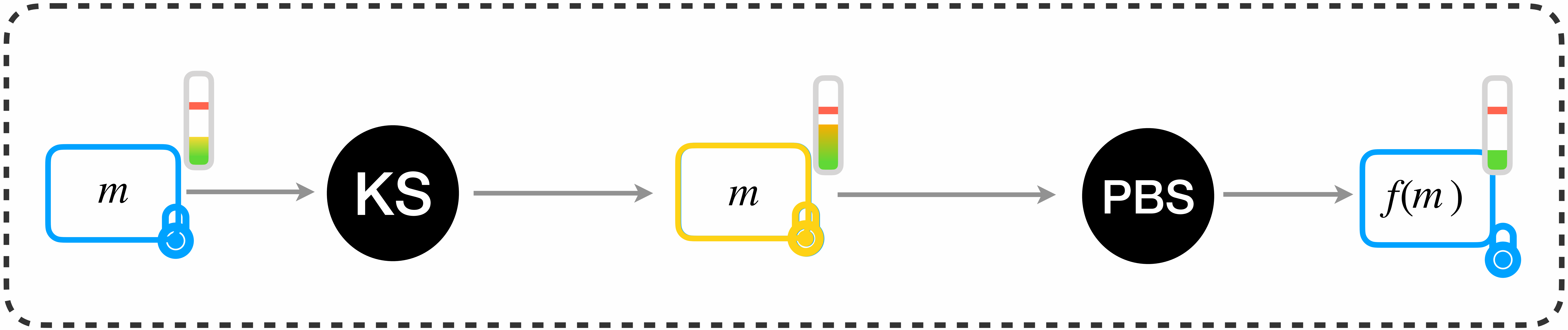
Programmable Bootstrapping

Reduce the noise

Evaluate univariate function

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC2009.
[CJP21] Ilaria Chillotti, Marc Joye, and Pascal Paillier. Programmable bootstrapping enables efficient homomorphic inference of deep neural networks. In CSCML 2021.

TFHE Bootstrapping Graph



Keyswitch

Switch from a secret key to another smaller secret key

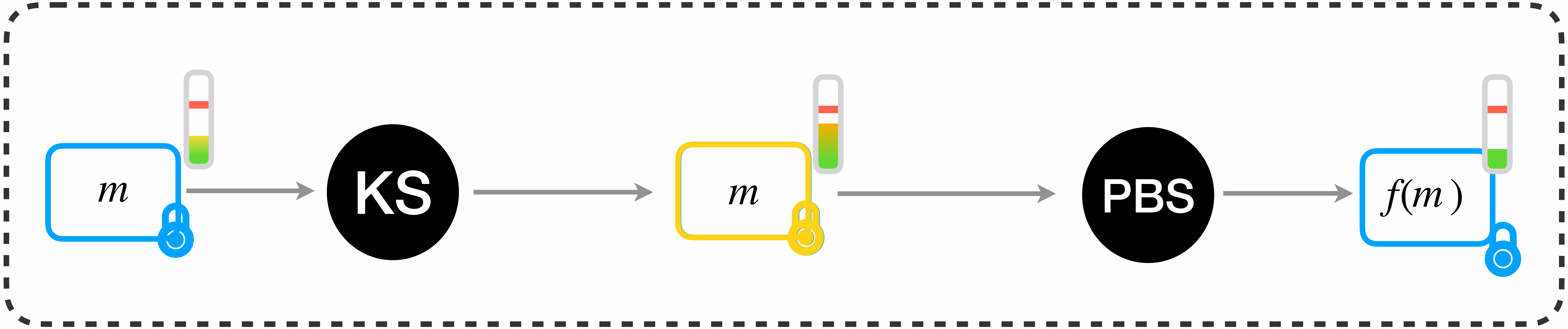
Programmable Bootstrapping

Reduce the noise

Evaluate univariate function

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC2009.
[CJP21] Ilaria Chillotti, Marc Joye, and Pascal Paillier. Programmable bootstrapping enables efficient homomorphic inference of deep neural networks. In CSCML 2021.

TFHE Bootstrapping Graph



Keyswitch

Switch from a secret key to another smaller secret key

Smaller Secret Key →

⇒ **Faster PBS**

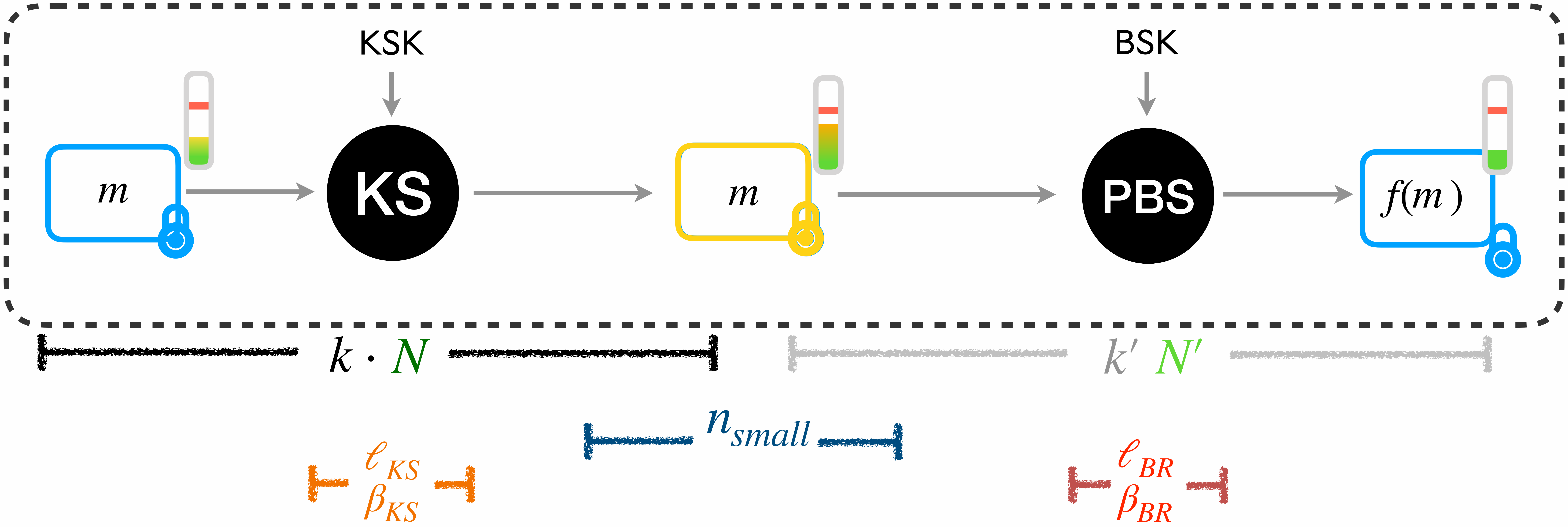
Programmable Bootstrapping

Reduce the noise

Evaluate univariate function

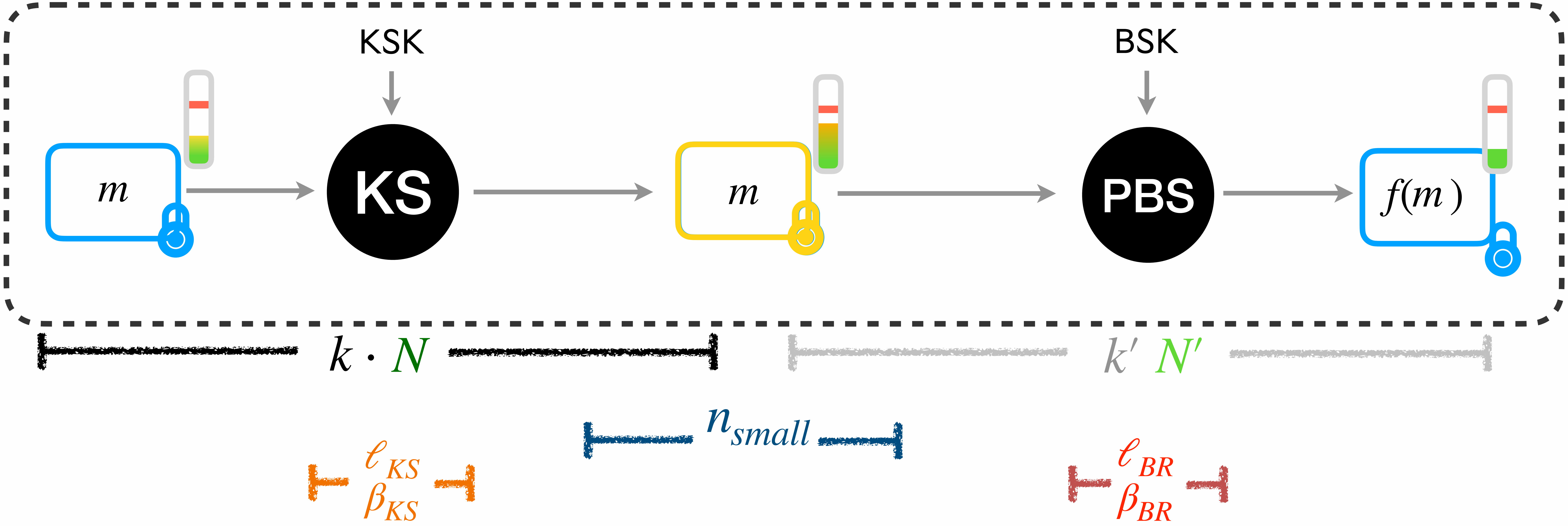
[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC2009.
[CJP21] Ilaria Chillotti, Marc Joye, and Pascal Paillier. Programmable bootstrapping enables efficient homomorphic inference of deep neural networks. In CSCML 2021.

TFHE Bootstrapping Graph



[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC2009.
[CJP21] Ilaria Chillotti, Marc Joye, and Pascal Paillier. Programmable bootstrapping enables efficient homomorphic inference of deep neural networks. In CSCML 2021.

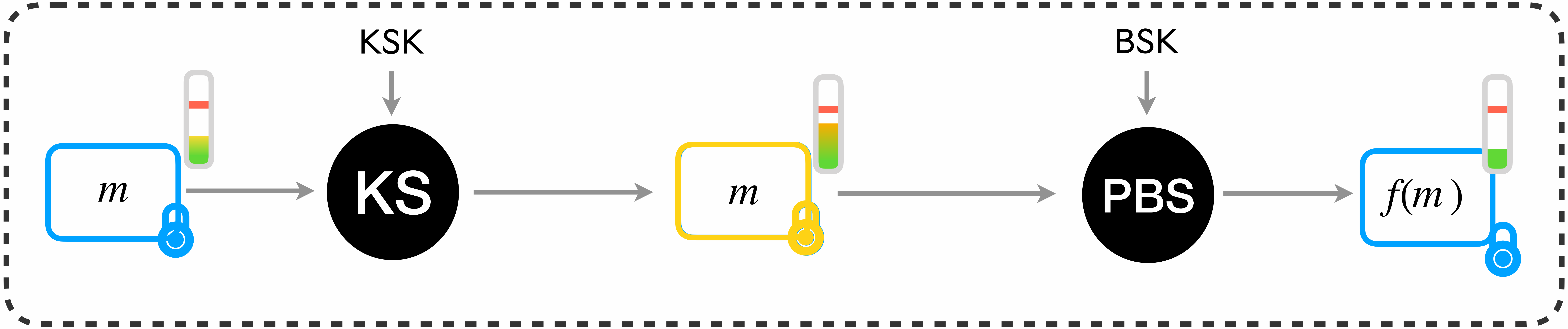
TFHE Bootstrapping Graph



Changing **one parameter** impacts:
the **security**, the **correctness**, the **other parameters** and the **execution time**.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC2009.
[CJP21] Ilaria Chillotti, Marc Joye, and Pascal Paillier. Programmable bootstrapping enables efficient homomorphic inference of deep neural networks. In CSCML 2021.

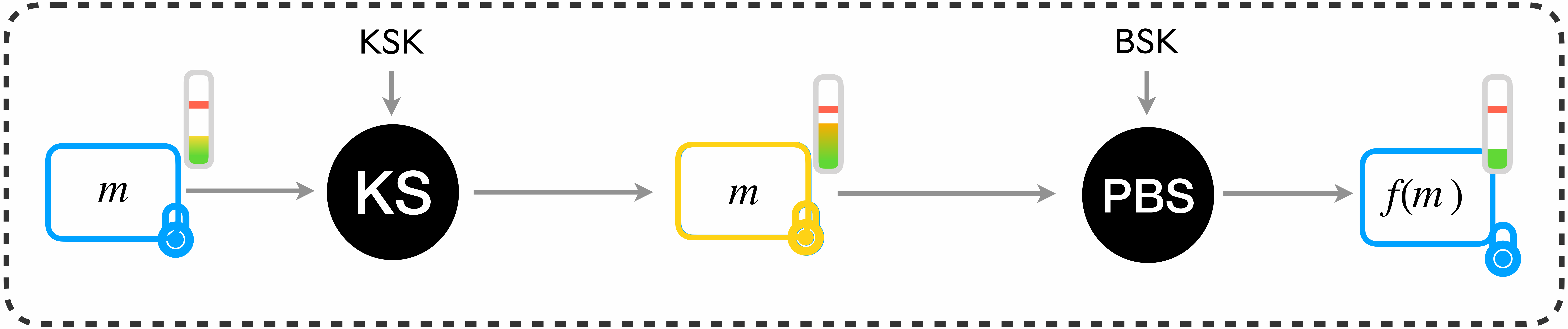
TFHE Bootstrapping Graph



Improving one operation leads to improve the whole graph

Improving operations :
– better complexity
– reduced noise growth

TFHE Bootstrapping Graph

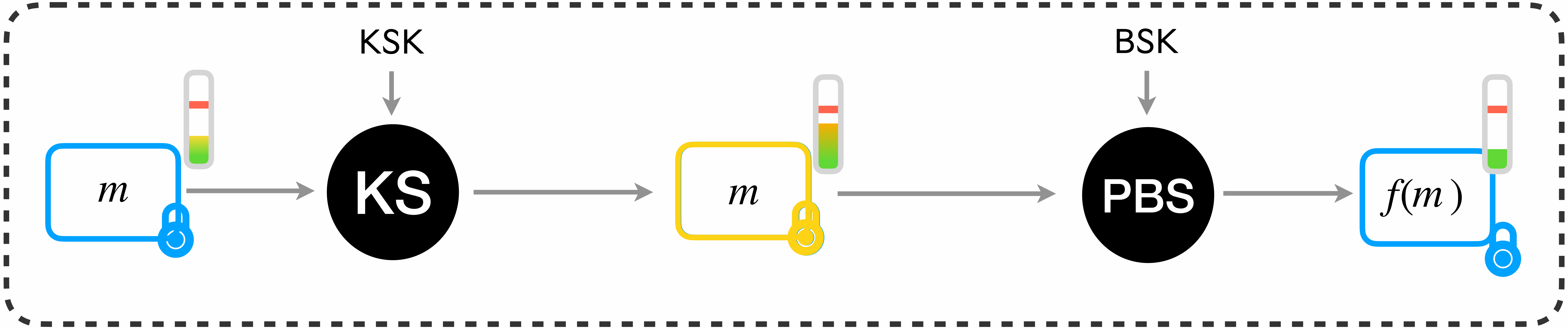


Improving one operation leads to improve the whole graph

Improving operations :
– better complexity
– reduced noise growth

A lot of improvements have been done on the PBS

TFHE Bootstrapping Graph



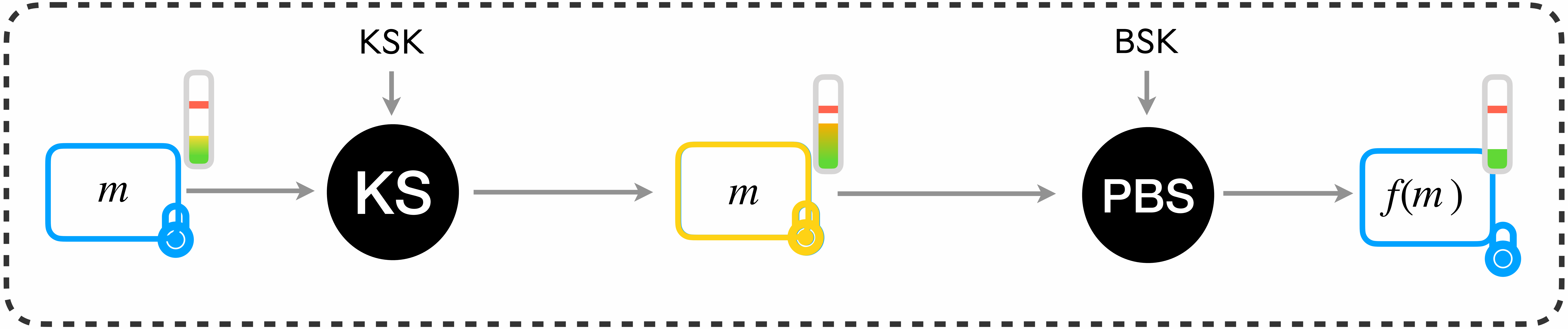
Improving one operation leads to improve the whole graph

Improving operations :
– better complexity
– reduced noise growth

A lot of improvements have been done on the PBS

We mainly focus of the **Keyswitch**

TFHE Bootstrapping Graph



Improving one operation leads to improve the whole graph

Improving operations :
– better complexity
– reduced noise growth

Can we explore **new assumptions** to improve the **bootstrapping** graph?

Summary

Can we explore **new assumptions** to improve the **bootstrapping** graph?

Our Contributions

Secret Keys with Shared Randomness

Partial Secret Keys

Summary

Can we explore **new assumptions** to improve the **bootstrapping** graph?

Our Contributions

Secret Keys with Shared Randomness

Partial Secret Keys

Up to **2.4 faster bootstrapping**

Size of evaluation keys **reduced** by a factor **2.7**

TFHE Background

Learning With Errors (LWE) Ciphertexts

Encrypt:

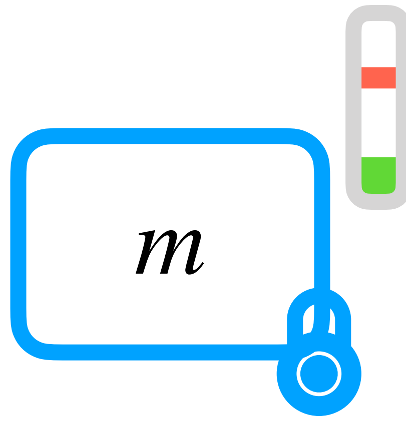
$sk = \vec{s}$

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

Learning With Errors (LWE) Ciphertexts

Encrypt:

$$sk = \overbrace{\boxed{\vec{s}}}^{s_i \hookleftarrow \mathbb{U}(\{0,1\})}$$



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

Learning With Errors (LWE) Ciphertexts

Encrypt:

$$sk = \overbrace{\vec{s}}^{s_i \hookleftarrow \mathbb{U}(\{0,1\})}$$

$$\boxed{m} = \vec{a}, b$$

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

Learning With Errors (LWE) Ciphertexts

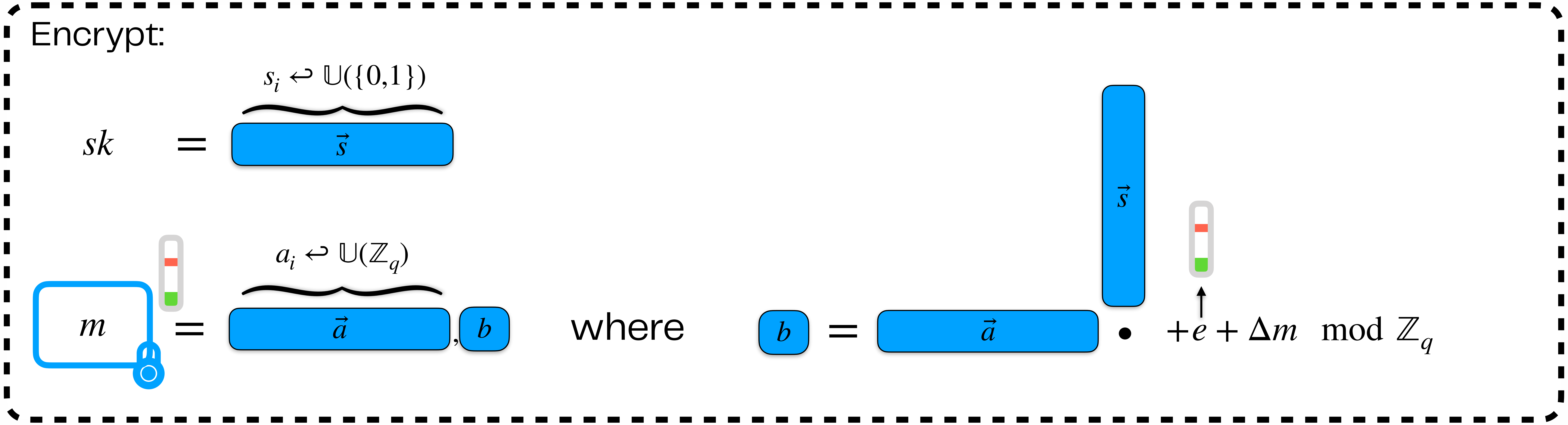
Encrypt:

$$sk = \overbrace{\vec{s}}^{s_i \hookleftarrow \mathbb{U}(\{0,1\})}$$

$$\boxed{m} = \overbrace{\vec{a}}^{a_i \hookleftarrow \mathbb{U}(\mathbb{Z}_q)}, b$$

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

Learning With Errors (LWE) Ciphertexts



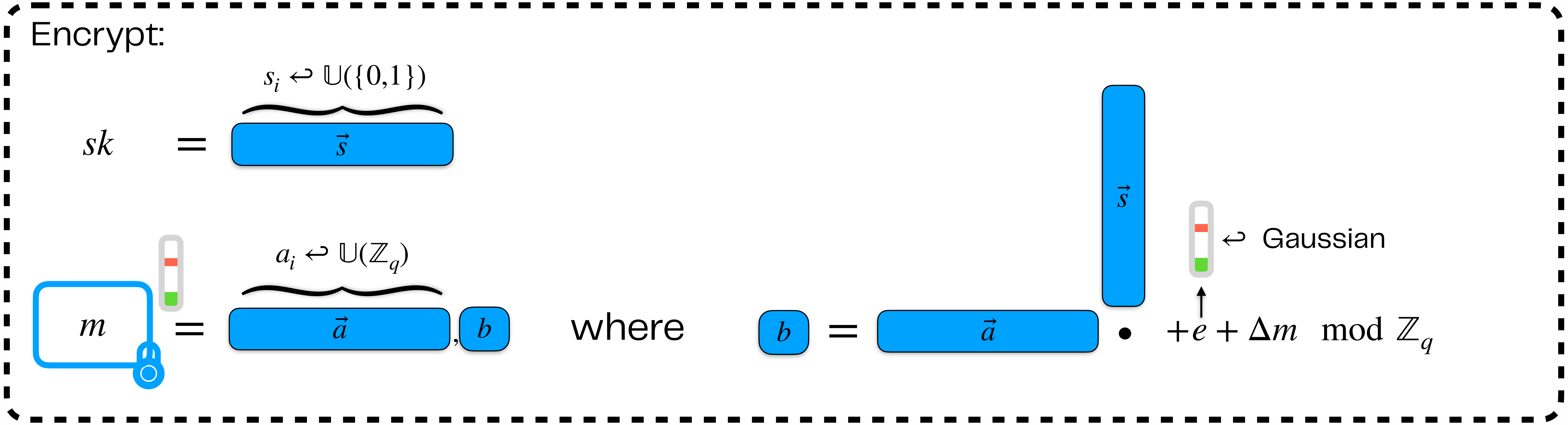
Encrypt:

$$sk = \overbrace{\vec{s}}^{s_i \hookleftarrow \mathbb{U}(\{0,1\})}$$

$$m = \overbrace{\vec{a}}^{a_i \hookleftarrow \mathbb{U}(\mathbb{Z}_q)}, b \quad \text{where} \quad b = \vec{a} \bullet \vec{s} + \overset{\uparrow}{e} + \Delta m \mod \mathbb{Z}_q$$

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

Learning With Errors (LWE) Ciphertexts



Encrypt:

$$sk = \overbrace{\vec{s}}^{s_i \hookleftarrow \mathbb{U}(\{0,1\})}$$

$$\boxed{m} = \overbrace{\vec{a}}^{a_i \hookleftarrow \mathbb{U}(\mathbb{Z}_q)}, b$$

where

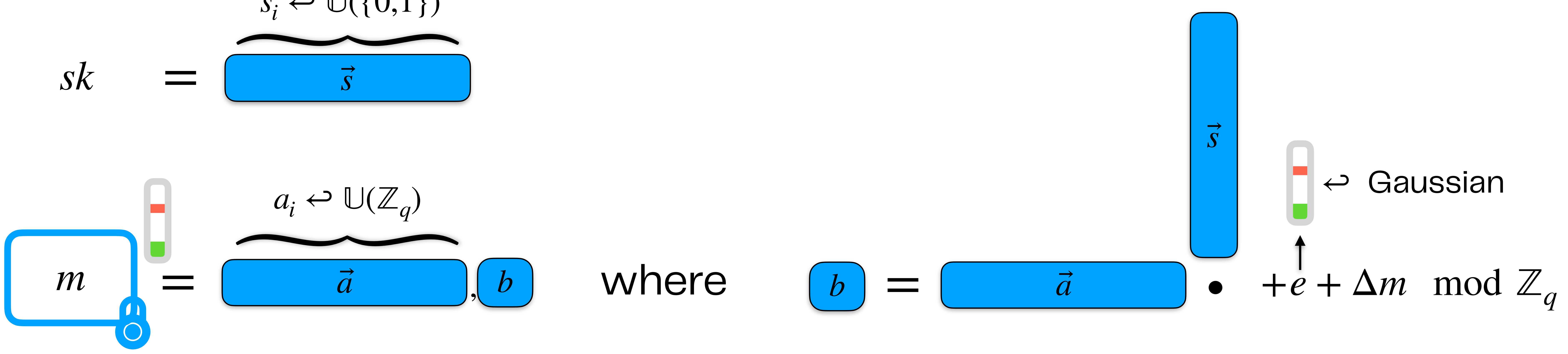
$$b = \vec{a} \bullet \vec{s} + \overset{\uparrow}{e} + \Delta m \mod \mathbb{Z}_q$$

$\hookleftarrow$ Gaussian

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

Learning With Errors (LWE) Ciphertexts

Encrypt:

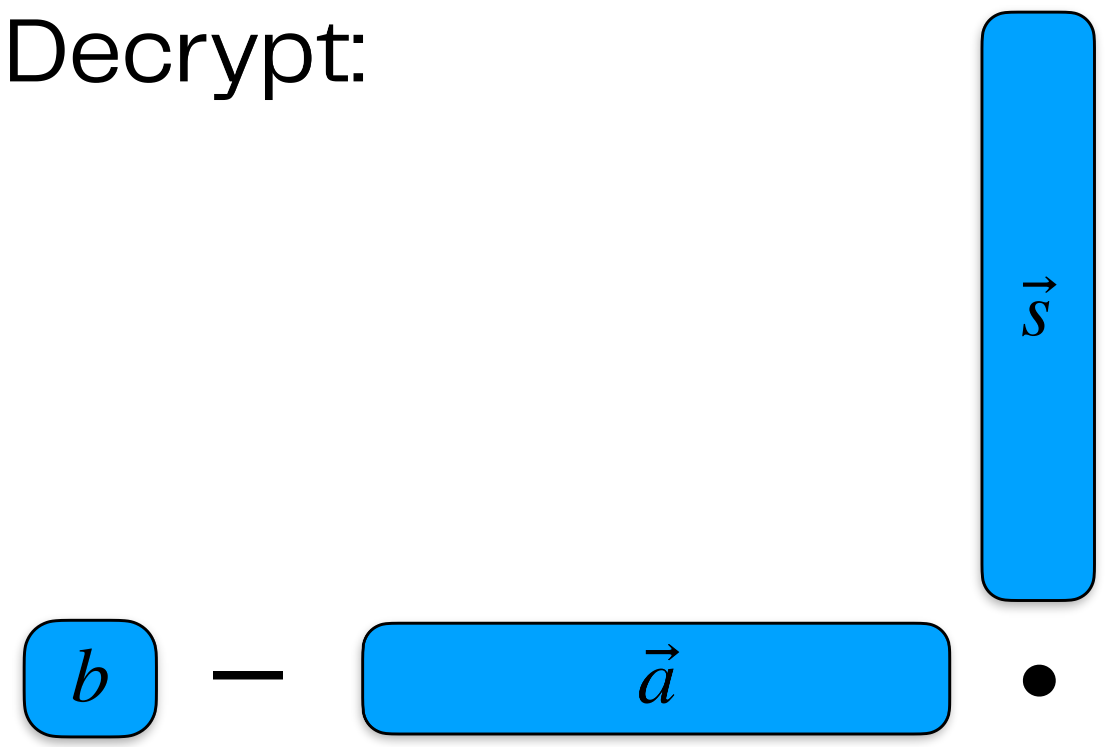


$$sk = \overbrace{\vec{s}}^{s_i \hookleftarrow \mathbb{U}(\{0,1\})}$$

$$\boxed{m} = \overbrace{\vec{a}}^{a_i \hookleftarrow \mathbb{U}(\mathbb{Z}_q)}, b \quad \text{where} \quad b = \vec{a} \bullet \vec{s} + e + \Delta m \mod \mathbb{Z}_q$$

$e \hookleftarrow$ Gaussian

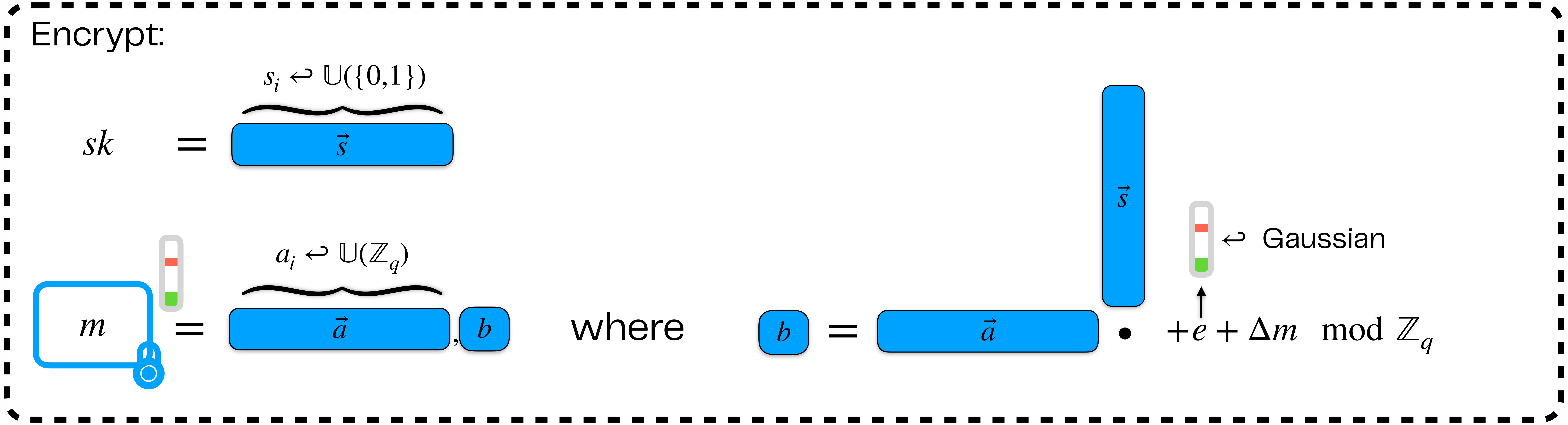
Decrypt:



$$b - \vec{a} \bullet \vec{s}$$

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

Learning With Errors (LWE) Ciphertexts



Encrypt:

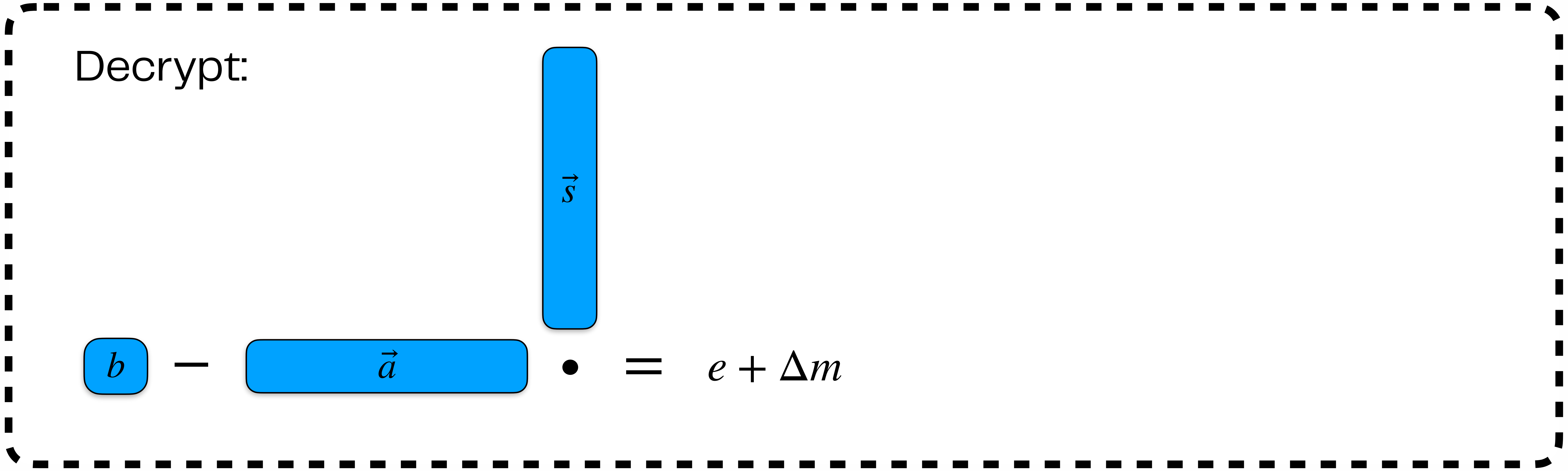
$$sk = \overbrace{\vec{s}}^{s_i \hookleftarrow \mathbb{U}(\{0,1\})}$$

$$m = \overbrace{\vec{a}}^{a_i \hookleftarrow \mathbb{U}(\mathbb{Z}_q)}, b$$

where

$$b = \vec{a} \bullet \vec{s} + e + \Delta m \mod \mathbb{Z}_q$$

$\hookleftarrow$ Gaussian

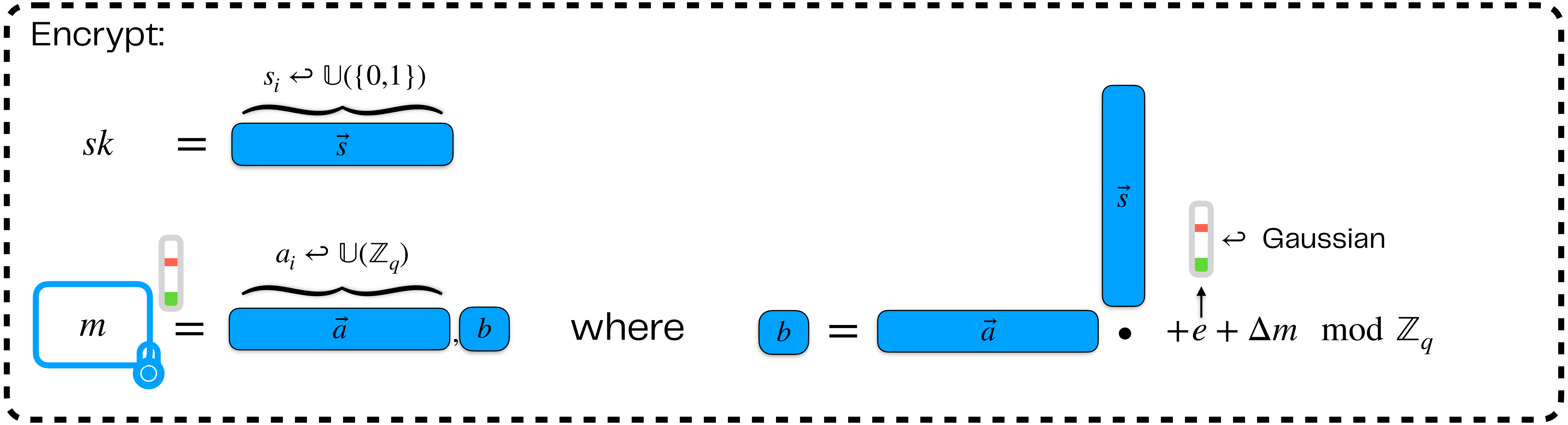


Decrypt:

$$b - \vec{a} \bullet \vec{s} = e + \Delta m$$

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

Learning With Errors (LWE) Ciphertexts



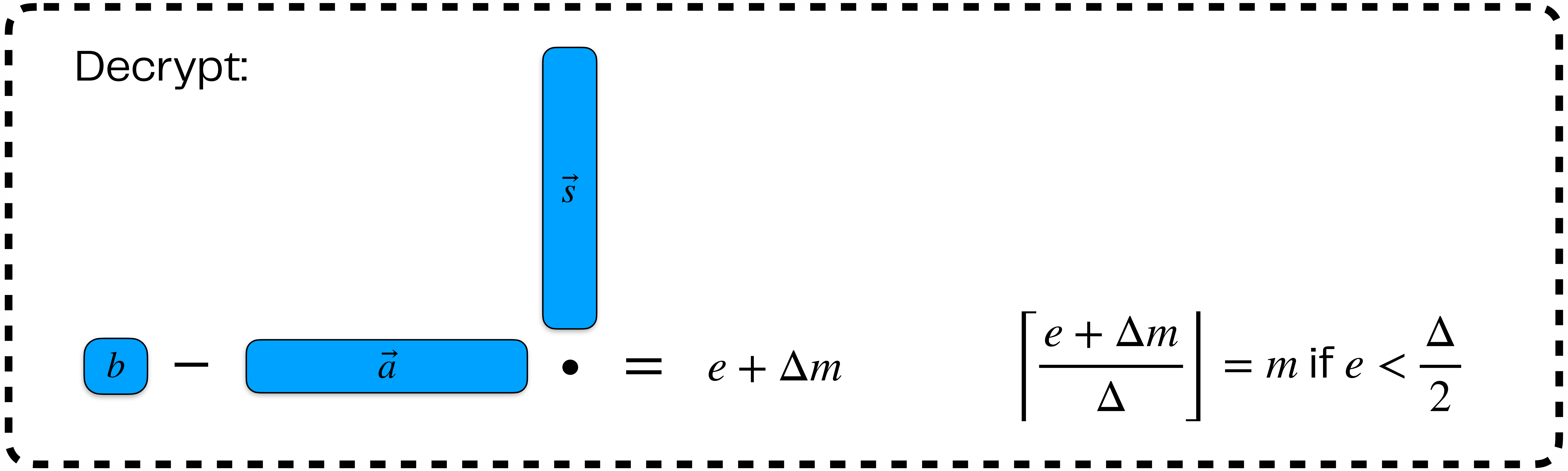
Encrypt:

$$sk = \overbrace{\vec{s}}^{s_i \hookleftarrow \mathbb{U}(\{0,1\})}$$

$$m = \overbrace{\vec{a}}^{a_i \hookleftarrow \mathbb{U}(\mathbb{Z}_q)}, b$$

where $b = \vec{a} \bullet \vec{s} + e + \Delta m \mod \mathbb{Z}_q$

$\hookleftarrow$ Gaussian



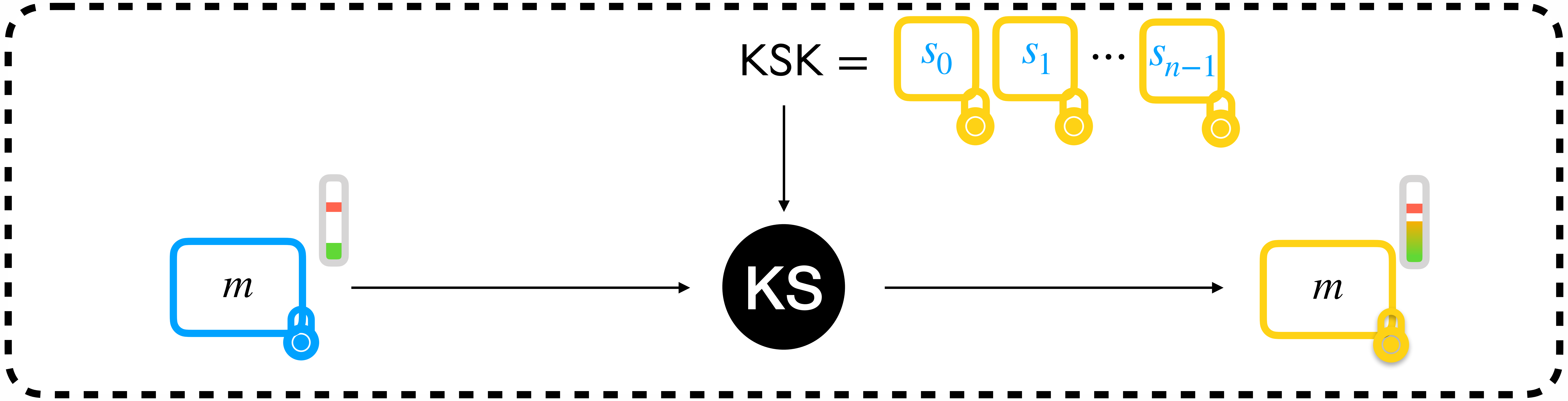
Decrypt:

$$b - \vec{a} \bullet \vec{s} = e + \Delta m$$

$$\left\lceil \frac{e + \Delta m}{\Delta} \right\rfloor = m \text{ if } e < \frac{\Delta}{2}$$

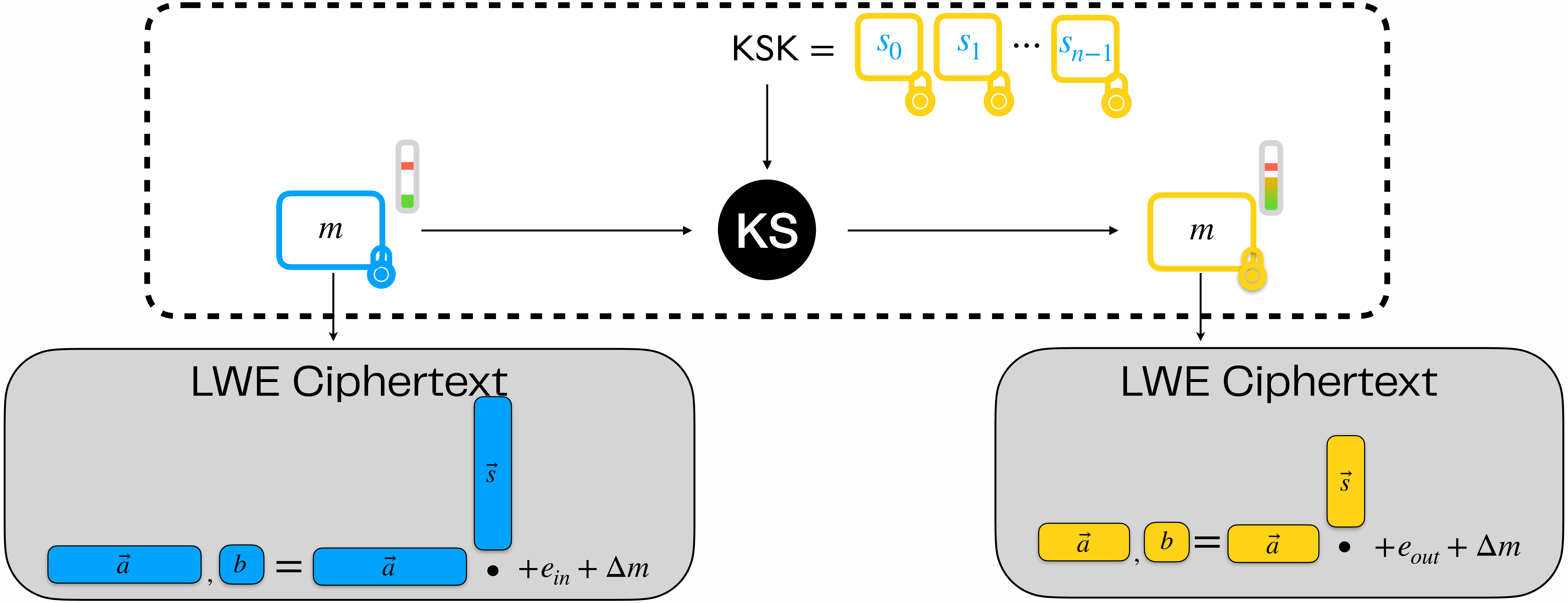
[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In STOC 2005. ACM, 2005.

Keyswitch



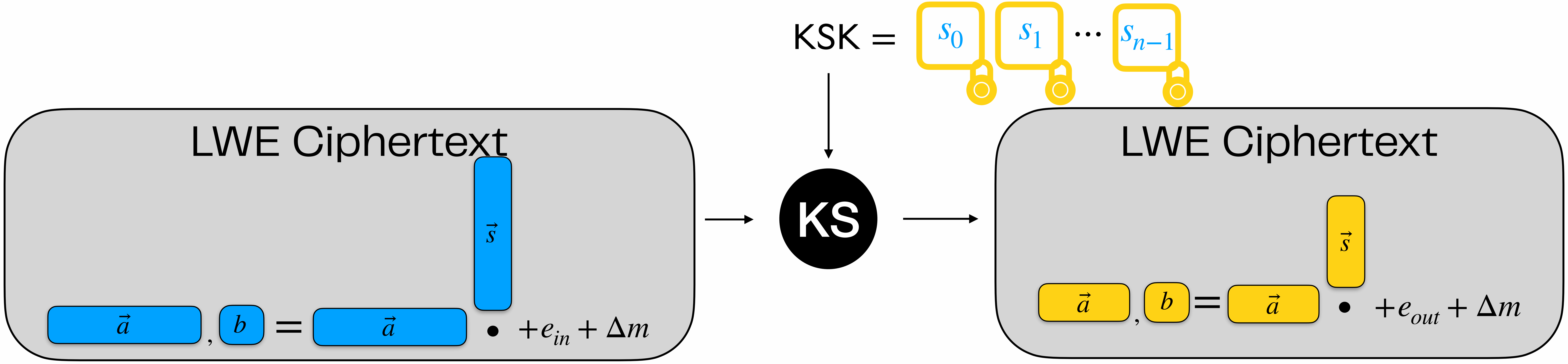
Switch from a secret key to another secret key

Keyswitch



Switch from a secret key to another secret key

Keyswitch



$\text{KSK} = s_0 \; s_1 \cdots s_{n-1}$

LWE Ciphertext

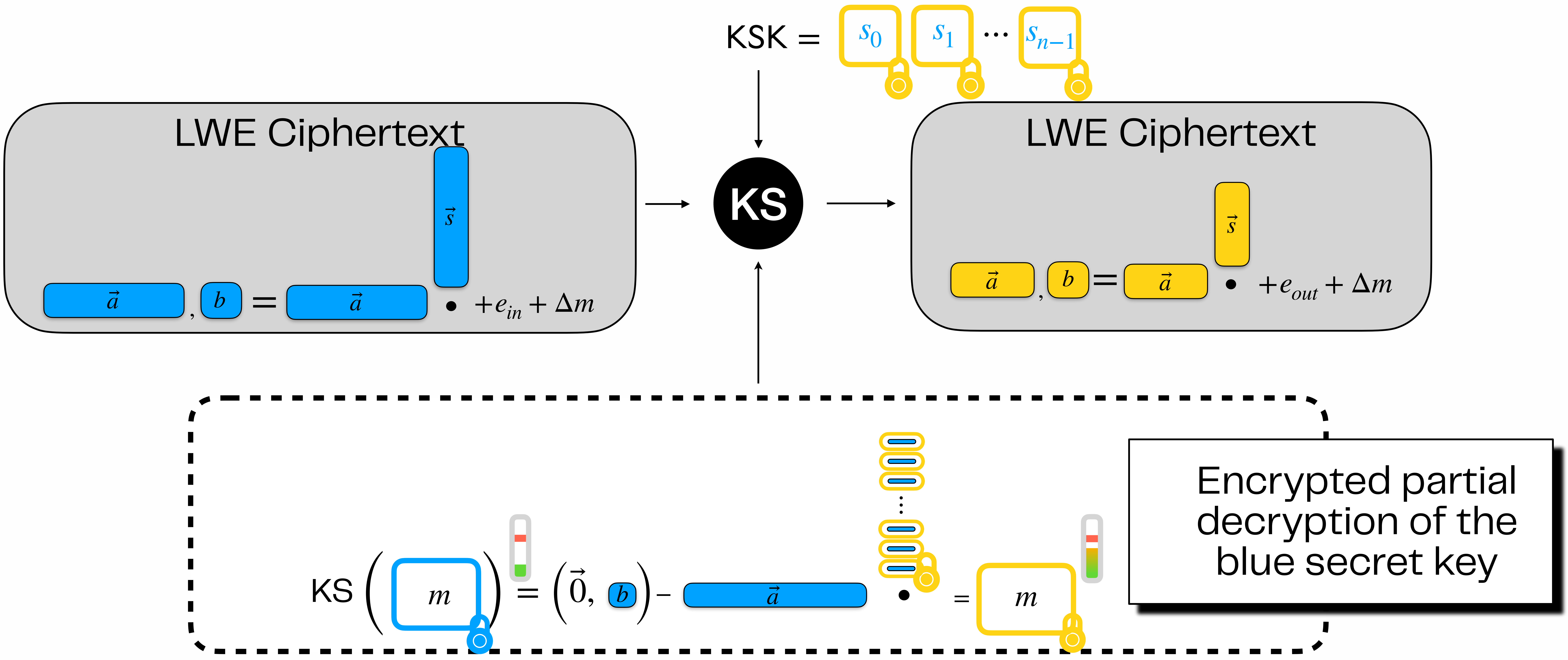
$\vec{a}, b = \vec{a} \bullet \vec{s} + e_{in} + \Delta m$

KS

LWE Ciphertext

$\vec{a}, b = \vec{a} \bullet \vec{s} + e_{out} + \Delta m$

Keyswitch



$\text{KSK} = \boxed{s_0}\ \boxed{s_1} \cdots \boxed{s_{n-1}}$

LWE Ciphertext

$\vec{a}\,,\ b = \vec{a} \bullet \vec{s} + e_{in} + \Delta m$

KS

LWE Ciphertext

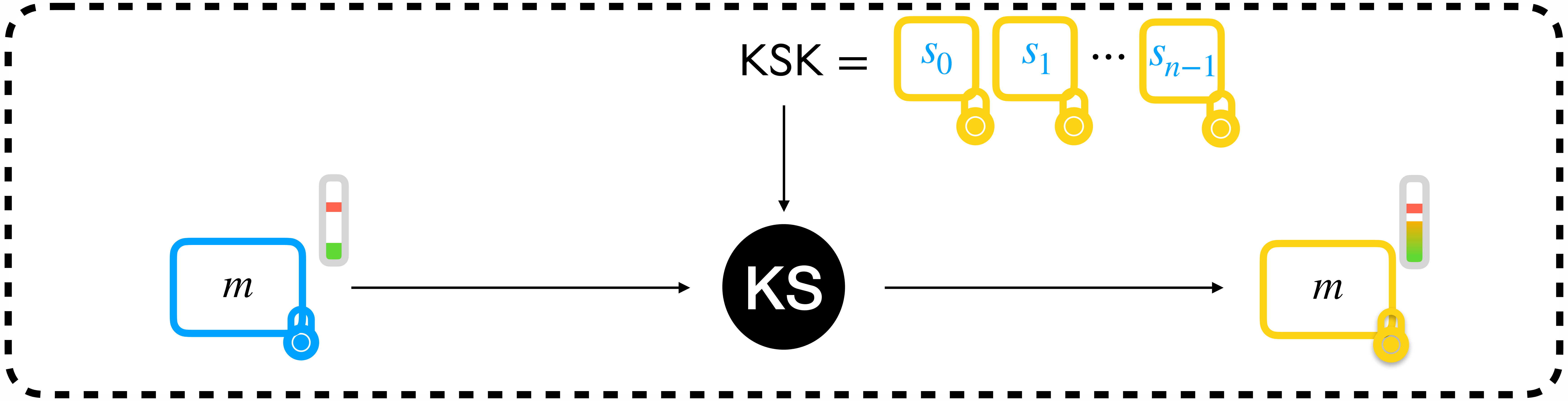
$\vec{a}\,,\ b = \vec{a} \bullet \vec{s} + e_{out} + \Delta m$

$\text{KS}\left(\boxed{m}\right) = \left(\vec{0},\ b\right) - \vec{a} \bullet \begin{matrix} \vdots \end{matrix} = \boxed{m}$

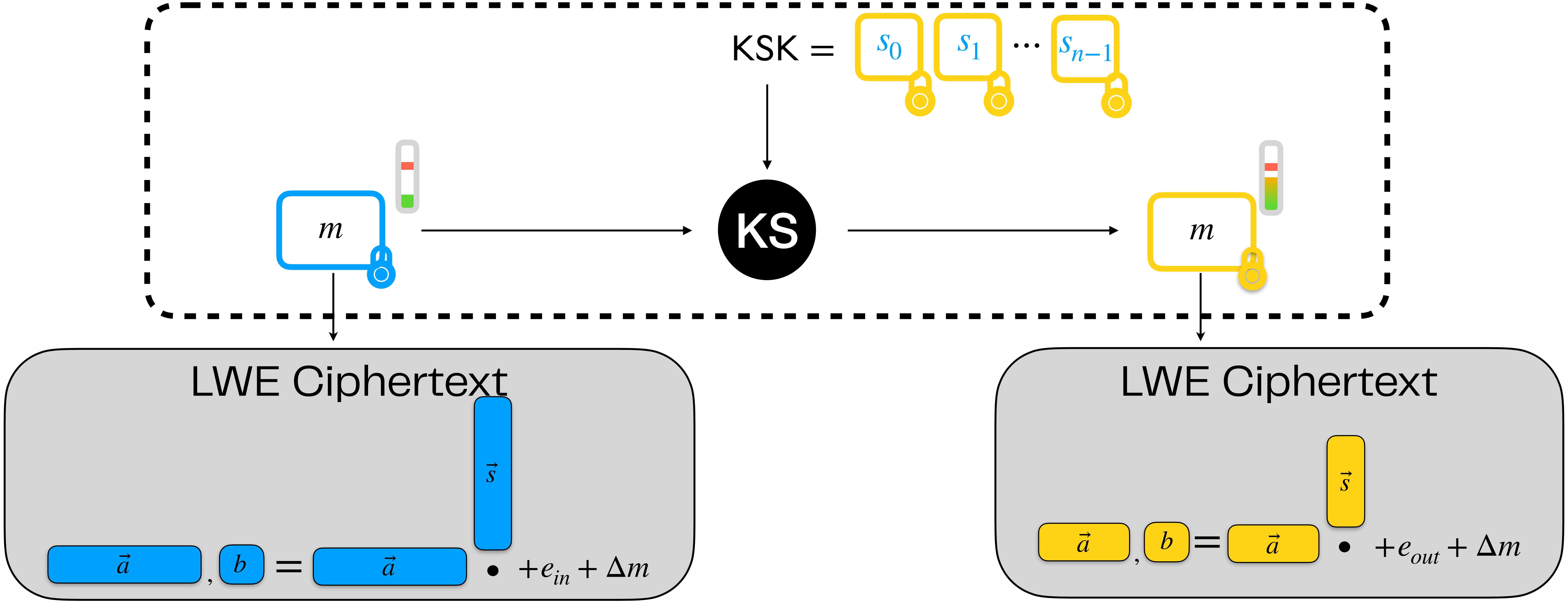
Encrypted partial decryption of the blue secret key

Secret Keys with Shared Randomness

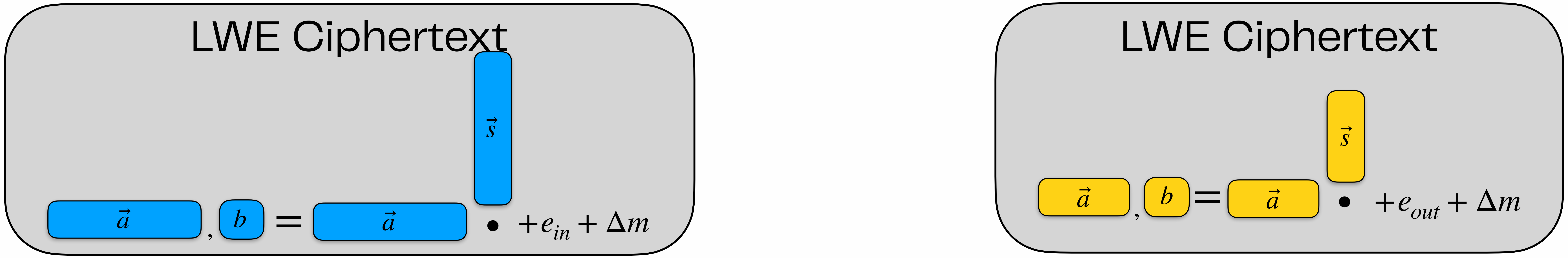
Shared Randomness



Shared Randomness



Shared Randomness



LWE Ciphertext

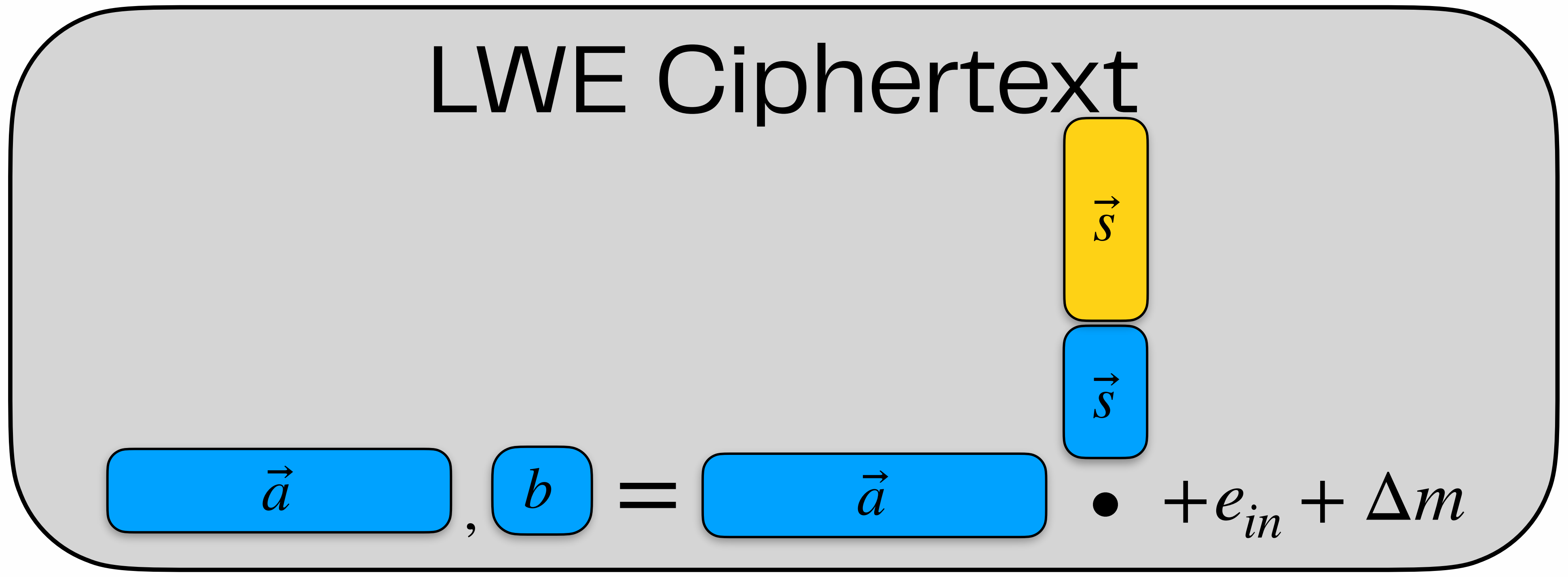
$\vec{a}, b = \vec{a} \bullet \vec{s} + e_{in} + \Delta m$

LWE Ciphertext

$\vec{a}, b = \vec{a} \bullet \vec{s} + e_{out} + \Delta m$

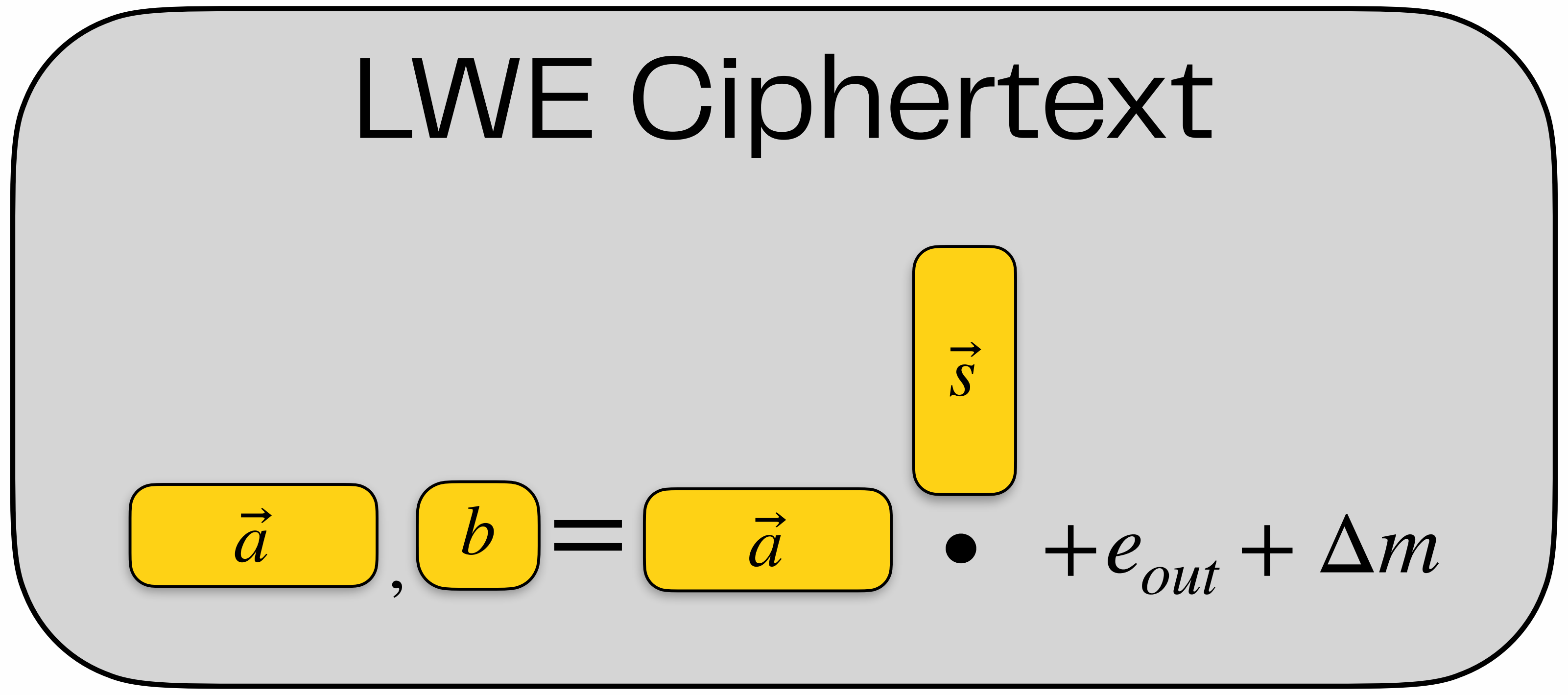
Reuse the randomness of the smaller secret key

Shared Randomness



LWE Ciphertext

$\vec{a}, b = \vec{a} \bullet \begin{pmatrix} \vec{s} \\ \vec{s} \end{pmatrix} + e_{in} + \Delta m$

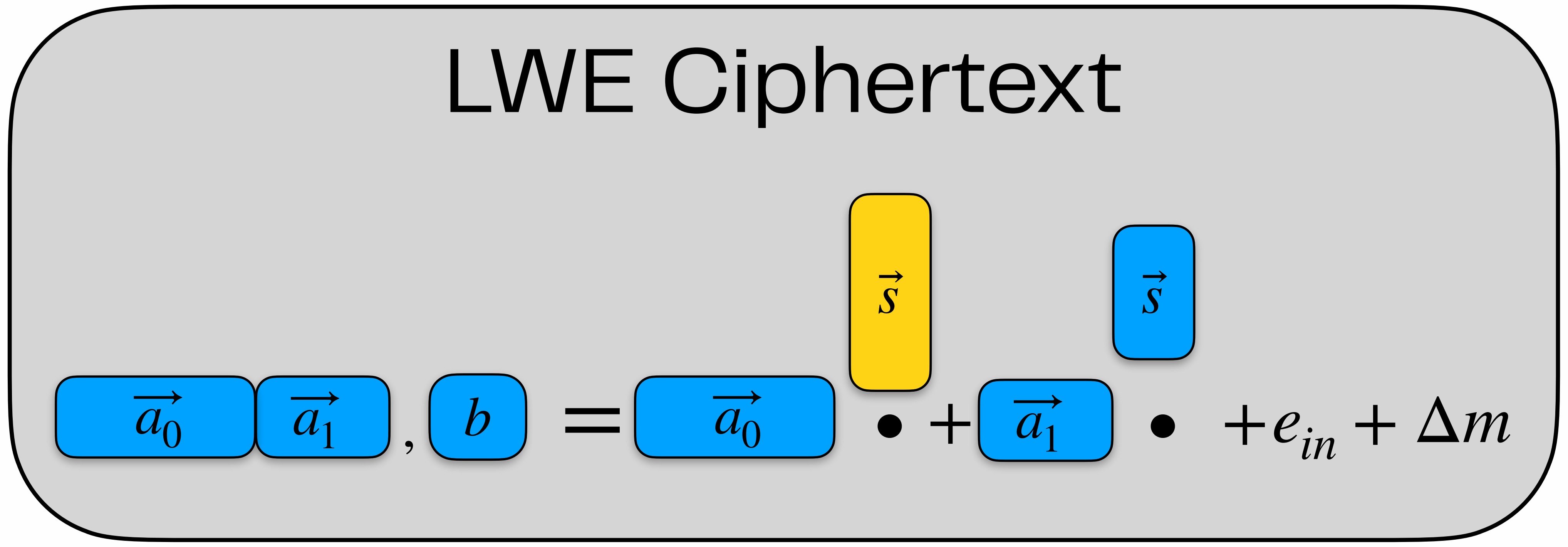


LWE Ciphertext

$\vec{a}, b = \vec{a} \bullet \vec{s} + e_{out} + \Delta m$

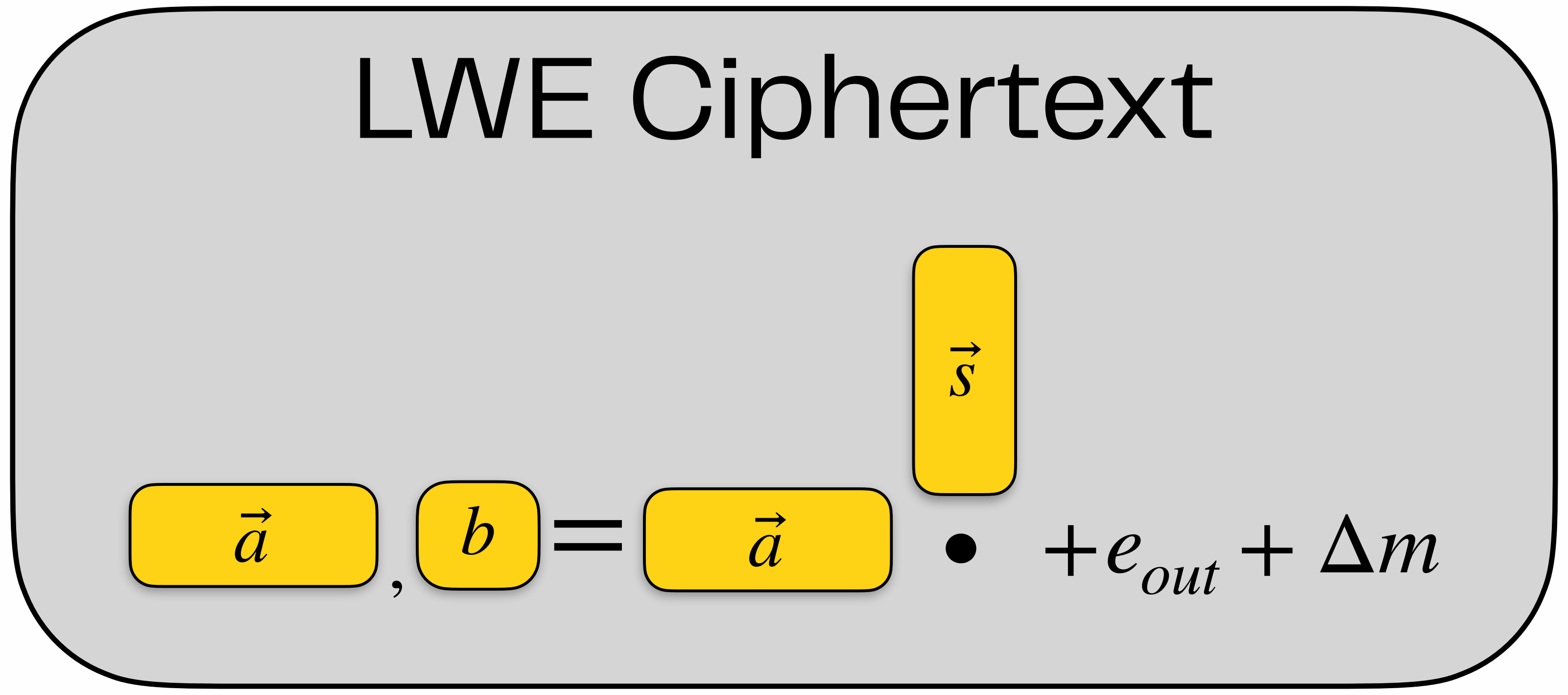
Reuse the randomness of the smaller secret key

Shared Randomness



LWE Ciphertext

$$\vec{a_0}\ \vec{a_1},\ b = \vec{a_0} \bullet \vec{s} + \vec{a_1} \bullet \vec{s} + e_{in} + \Delta m$$

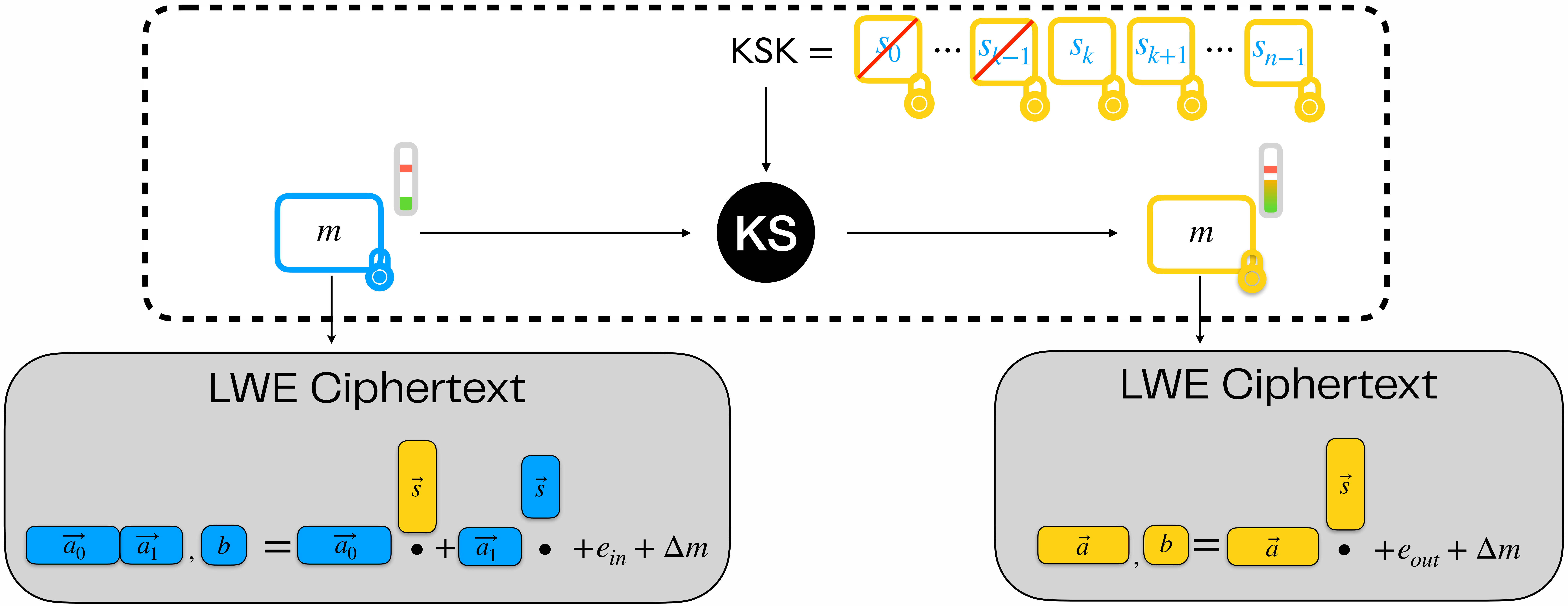


LWE Ciphertext

$$\vec{a},\ b = \vec{a} \bullet \vec{s} + e_{out} + \Delta m$$

Reuse the randomness of the smaller secret key

Shared Randomness



$\text{KSK} = \cancel{s_0} \cdots \cancel{s_{k-1}}\ s_k\ s_{k+1} \cdots s_{n-1}$

LWE Ciphertext

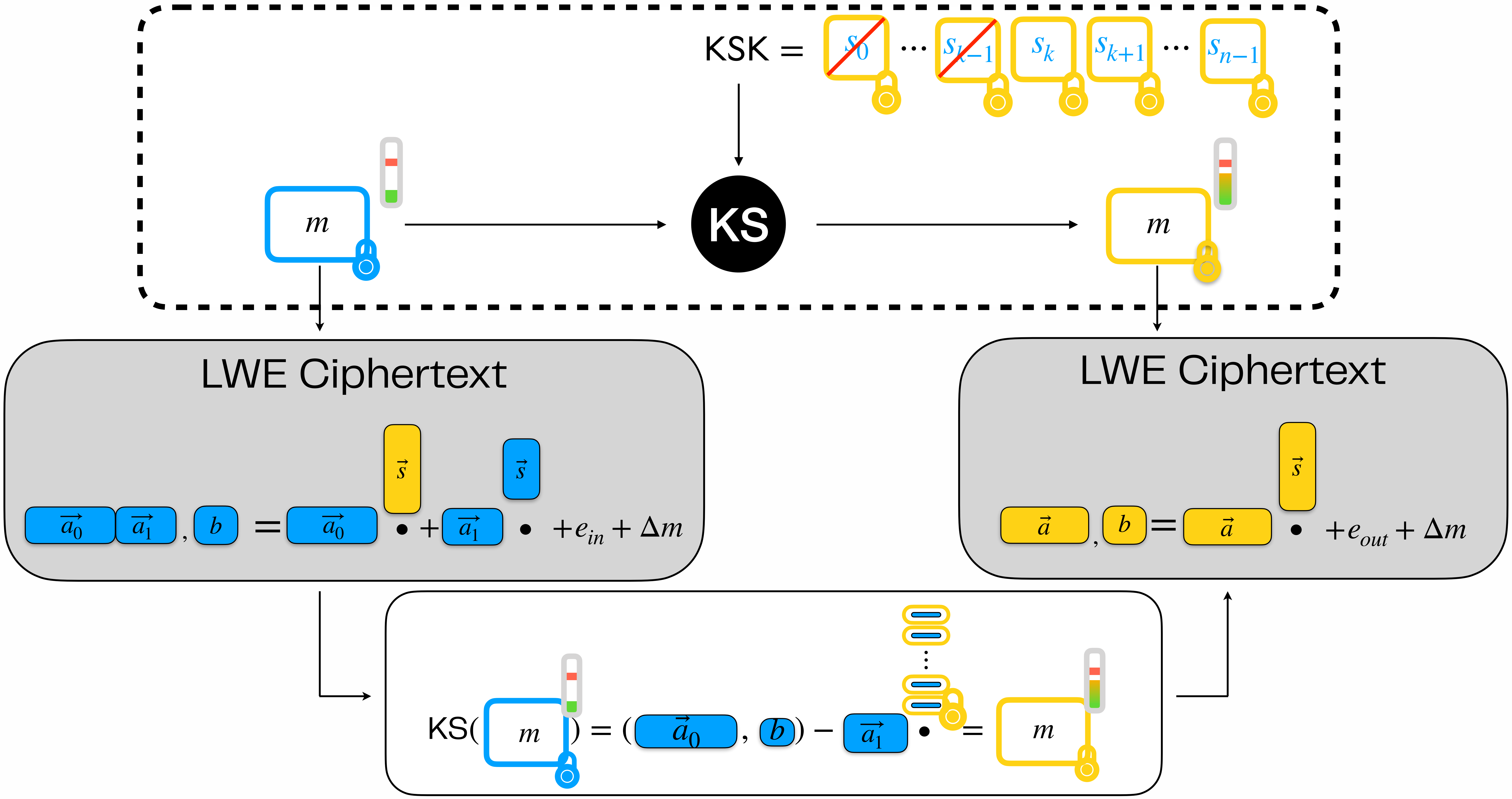
$\vec{a_0}\ \vec{a_1}, b = \vec{a_0} \bullet \vec{s} + \vec{a_1} \bullet \vec{s} + e_{in} + \Delta m$

LWE Ciphertext

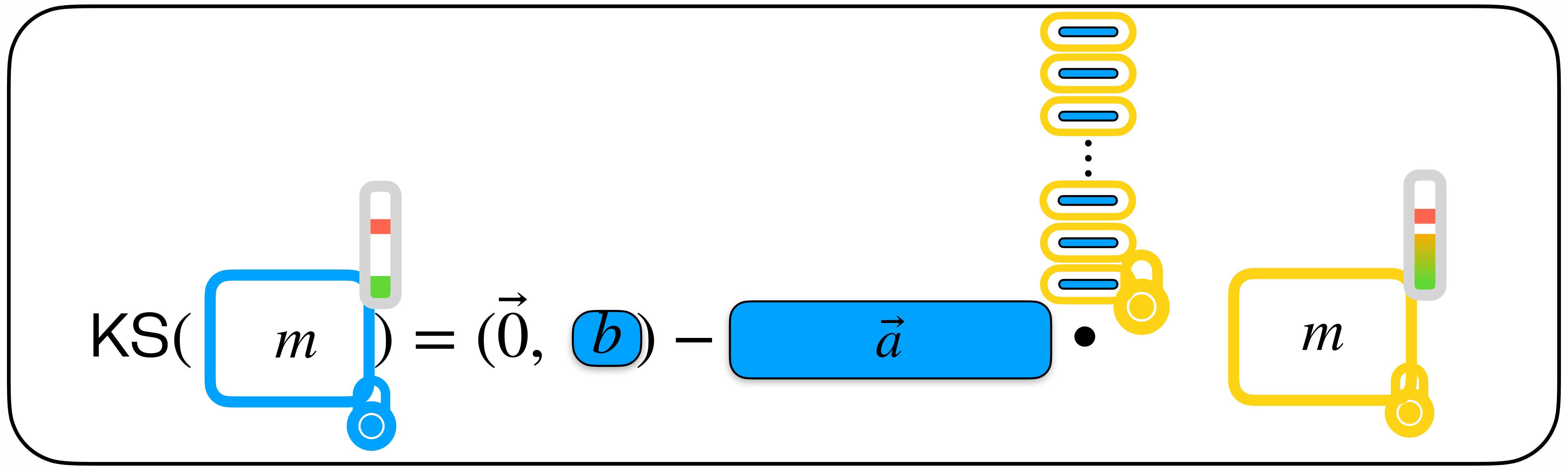
$\vec{a}, b = \vec{a} \bullet \vec{s} + e_{out} + \Delta m$

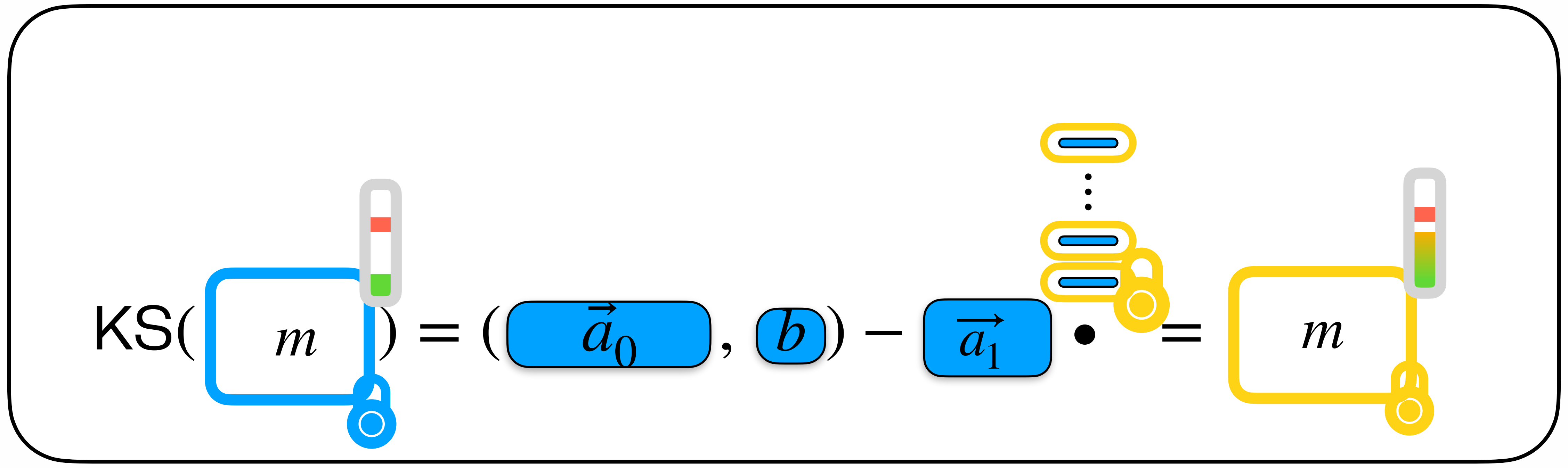
Reduce the size of the Keyswitching Key (KSK)

Shared Randomness



Advantages of Shared Randomness

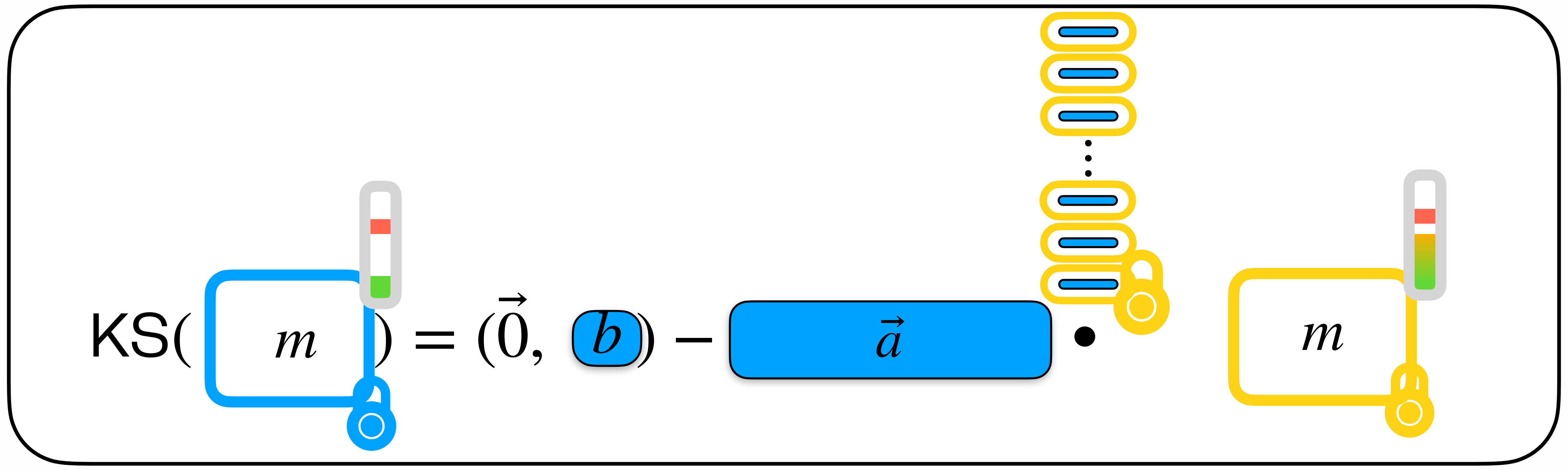


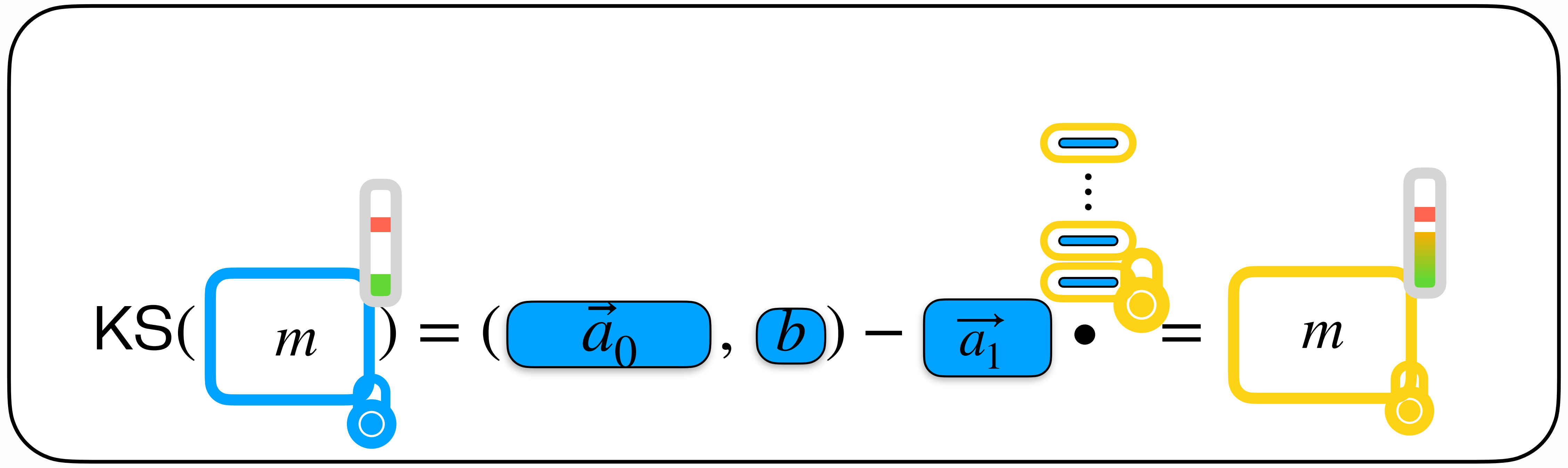


Keyswitch

KS with Shared Randomness

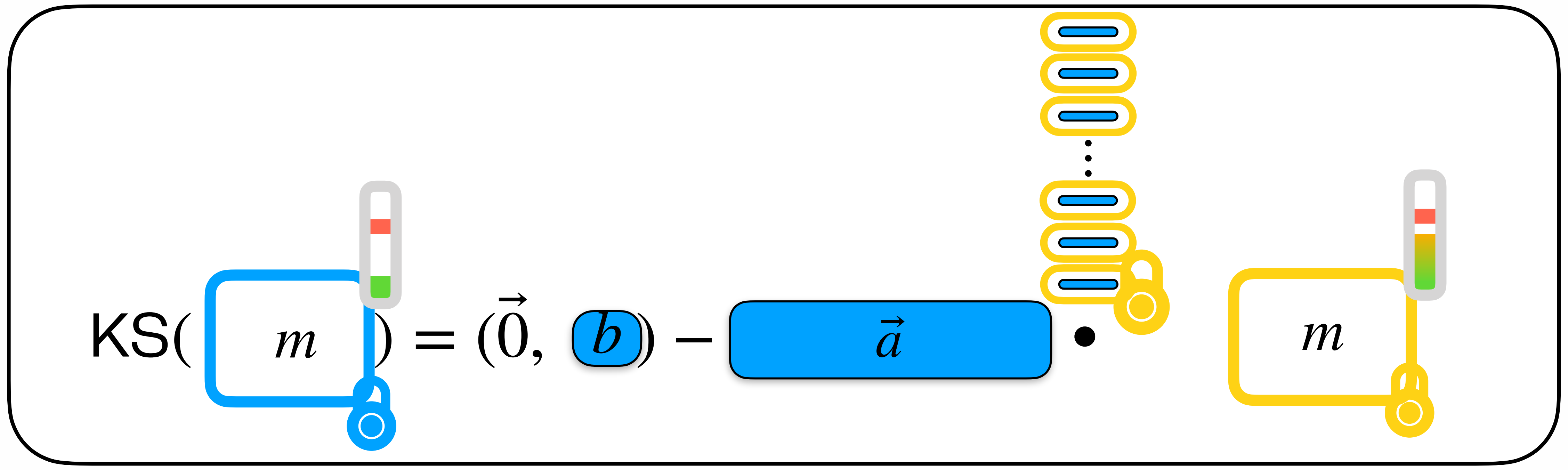
Advantages of Shared Randomness

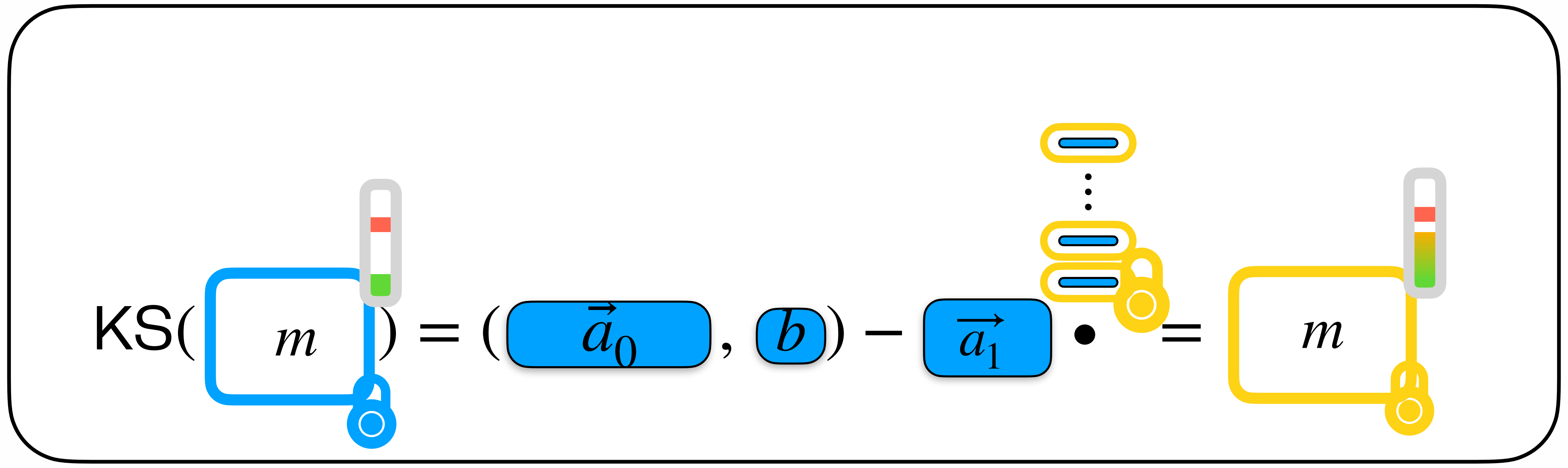




Keyswitch	**KS with Shared Randomness**
KSK composed of **n ciphertexts**	**KSK** composed of **n-k ciphertexts**

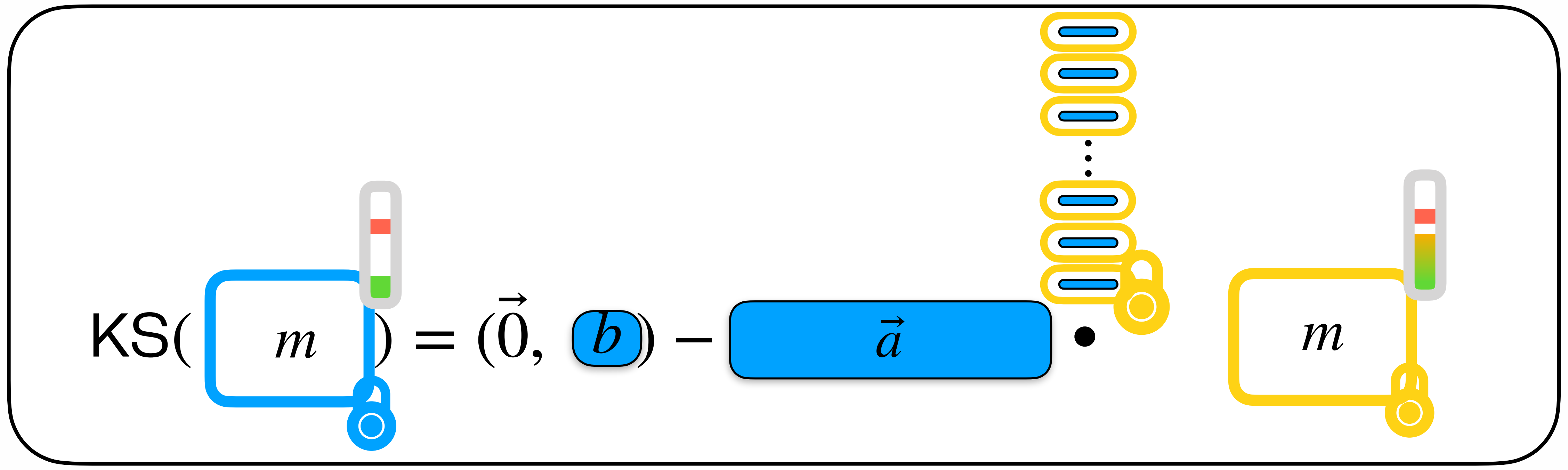
Advantages of Shared Randomness

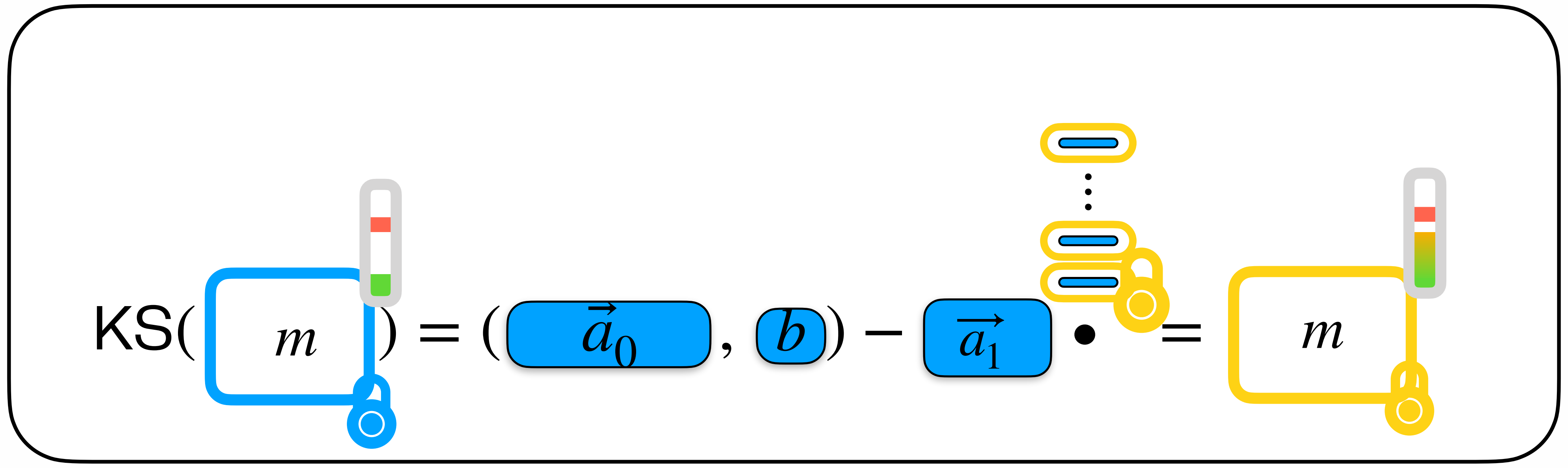




Keyswitch	**KS with Shared Randomness**
KSK composed of **n ciphertexts**	**KSK** composed of **n-k ciphertexts**
O(n) operations	**O(n-k) operations**

Advantages of Shared Randomness





Keyswitch

KSK composed of
n ciphertexts

O(n) operations

KS with Shared Randomness

KSK composed of
n-k ciphertexts

O(n-k) operations

Faster
with **less noise**

Stair Keyswitch

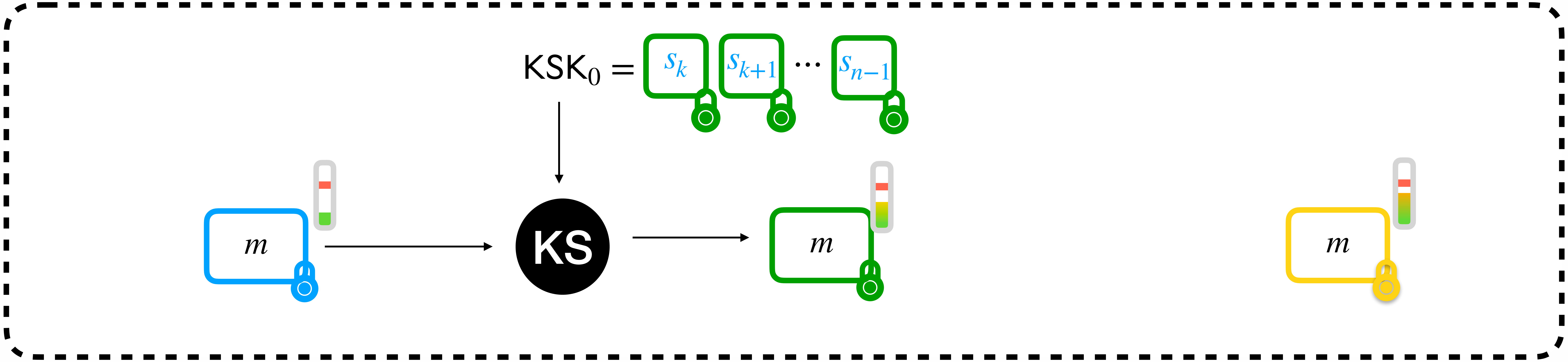
Perform the Keyswitch in several steps

Balance the cost and the noise of the Keyswitch

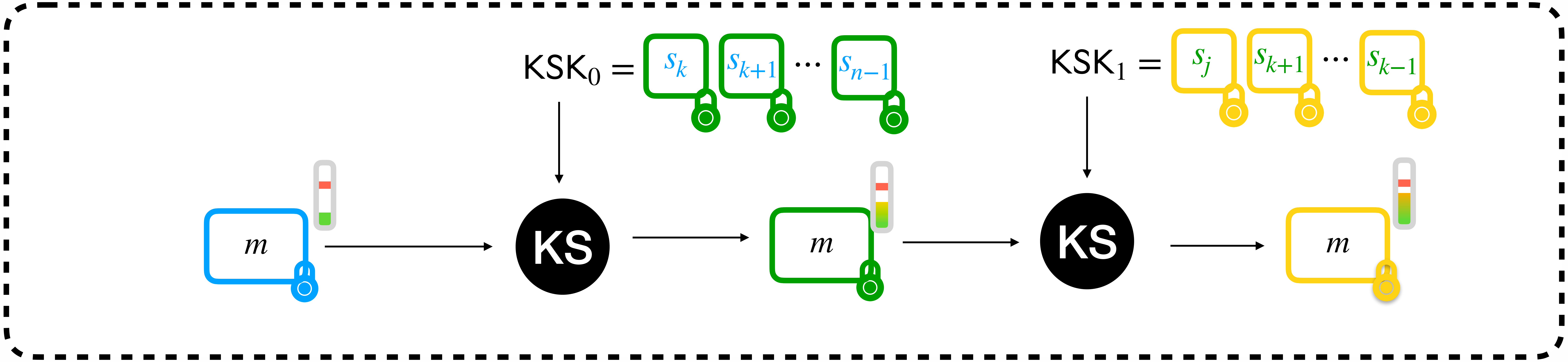
More parameter choices

Stair Keyswitch

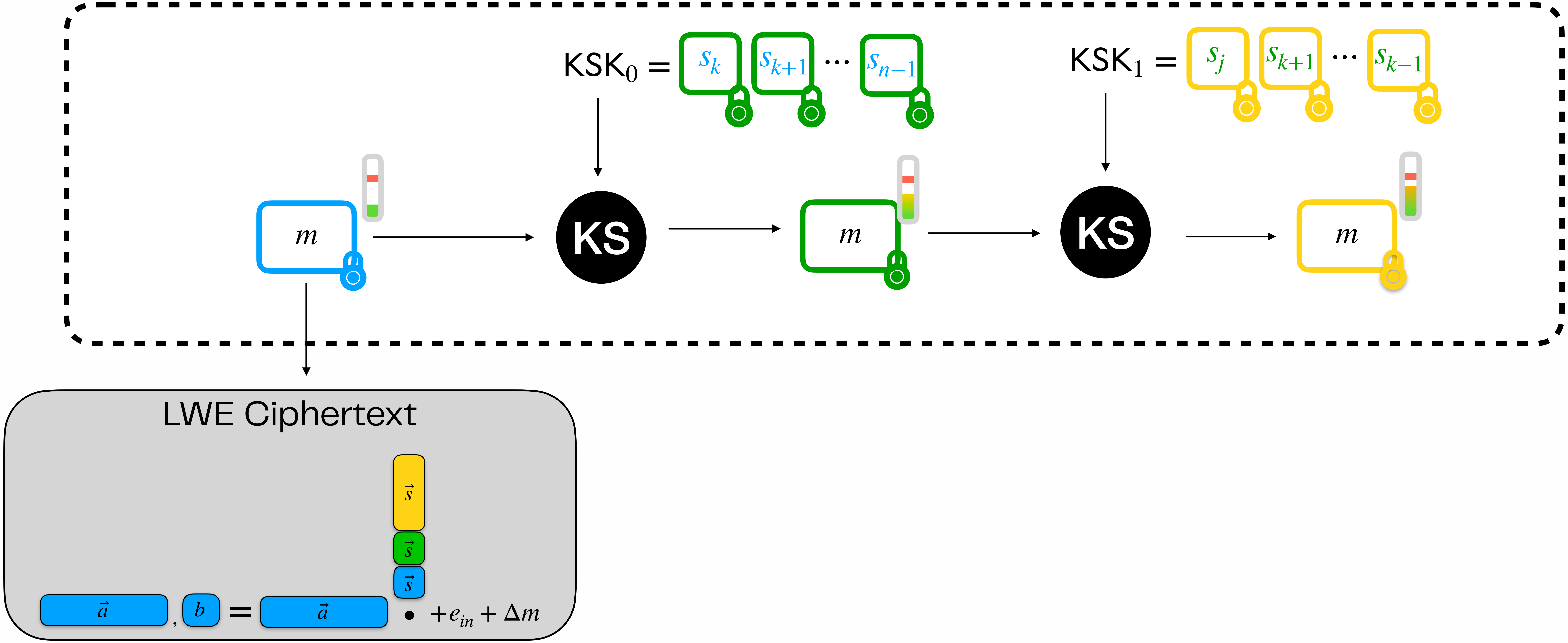
Stair Keyswitch



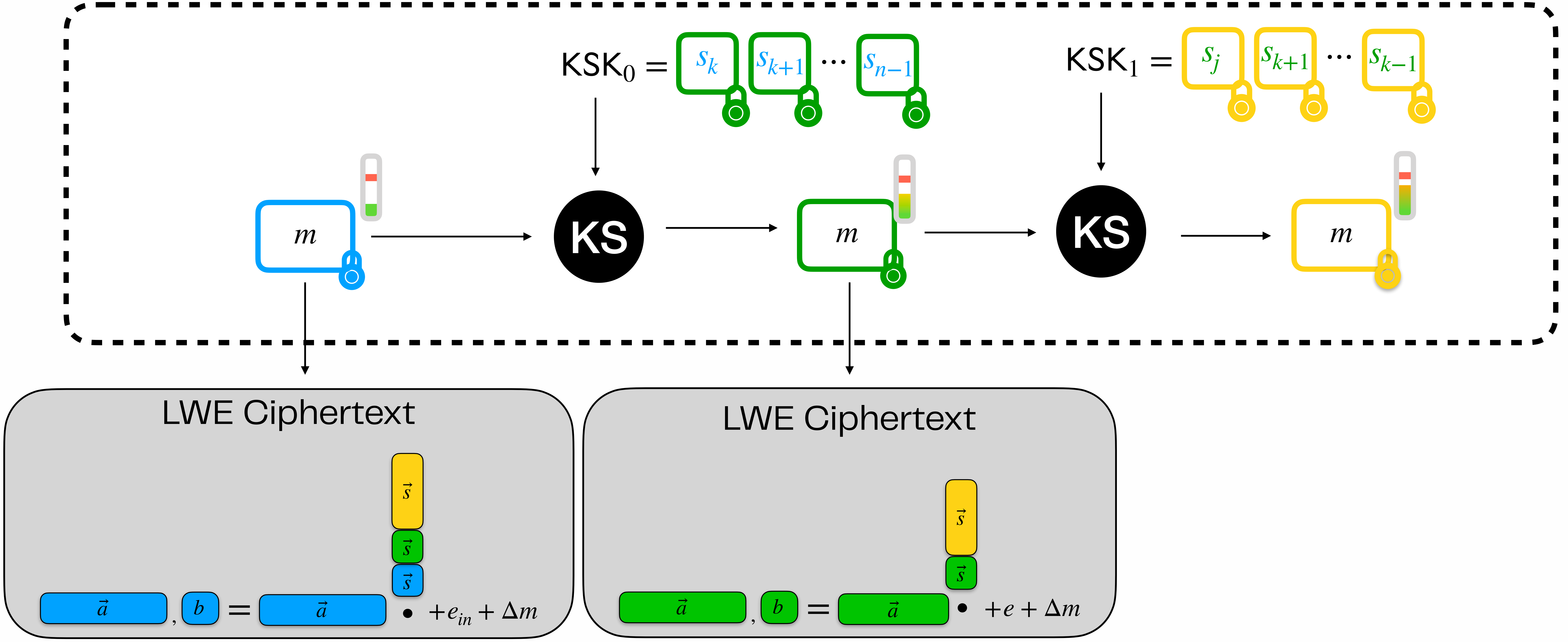
Stair Keyswitch



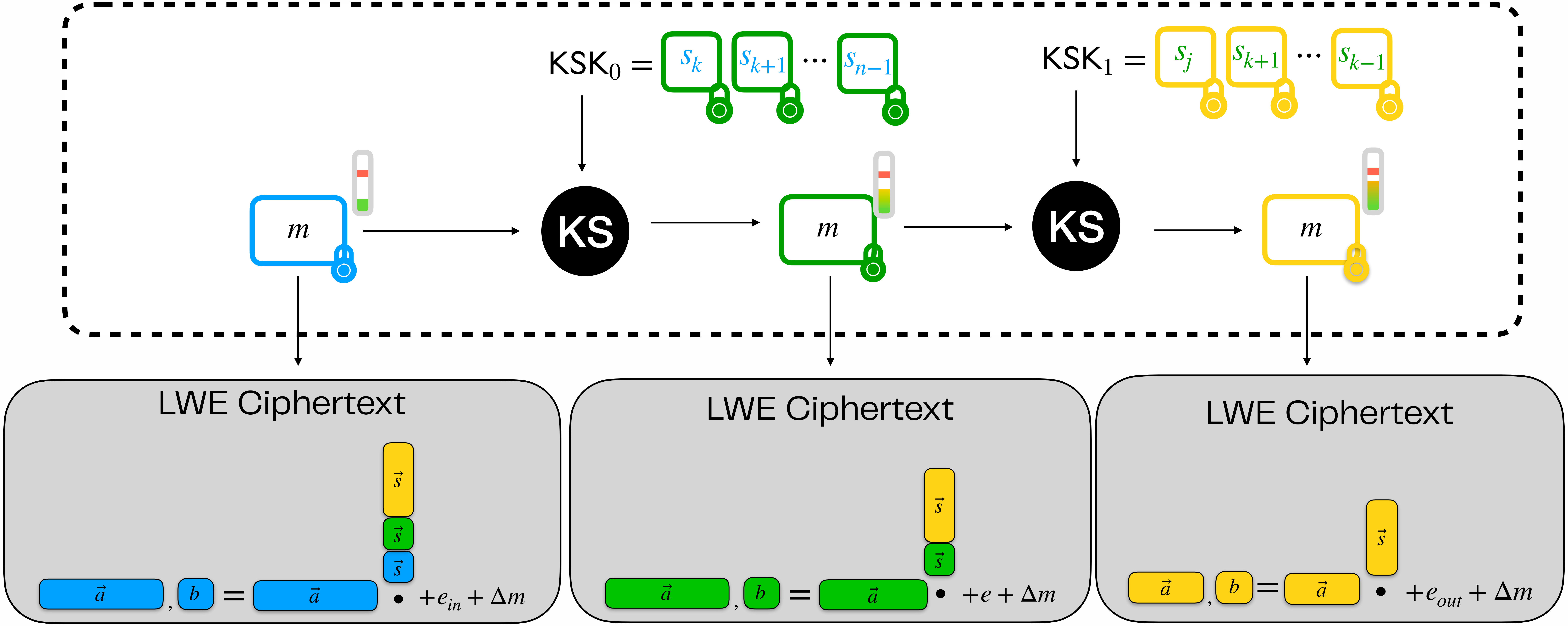
Stair Keyswitch



Stair Keyswitch



Stair Keyswitch



Benchmarks

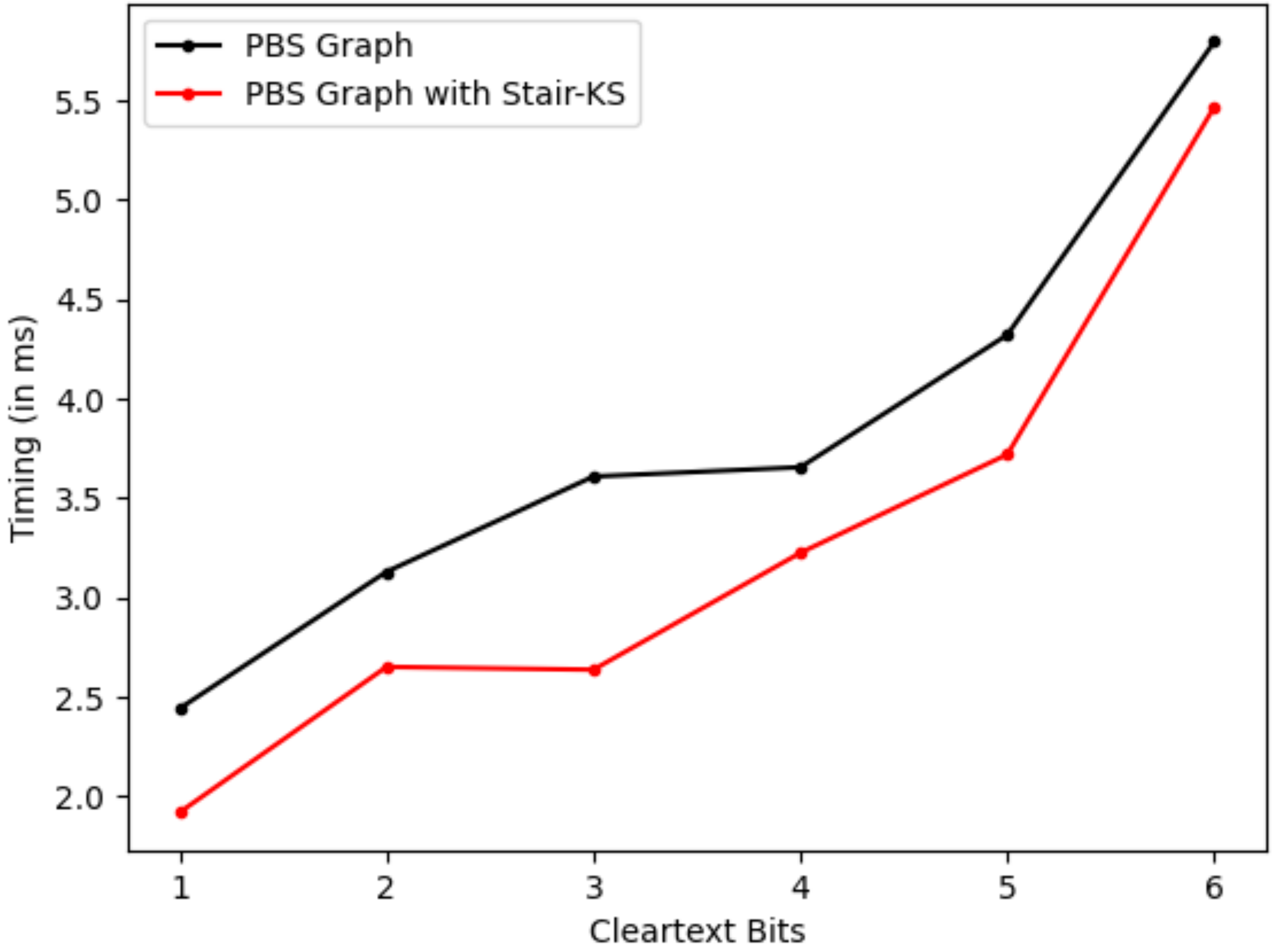
Bootstrapping Graph

with $P_{\text{fail}} = 2^{-14}$

using TFHE-rs

Speed-ups between **1.2** and **1.9**

Result based on **new assumption**



AWS M6i.metal Intel Xeon 8375C (Ice Lake) 128 vCPUs & 512GiB

Partial
Secret Keys

Improving Keyswitch

How to improve the Keyswitch?

Improving Keyswitch

How to improve the Keyswitch?

Use polynomials to compute the Keyswitch dot product → **Use the FFT** → **Faster computation**

Improving Keyswitch

How to improve the Keyswitch?

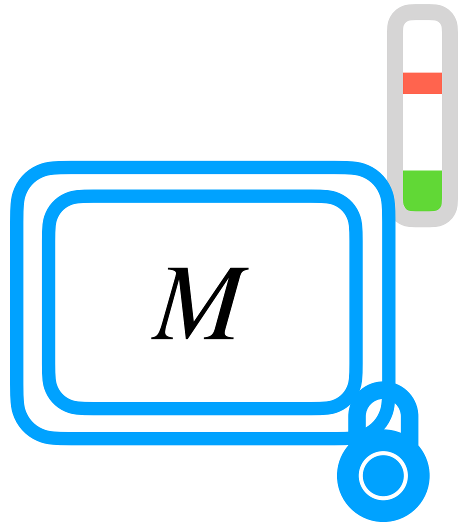
Use polynomials to compute the Keyswitch dot product → Use the FFT → Faster computation

Use the RLWE assumption

Ring LWE Ciphertexts

Encrypt:

$\mathscr{R}_q = \mathbb{Z}_q[X]/\langle X^N + 1\rangle$ with N a power of two



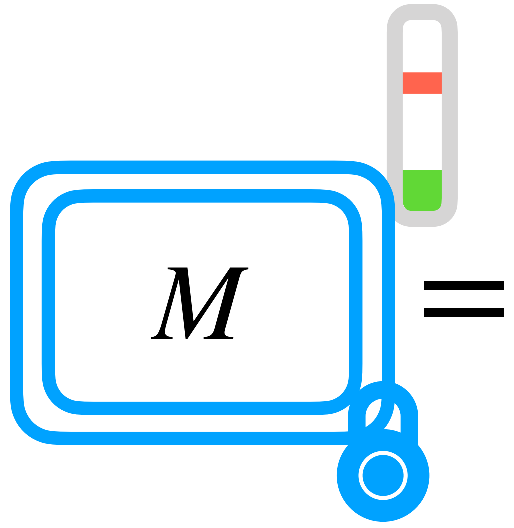
[SSTX09] Damien Stehlé, Ron Steinfeld, Keisuke Tanaka, and Keita Xagawa. Efficient public key encryption based on ideal lattices. In ASIACRYPT 2009. Springer,2009
[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In EUROCRYPT 2010. Springer, 2010.

Ring LWE Ciphertexts

Encrypt:

$\mathscr{R}_q = \mathbb{Z}_q[X]/\langle X^N + 1\rangle$ with N a power of two

$$sk = \boxed{S} \in \mathscr{R}_2$$



$$\boxed{M} = (\boxed{A}, \boxed{B}) \in \mathscr{R}_q^2$$

[SSTX09] Damien Stehlé, Ron Steinfeld, Keisuke Tanaka, and Keita Xagawa. Efficient public key encryption based on ideal lattices. In ASIACRYPT 2009. Springer,2009

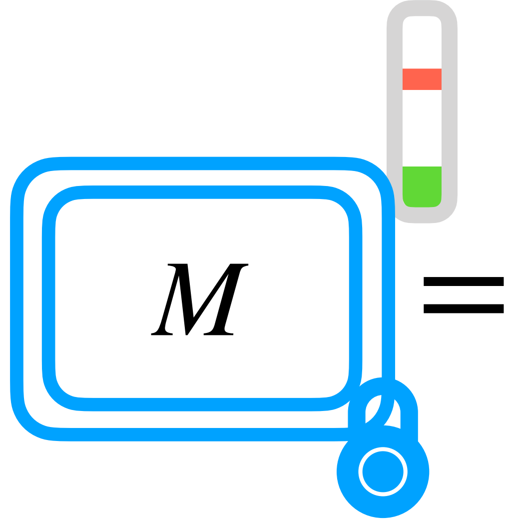
[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In EUROCRYPT 2010. Springer, 2010.

Ring LWE Ciphertexts

Encrypt:

$\mathscr{R}_q = \mathbb{Z}_q[X]/\langle X^N + 1 \rangle$ with N a power of two

$$sk = \boxed{S} \in \mathscr{R}_2$$



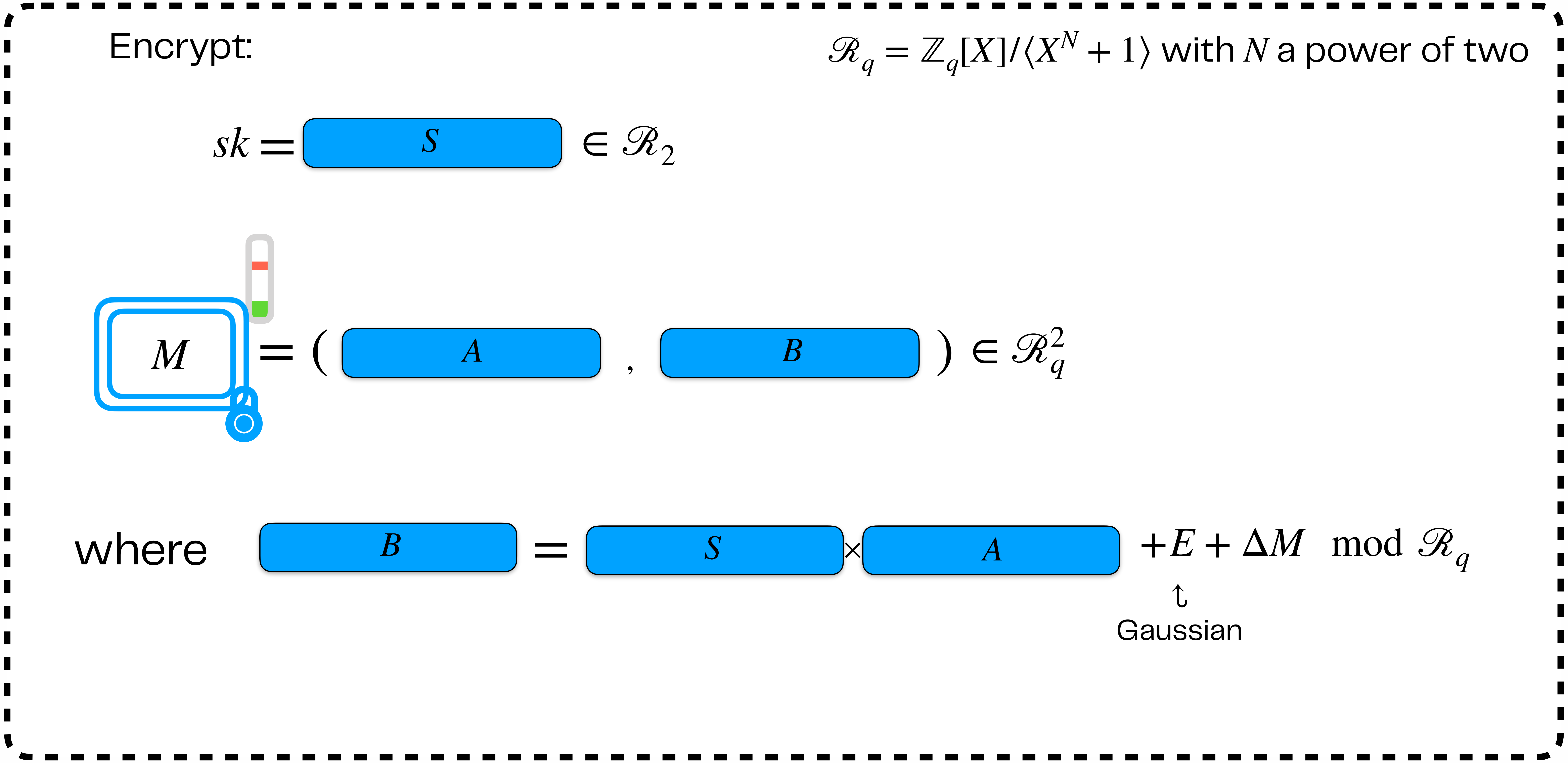
$$\boxed{M} = \left(\boxed{A} , \boxed{B} \right) \in \mathscr{R}_q^2$$

where $\boxed{B} = \boxed{S} \times \boxed{A} + E + \Delta M \mod \mathscr{R}_q$

[SSTX09] Damien Stehlé, Ron Steinfeld, Keisuke Tanaka, and Keita Xagawa. Efficient public key encryption based on ideal lattices. In ASIACRYPT 2009. Springer,2009

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In EUROCRYPT 2010. Springer, 2010.

Ring LWE Ciphertexts



Encrypt:

$\mathscr{R}_q = \mathbb{Z}_q[X]/\langle X^N + 1\rangle$ with N a power of two

$$sk = \boxed{S} \in \mathscr{R}_2$$

$$\boxed{M} = \left(\boxed{A}, \boxed{B}\right) \in \mathscr{R}_q^2$$

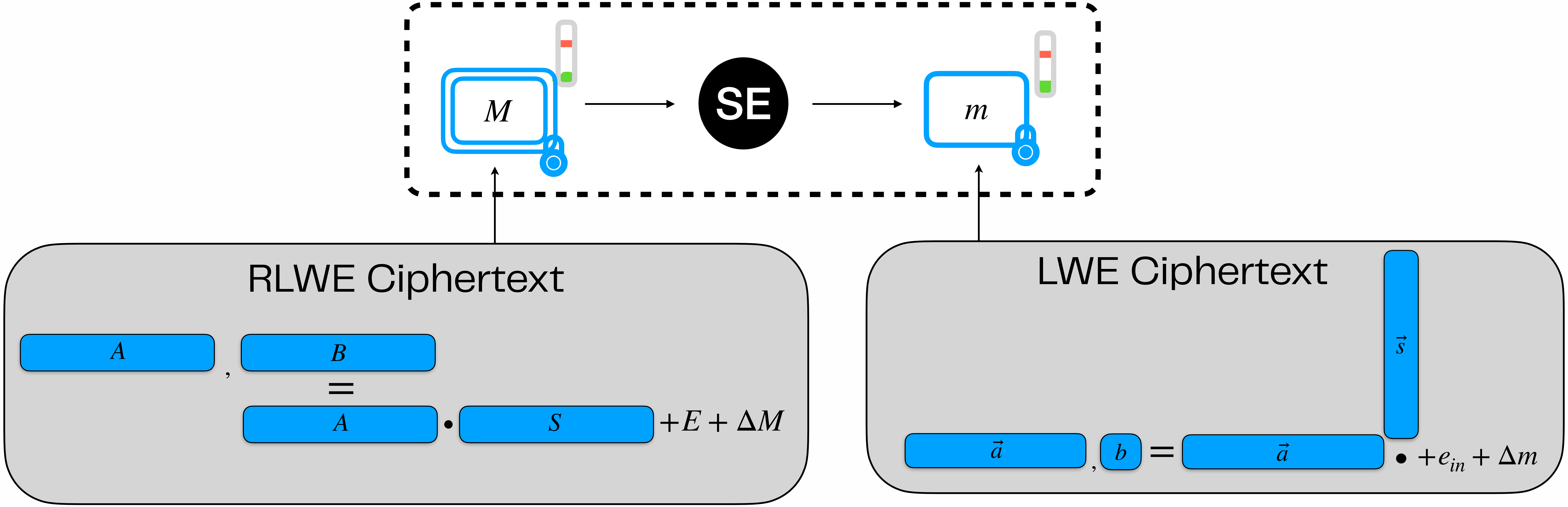
where $\boxed{B} = \boxed{S} \times \boxed{A} + E + \Delta M \mod \mathscr{R}_q$

↑
Gaussian

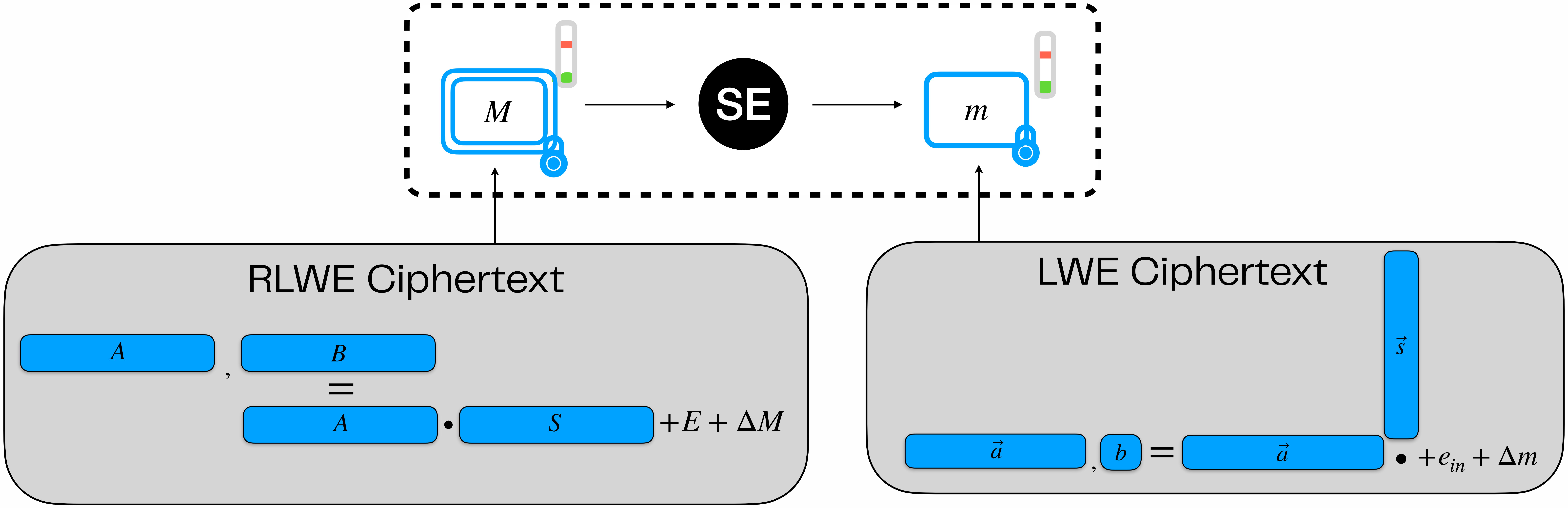
[SSTX09] Damien Stehlé, Ron Steinfeld, Keisuke Tanaka, and Keita Xagawa. Efficient public key encryption based on ideal lattices. In ASIACRYPT 2009. Springer,2009

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In EUROCRYPT 2010. Springer, 2010.

Sample Extract



Sample Extract



RLWE Ciphertext

$A, B = A \bullet S + E + \Delta M$

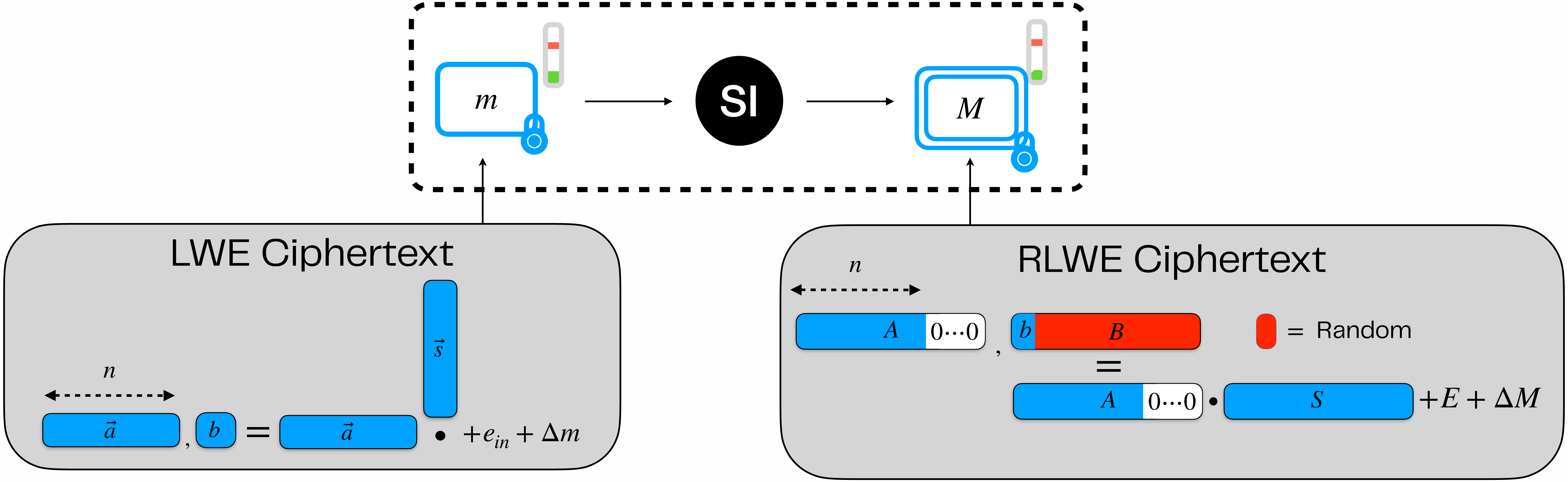
LWE Ciphertext

$\vec{a}, b = \vec{a} \bullet \vec{s} + e_{in} + \Delta m$

Free Computation

Noiseless Operation

Sample Insert



LWE Ciphertext

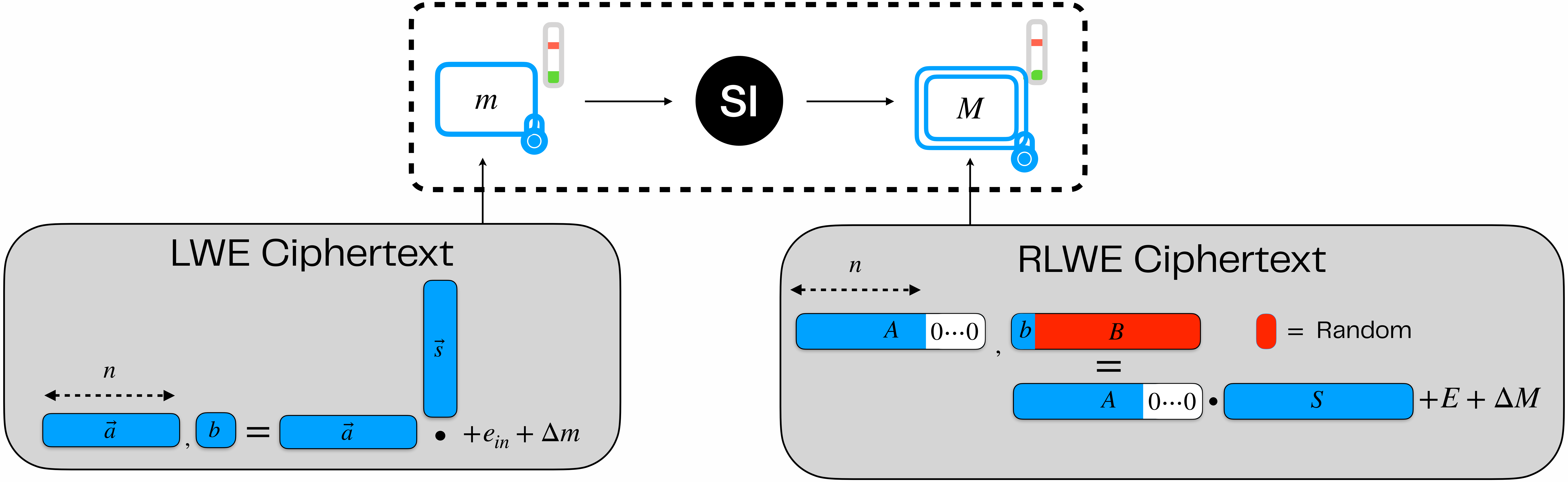
$$\vec{a}, b = \vec{a} \bullet \vec{s} + e_{in} + \Delta m$$

RLWE Ciphertext

$$(A \,|\, 0\cdots0), (b \,|\, B) = (A \,|\, 0\cdots0) \bullet S + E + \Delta M$$

[CDKS20] Hao Chen, Wei Dai, Miran Kim, and Yongsoo Song. Efficient homomorphic conversion between (ring) LWE ciphertexts. IACR Cryptol.ePrint Arch., 2020.

Sample Insert

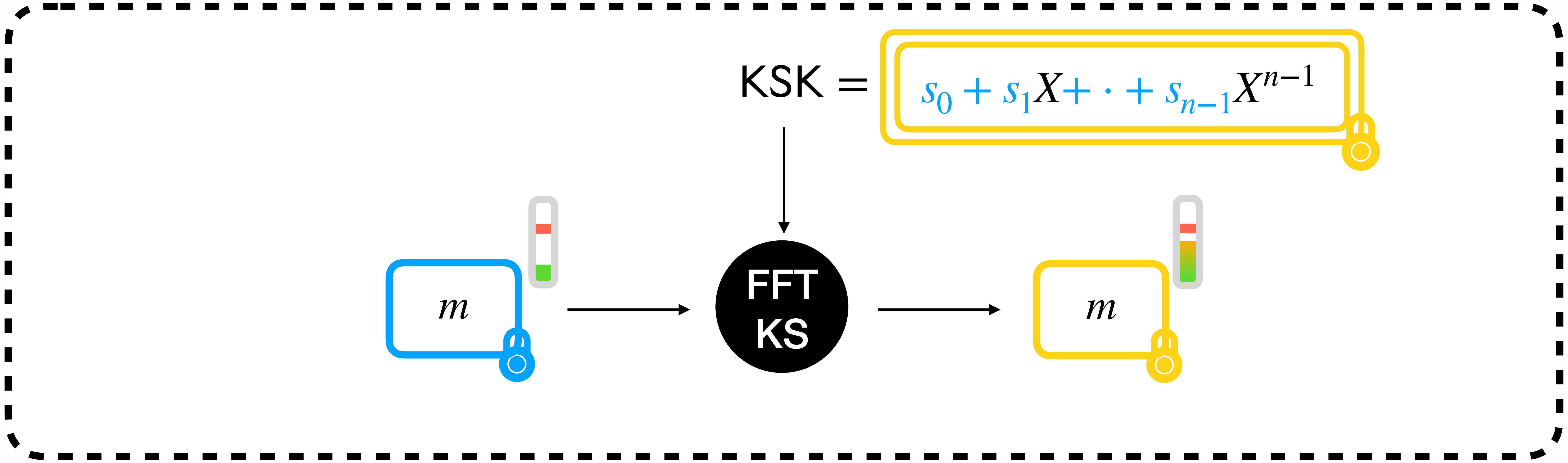


Free Computation

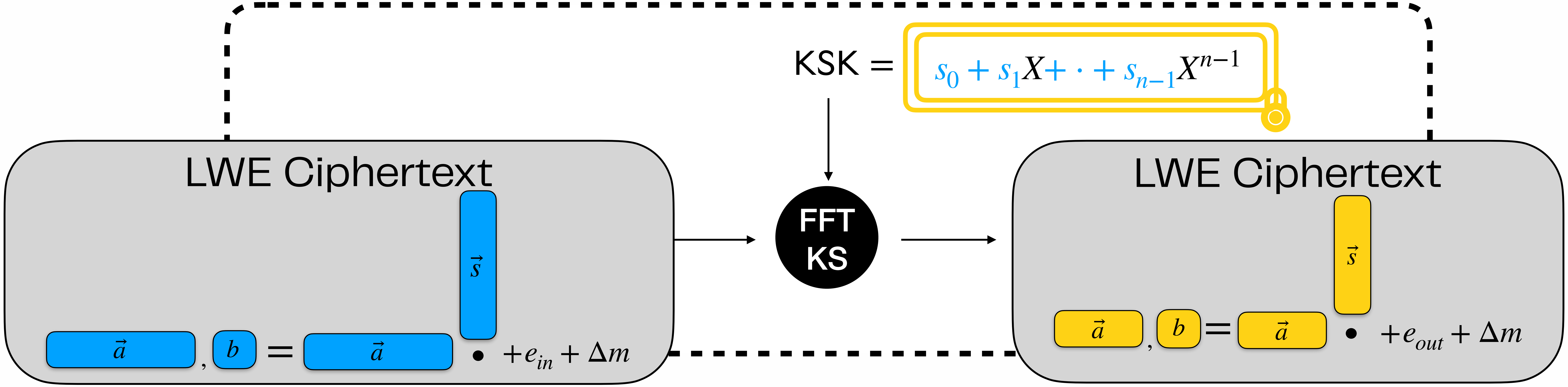
Noiseless Operation

[CDKS20] Hao Chen, Wei Dai, Miran Kim, and Yongsoo Song. Efficient homomorphic conversion between (ring) LWE ciphertexts. IACR Cryptol.ePrint Arch., 2020.

FFT-Keyswitch



FFT-Keyswitch



KSK = $s_0 + s_1X + \cdot + s_{n-1}X^{n-1}$

LWE Ciphertext

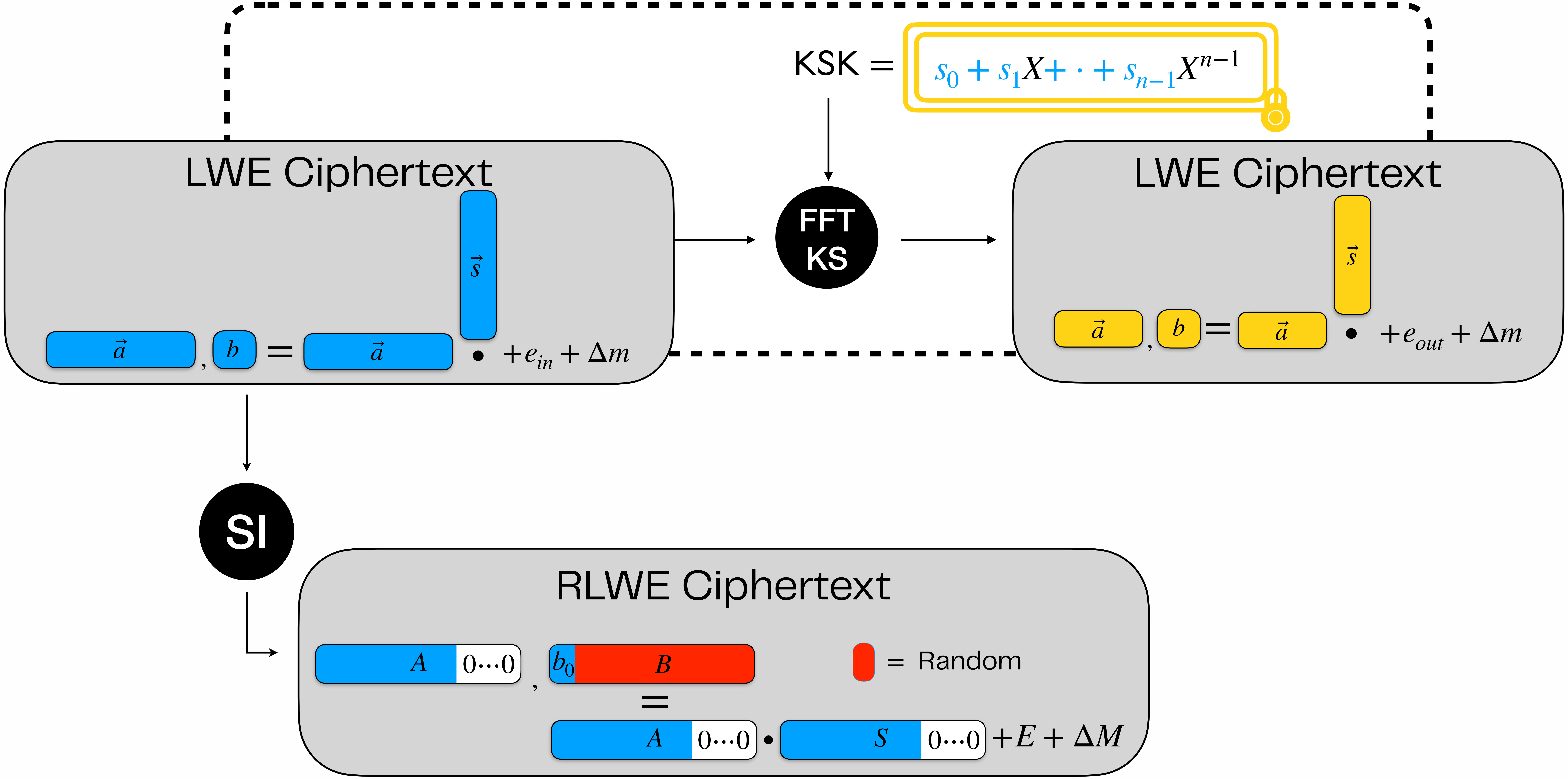
$\vec{a}, b = \vec{a} \bullet \vec{s} + e_{in} + \Delta m$

FFT KS

LWE Ciphertext

$\vec{a}, b = \vec{a} \bullet \vec{s} + e_{out} + \Delta m$

FFT-Keyswitch



KSK = $s_0 + s_1X + \cdot + s_{n-1}X^{n-1}$

LWE Ciphertext

$\vec{a}, b = \vec{a} \bullet \vec{s} + e_{in} + \Delta m$

FFT KS

LWE Ciphertext

$\vec{a}, b = \vec{a} \bullet \vec{s} + e_{out} + \Delta m$

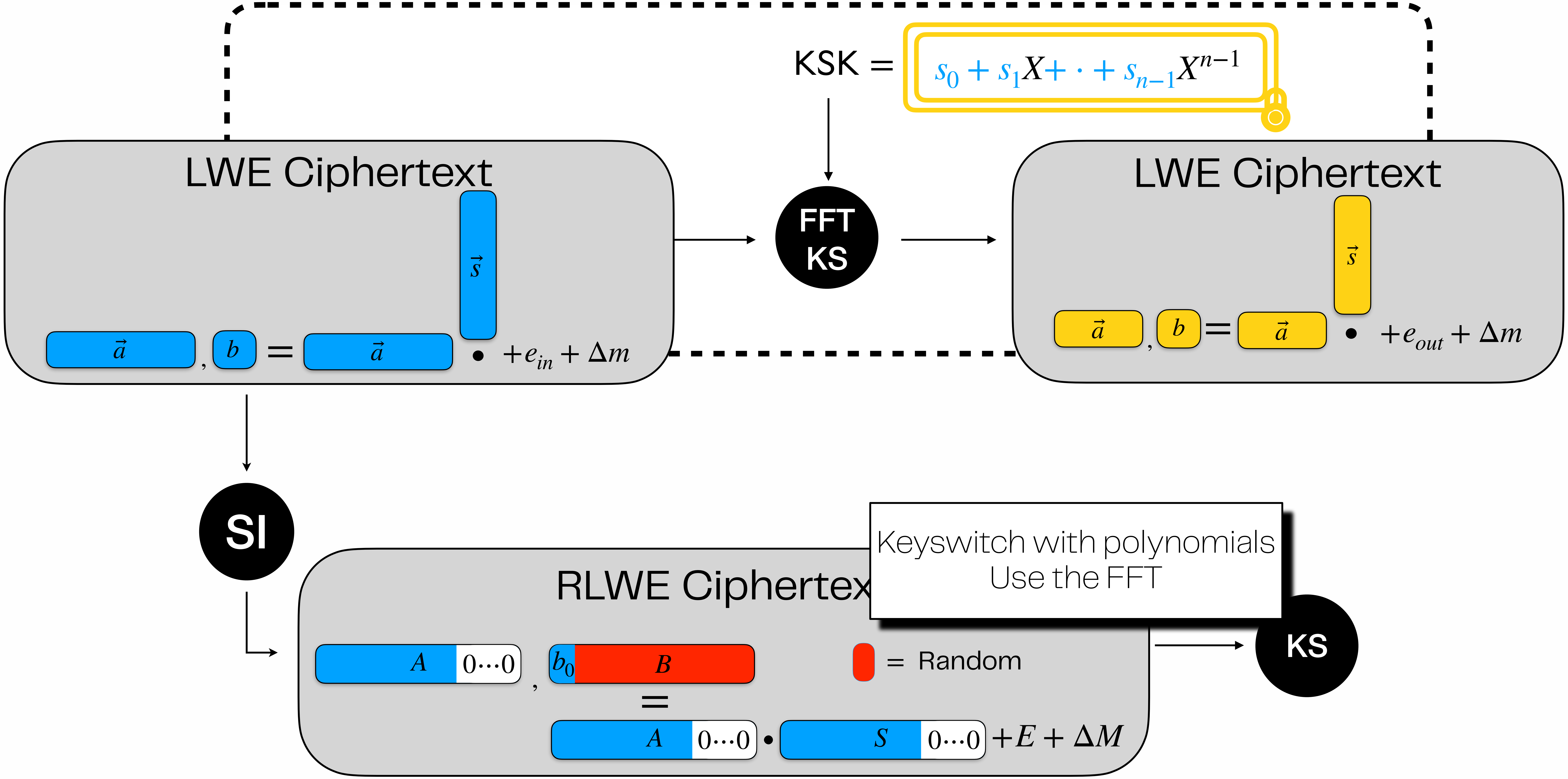
SI

RLWE Ciphertext

$A \; 0 \cdots 0, \; b_0 \; B = A \; 0 \cdots 0 \bullet S \; 0 \cdots 0 + E + \Delta M$

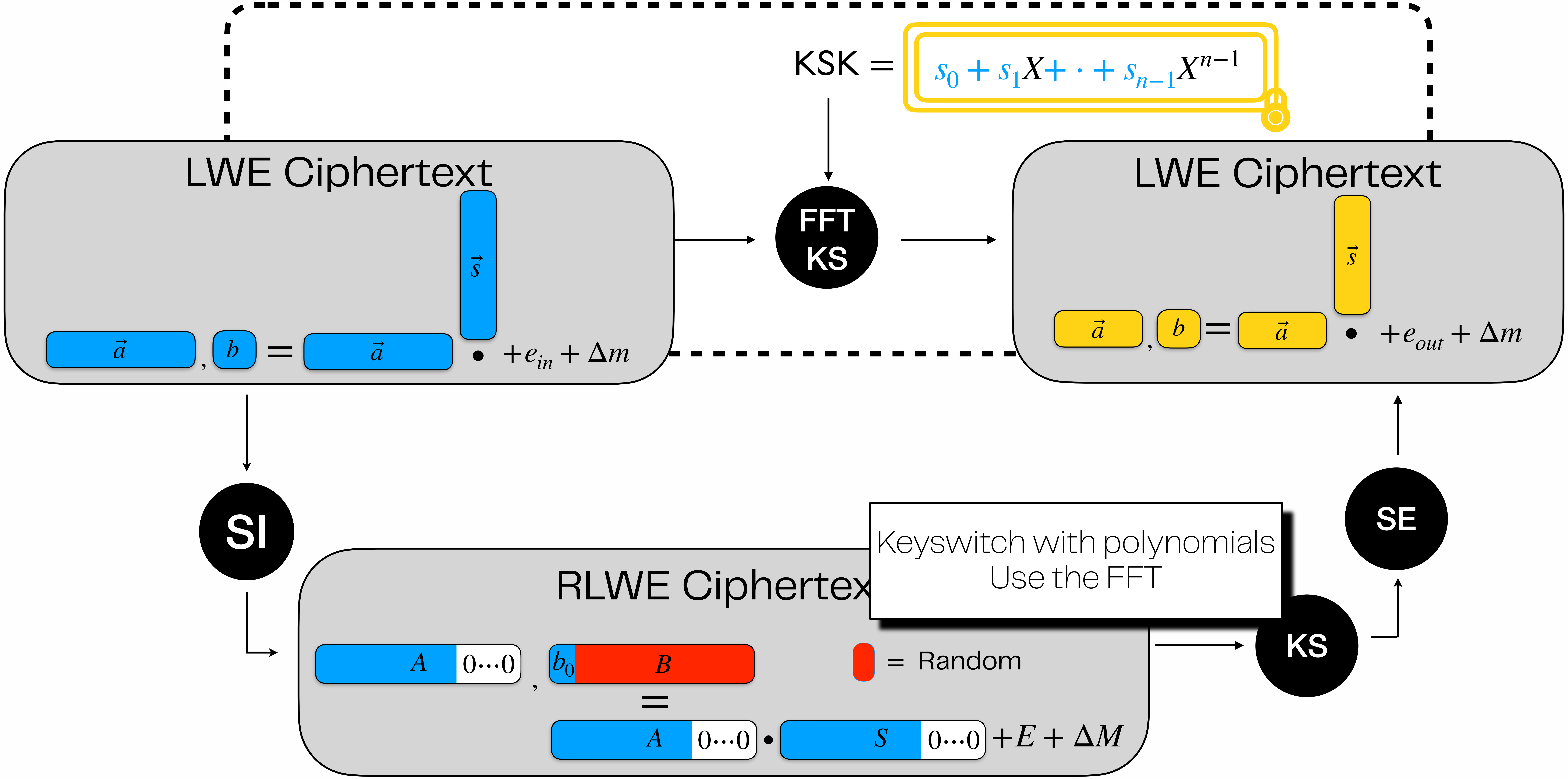
🟥 = Random

FFT-Keyswitch



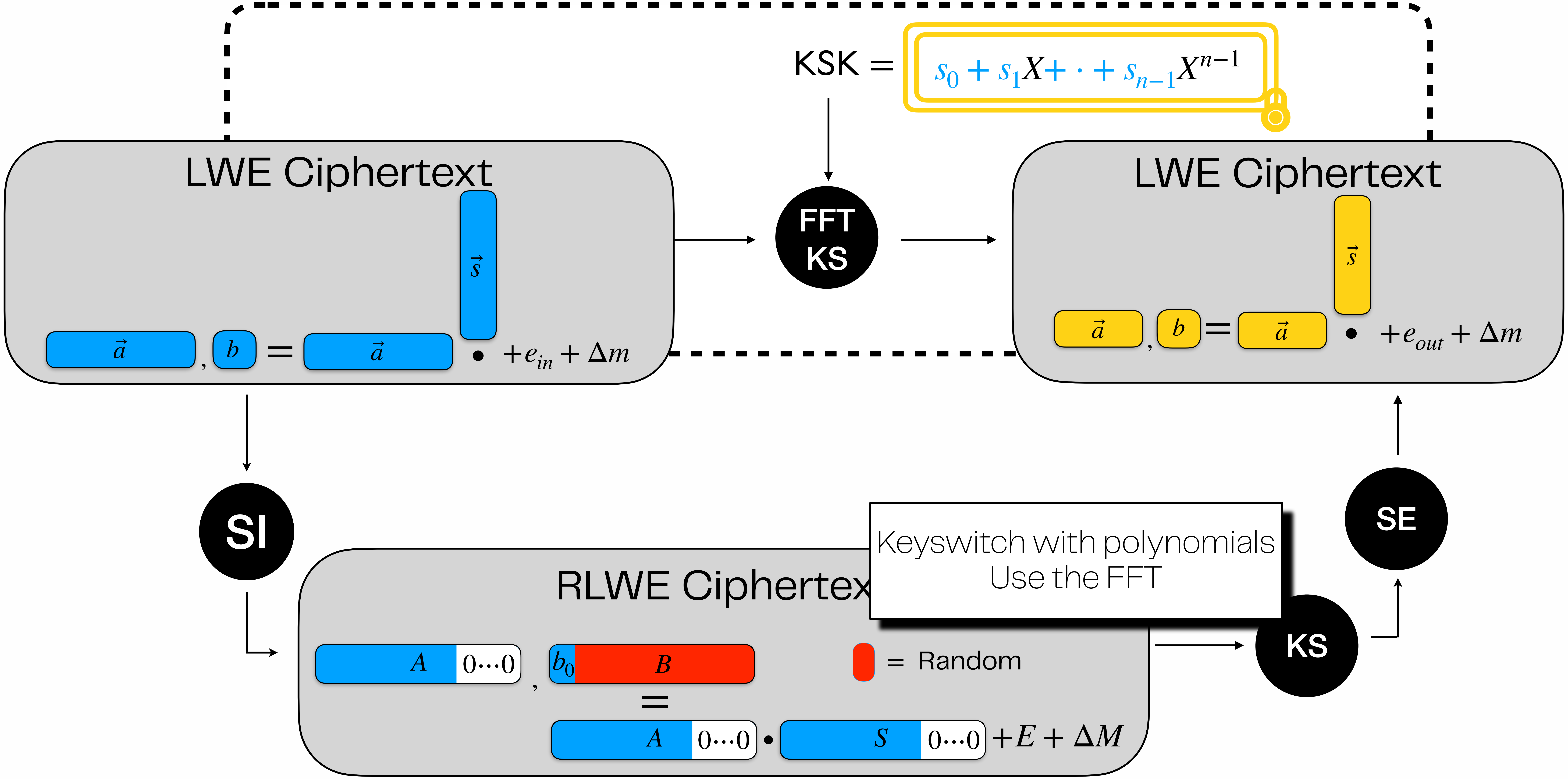
Keyswitch with polynomials
Use the FFT

FFT-Keyswitch



Keyswitch with polynomials
Use the FFT

FFT-Keyswitch



KSK = $s_0 + s_1X + \cdot + s_{n-1}X^{n-1}$

LWE Ciphertext

$\vec{a}, b = \vec{a} \bullet \vec{s} + e_{in} + \Delta m$

FFT KS

LWE Ciphertext

$\vec{a}, b = \vec{a} \bullet \vec{s} + e_{out} + \Delta m$

SI

RLWE Ciphertext

$A\ 0\cdots0,\ b_0\ B = A\ 0\cdots0 \bullet S\ 0\cdots0 + E + \Delta M$

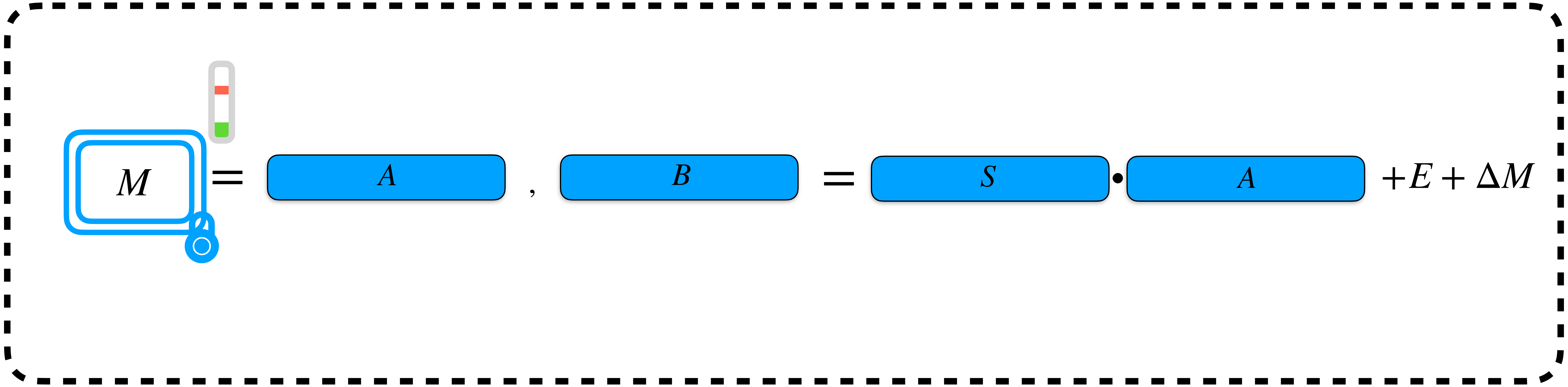
🟥 = Random

KS

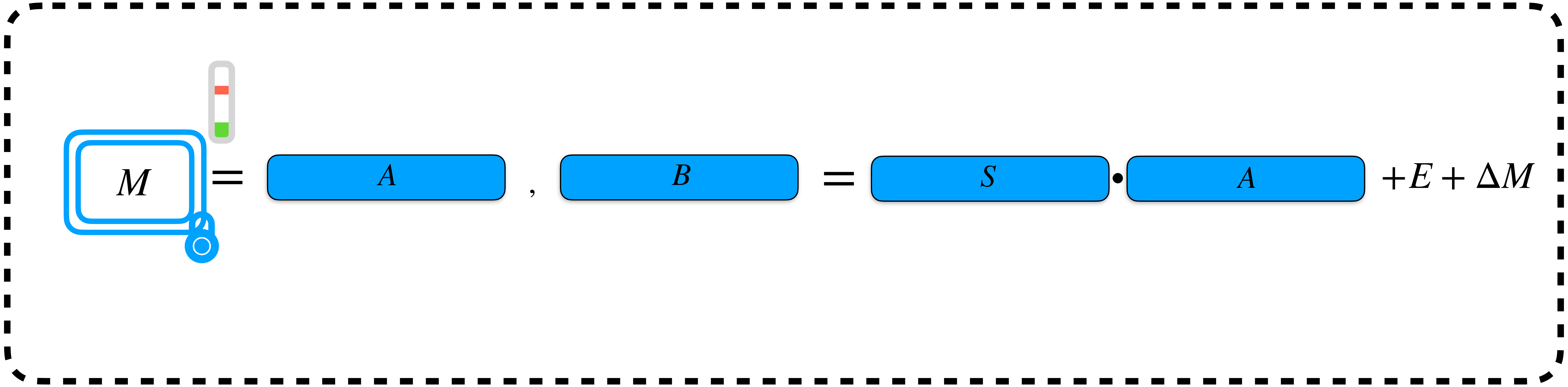
SE

Keyswitch with polynomials
Use the FFT

Partial Secret Key



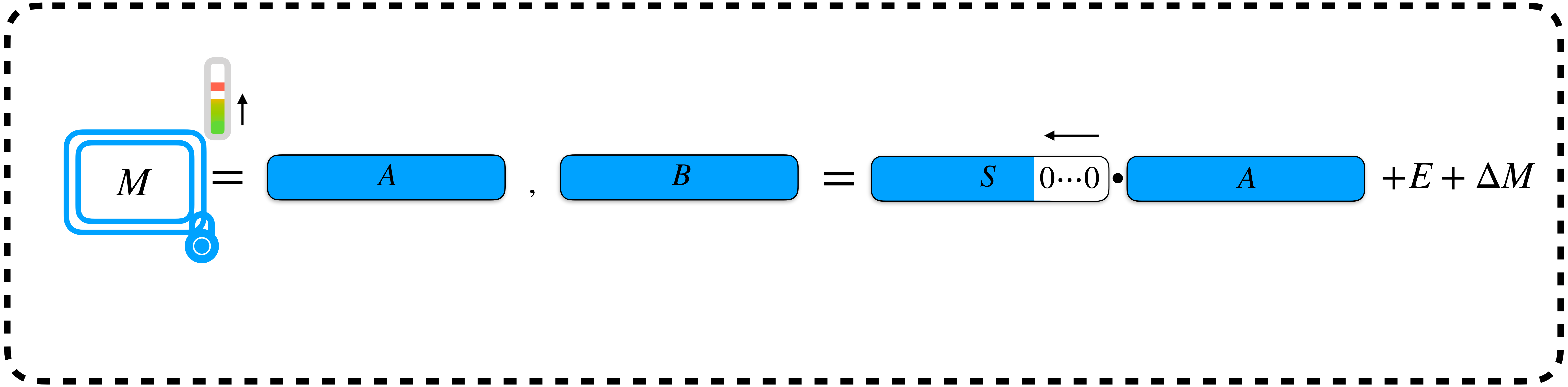
Partial Secret Key



$$M = A \;,\; B = S \bullet A + E + \Delta M$$

Limited to polynomials of a degree that is a power of two.

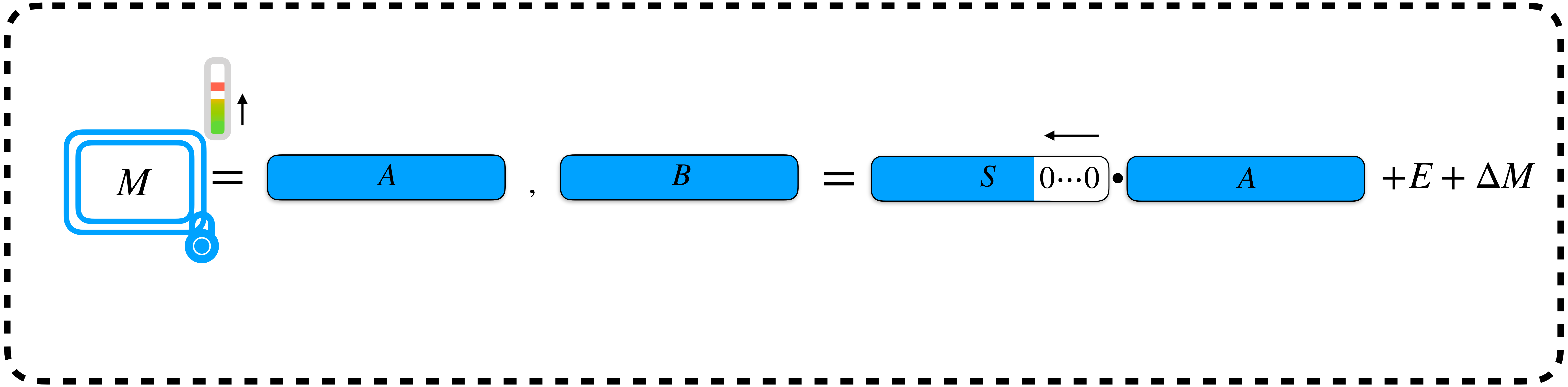
Partial Secret Key



Reduce the number of unknown coefficients

Add more noise to keep the security

Partial Secret Key

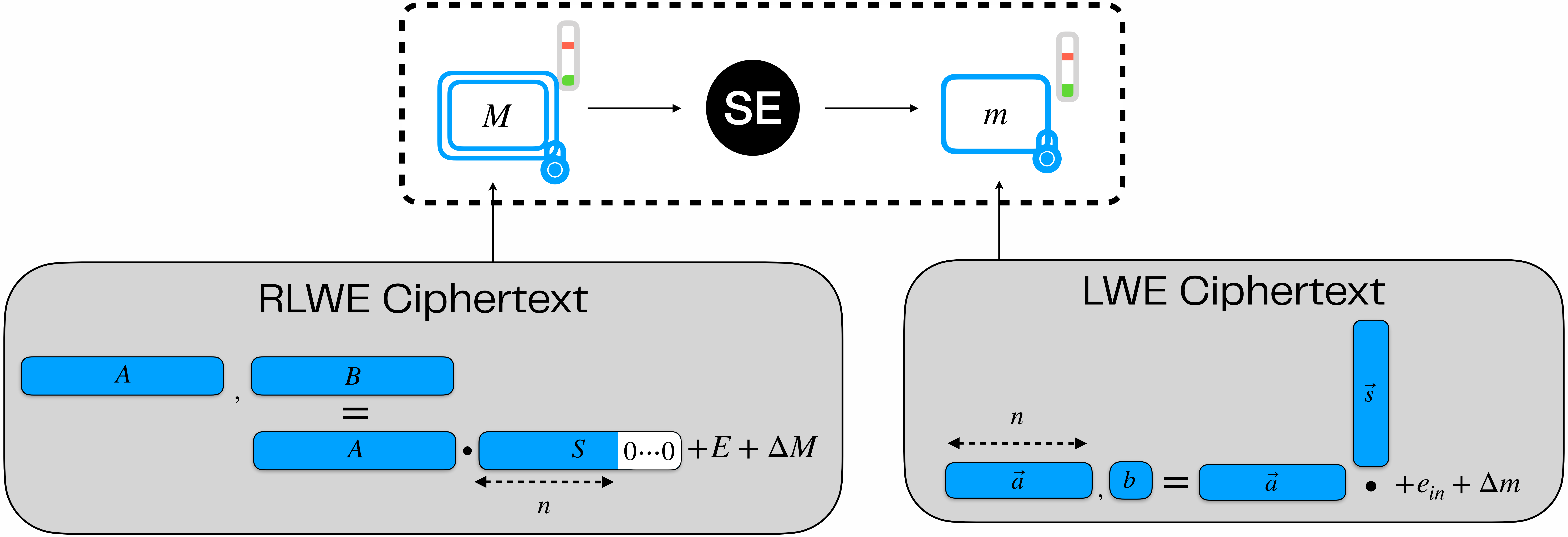


Reduce the number of unknown coefficients

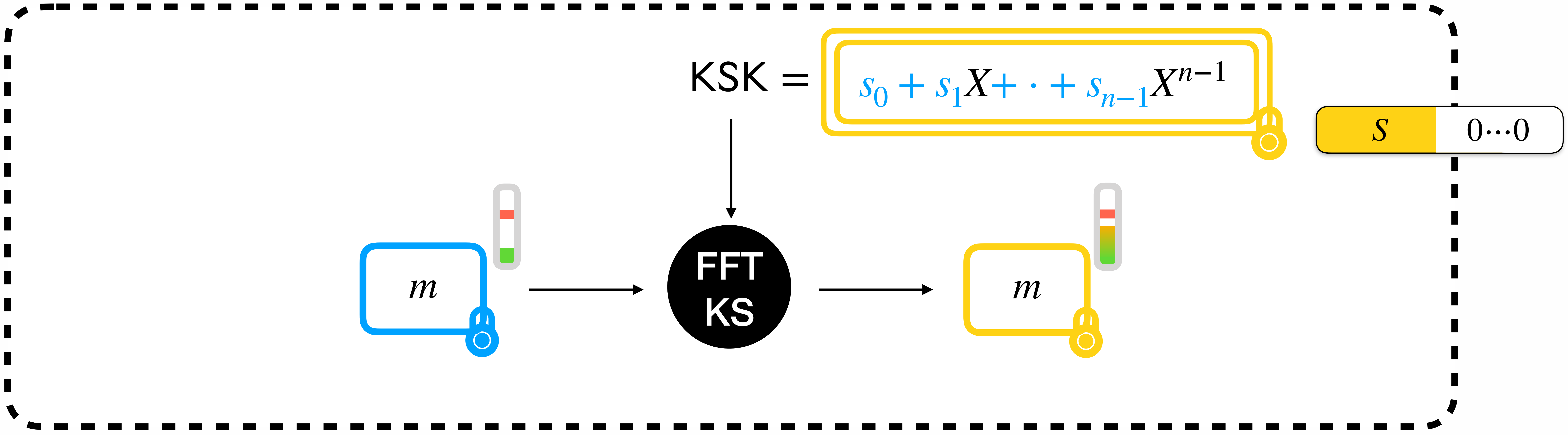
Add more noise to keep the security

The number of secret elements is no longer limited to a power of two

Sample Extract

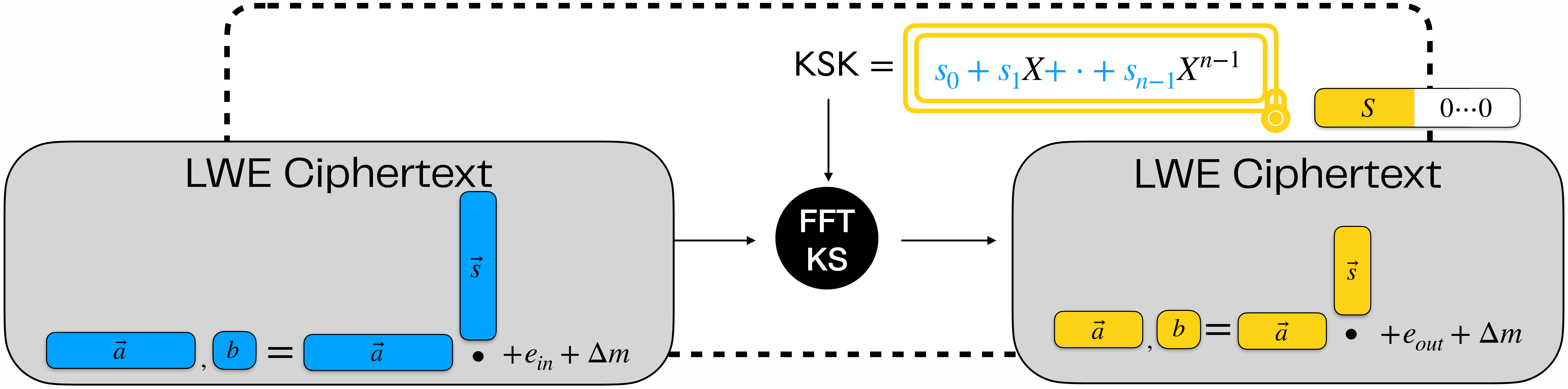


FFT-Keyswitch



KSK = $s_0 + s_1X + \cdot + s_{n-1}X^{n-1}$

FFT-Keyswitch



KSK = $s_0 + s_1X + \cdot + s_{n-1}X^{n-1}$

S | $0 \cdots 0$

LWE Ciphertext

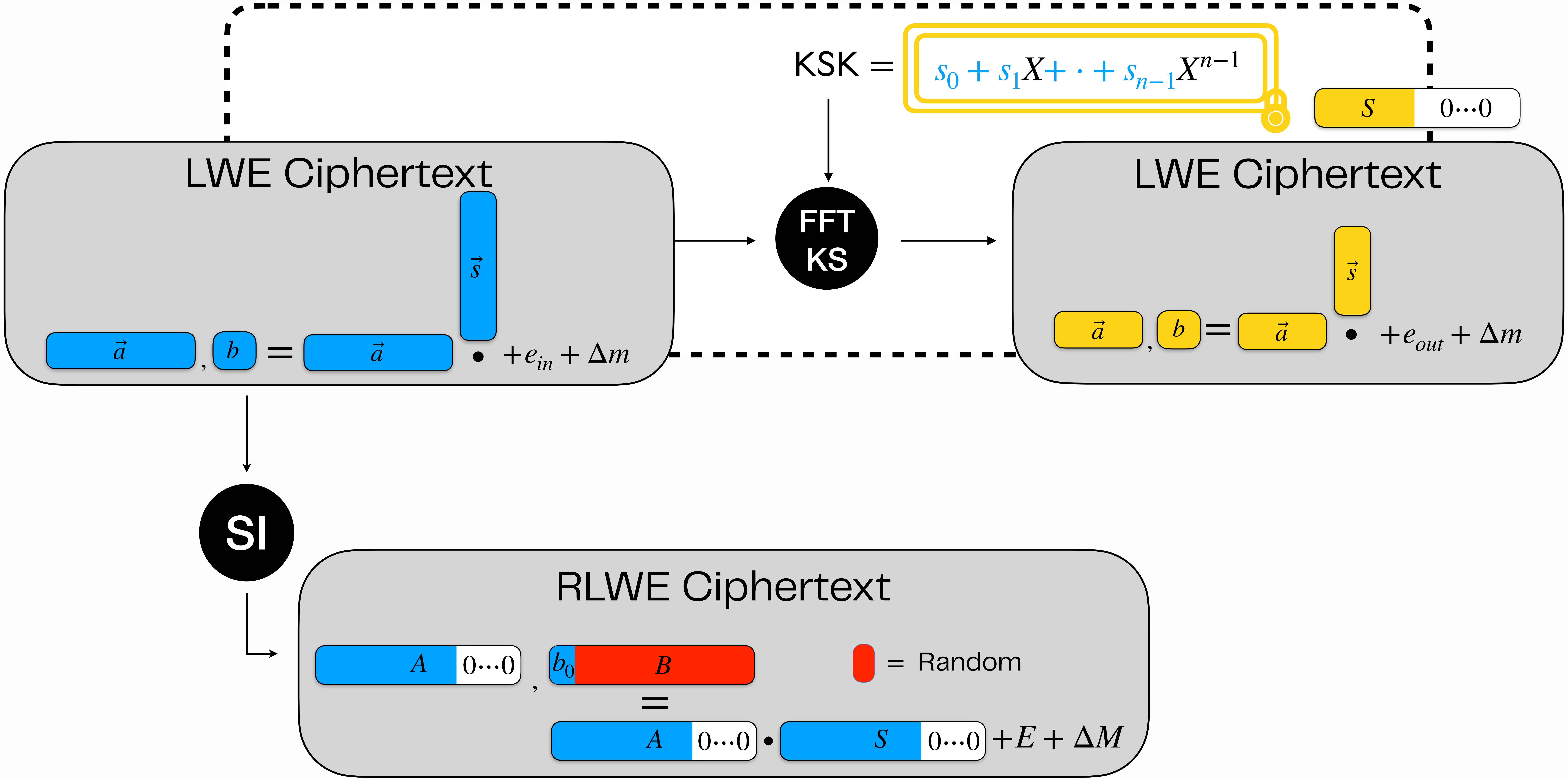
$\vec{a}, b = \vec{a} \bullet \vec{s} + e_{in} + \Delta m$

FFT KS

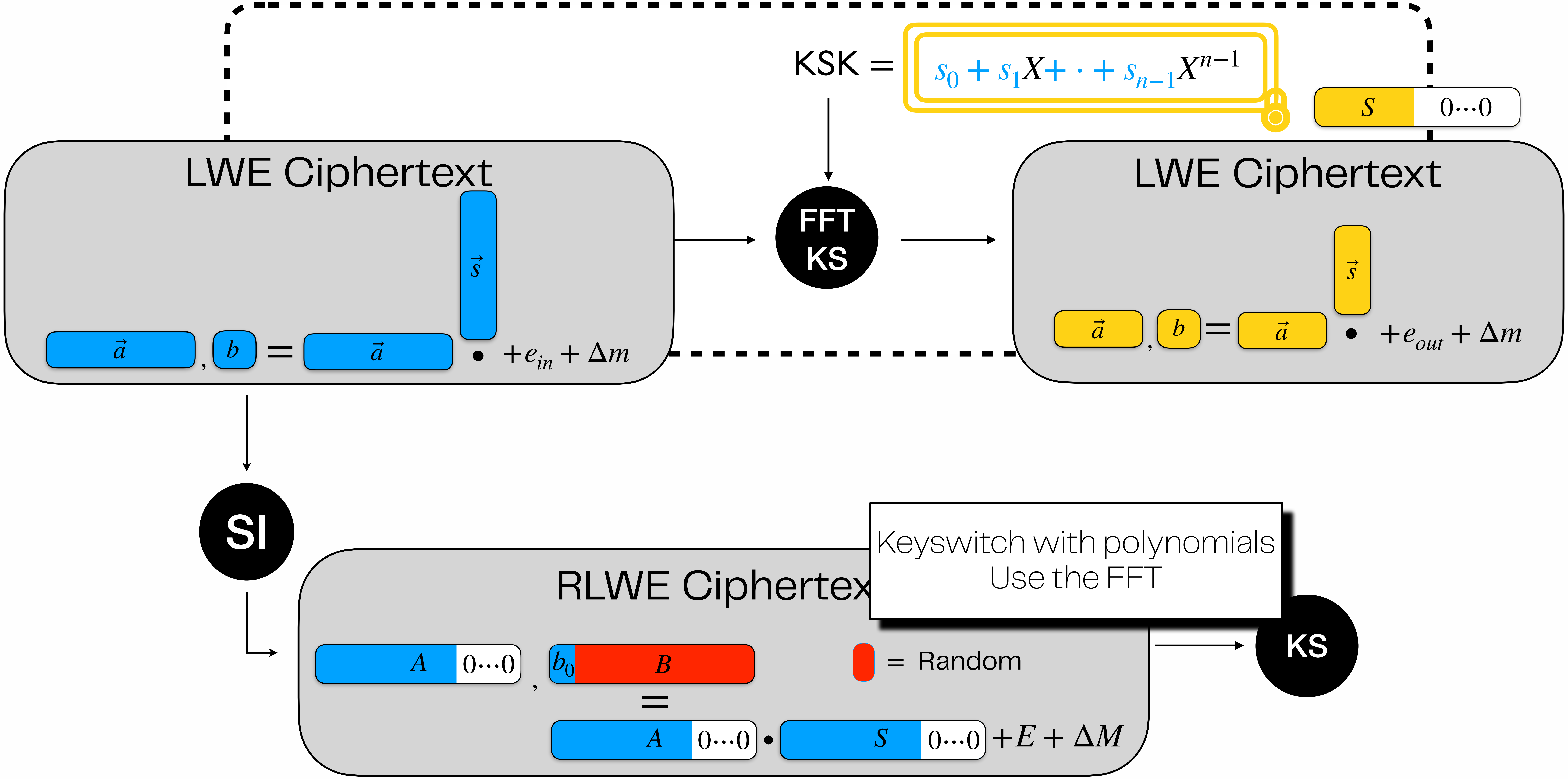
LWE Ciphertext

$\vec{a}, b = \vec{a} \bullet \vec{s} + e_{out} + \Delta m$

FFT-Keyswitch



FFT-Keyswitch



KSK = $s_0 + s_1X + \cdot + s_{n-1}X^{n-1}$

$S \; 0\cdots0$

LWE Ciphertext

$\vec{a}, b = \vec{a} \bullet \vec{s} + e_{in} + \Delta m$

FFT KS

LWE Ciphertext

$\vec{a}, b = \vec{a} \bullet \vec{s} + e_{out} + \Delta m$

SI

RLWE Ciphertext

$A \; 0\cdots0, \; b_0 \, B$

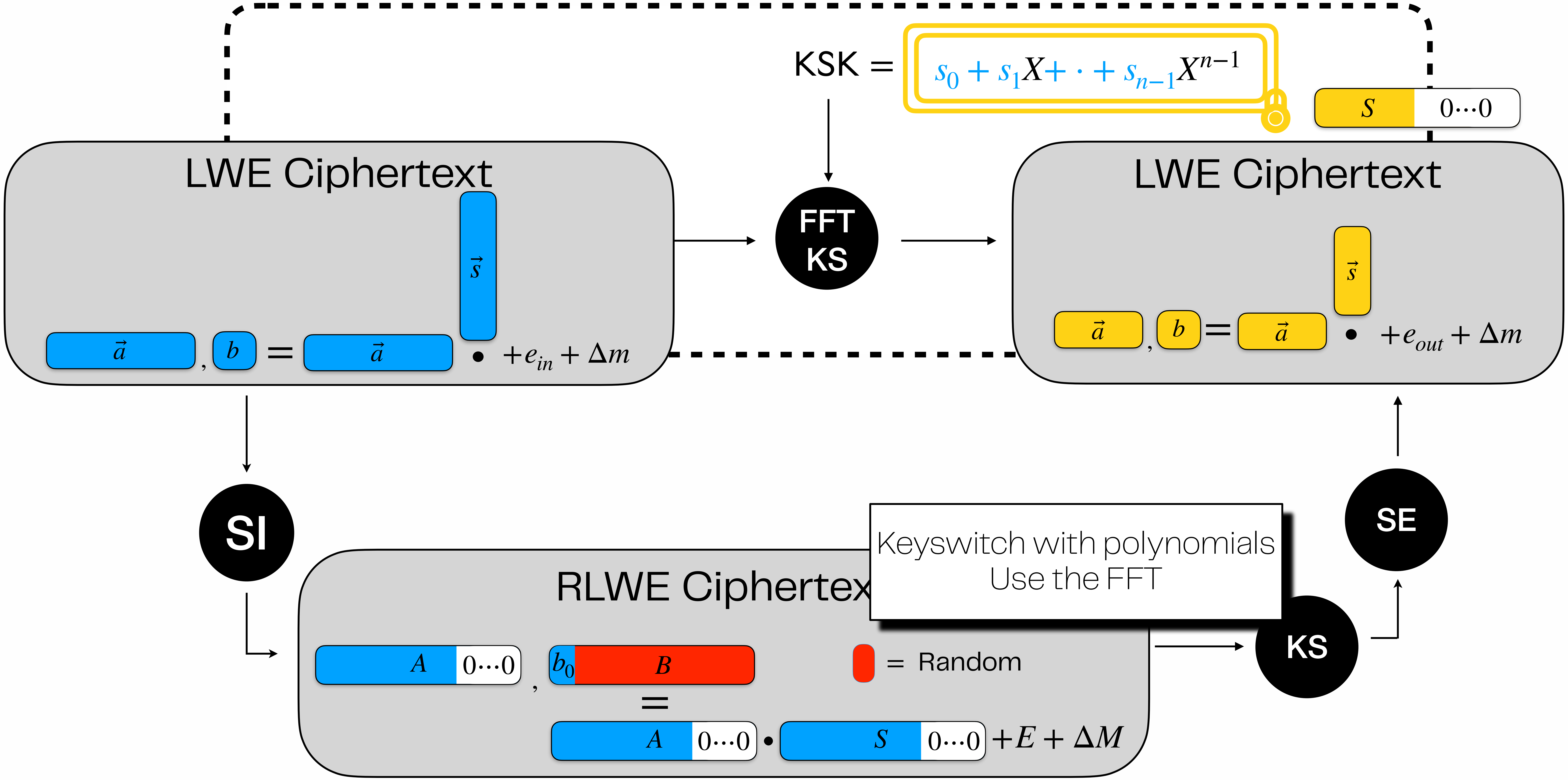
$= A \; 0\cdots0 \bullet S \; 0\cdots0 + E + \Delta M$

🟥 = Random

Keyswitch with polynomials
Use the FFT

KS

FFT-Keyswitch



Keyswitch with polynomials
Use the FFT

Advantages of Partial Secret Keys

Use the FFT → Better complexity

Smaller key-switching key

Advantages of Partial Secret Keys

Use the FFT → Better complexity

Smaller key-switching key

Less noise added in the Bootstrapping graph

More parameters choices

Advantages of Partial Secret Keys

Use the FFT → Better complexity

Smaller key-switching key

Less noise added in the Bootstrapping graph

More parameters choices

Can be combined with the secret keys with shared randomness

Benchmarks

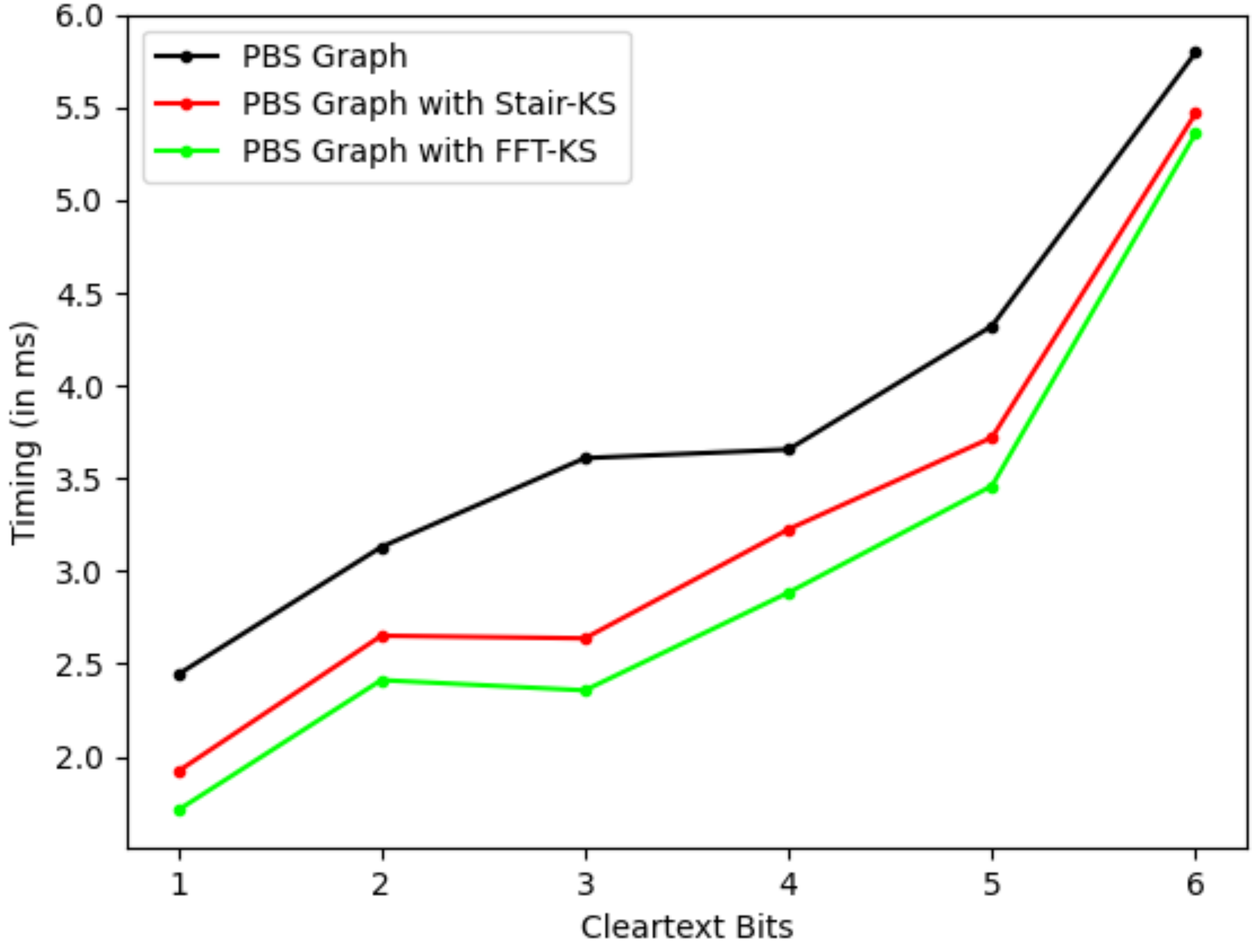
Bootstrapping Graph

with $P_{\text{fail}} = 2^{-14}$

using TFHE-rs

Speed-ups between **1.3** and **2.4**

Results based on **new assumptions**



AWS M6i.metal Intel Xeon 8375C (Ice Lake) 128 vCPUs& 512GiB

Conclusion

Conclusion

Can we explore **new assumptions** to improve the **bootstrapping** graph ?

Two new Assumptions

Partial Secret Keys

Secret Keys with Shared Randomness

Novelties

New Algorithms

Noise Analysis

Security Analysis

Practical Results

Reduction of the public materials between **1.5** and **2.7**

Speed-ups between **1.3** and **2.4**

Thank you.

eprint 2023/979

To appear in CCS 2024

ZAMA

Contact and Links

loris.bergerat@zama.ai

zama.ai

Github

Community links