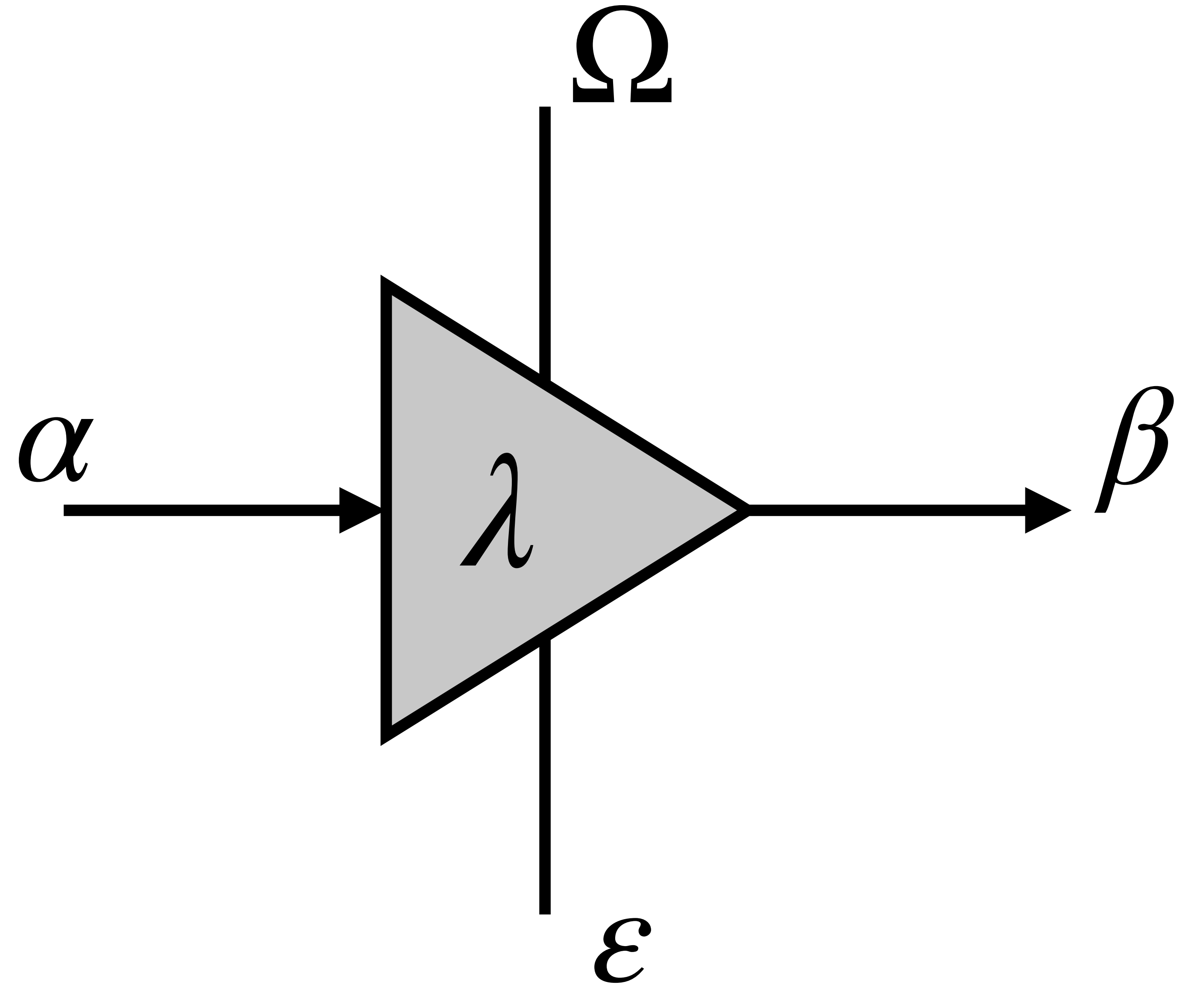
LWE-based Cryptography
Elementary Principles and Constructions



Tomáš Rosa, Ph.D.
Cryptology and Biometrics Competence Centre, Raiffeisenbank, Prague
Faculty of Mathematics and Physics, Charles University, Prague

The **Two Flavors** of Quantum-Resistant Mechanisms

- **Cryptographic protocols based on quantum mechanics laws**
  - Quantum Key Distribution (QKD), for instance
  - unconditionally secure, provided everything in the whole scheme is
  - speed versus distance limits
  - cloud limits or even impossibility
  - not every classical scheme has its practical quantum variant, e.g. signatures
  - security authorities NSA, BSI, NCSC, ANSSI stay highly reserved at this moment
- **Classical algorithms for classical computing platforms**
  - post-quantum cryptographic suites
  - recommended widespread approach and our main topic here

The Algorithmic Approach of PQC

Traditional cryptosystems		Purpose	PQC Replacements	
Integer factorization	RSA	Encryption	Crystals-Kyber (ML-KEM, FIPS 203)	Learning with errors
Discrete logarithm	ElGamal			
	DH			
Elliptic curve discrete logarithm	ECDH			
Integer factorization	RSA	Signature	Crystals-Dilithium (ML-DSA, FIPS 204)	Learning with errors Short integer solution
Discrete logarithm	DSA		Falcon (FN-DSA, FIPS 206)*	
Elliptic curve discrete logatithm	ECDSA		SPHINCS+ (SLH-DSA, FIPS 205)	Hash inversion

*) FIPS 206 draft is "… planned for late 2024."

Learning With Errors (LWE)
standard, decision version

Definition 1. For positive integers m, n, q, and $\beta < q$, the $\mathsf{LWE}_{n,m,q,\beta}$ problem asks to distinguish between the following two distributions:

1. $(\mathbf{A}, \mathbf{A}\mathbf{s} + \mathbf{e})$, where $\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}, \mathbf{s} \leftarrow [\beta]^m, \mathbf{e} \leftarrow [\beta]^n$
2. $(\mathbf{A}, \mathbf{u})$, where $\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}$ and $\mathbf{u} \leftarrow \mathbb{Z}_q^n$.

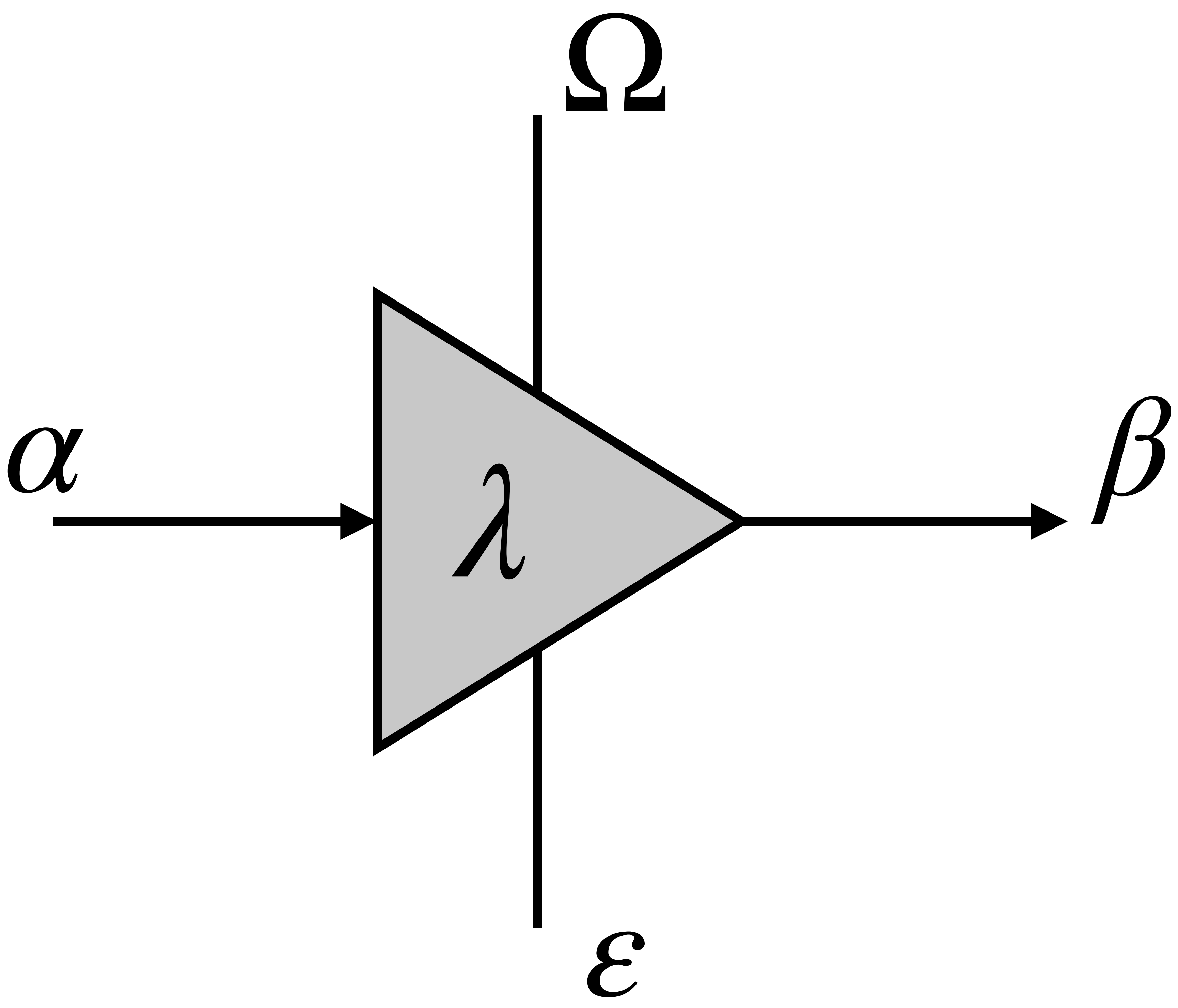
$a \leftarrow S$ means that a is chosen uniformly at random from the set S

$$[\beta] = \{-\beta, \ldots, -1, 0, 1, \ldots, \beta\}$$

furthermore, in practice, we usually set $m = n$

[Lyubashevsky, https://ia.cr/2024/1287], cf. also [Peikert, https://ia.cr/2015/939]

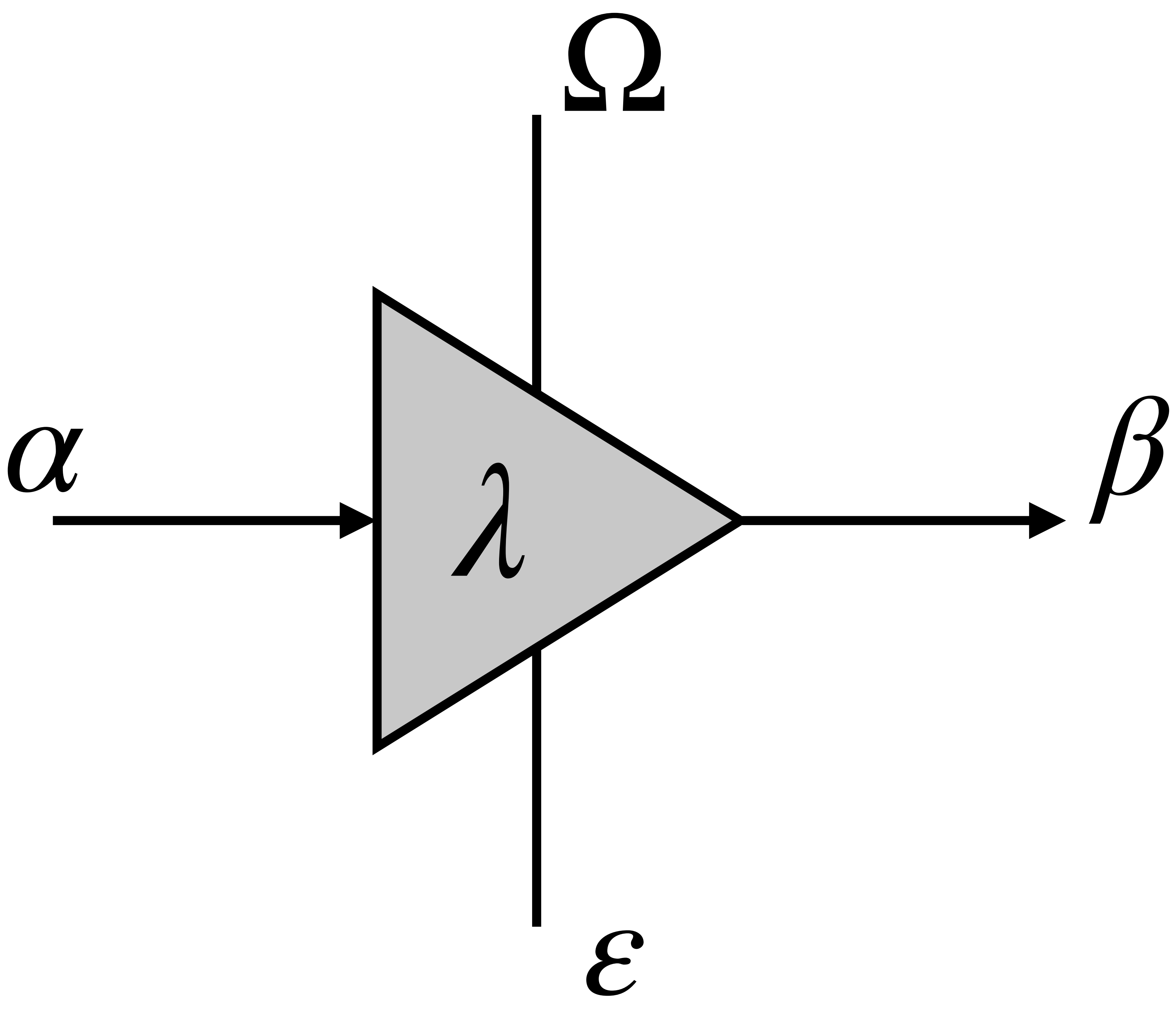
LWE Gate - General Definition



$$\Omega \times \alpha + \varepsilon = \beta$$

Standard-LWE λ_0	$\Omega \in \mathbb{F}_q^{n \times m} = \mathbb{Z}_q^{n \times m}$ $\alpha \in \mathbb{F}_q^m$ $\beta, \varepsilon \in \mathbb{F}_q^n$
Ring-LWE λ_ρ	$\Omega \in R_q = \mathbb{Z}_q[x] / \langle x^n + 1 \rangle$ $\alpha \in R_q$ $\beta, \varepsilon \in R_q$
Module-LWE λ_μ	$\Omega \in R_q^{n \times m}, R_q$ *see above* $\alpha \in R_q^m$ $\beta, \varepsilon \in R_q^n$

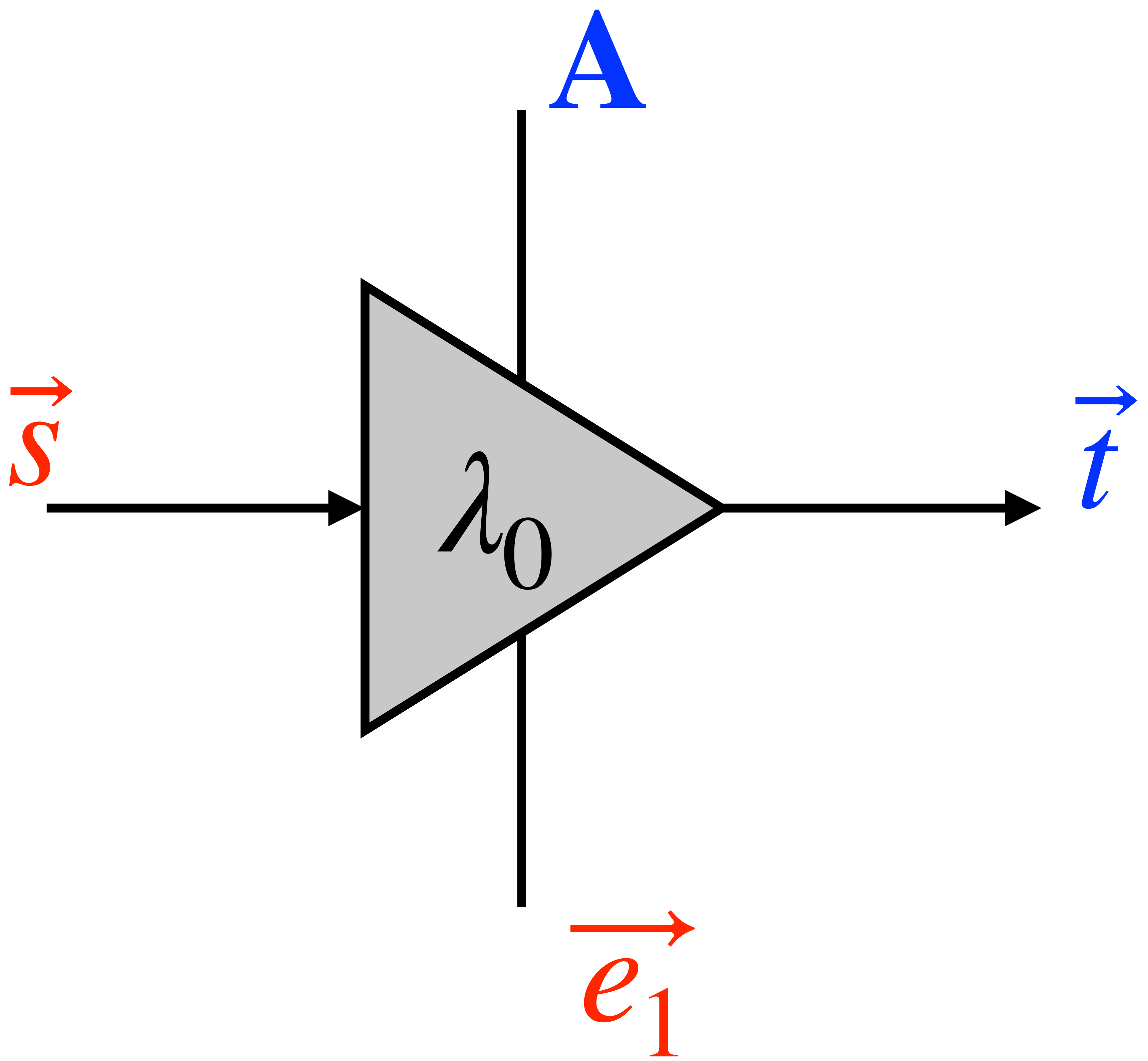
LWE Gate - Security Arguments



$$\Omega \times \alpha + \varepsilon = \beta$$

Standard-LWE λ_0	β indistinguishable from $u \leftarrow [\mathbb{Z}_q^n]$ in particular, $\beta \mapsto \alpha$ is hard
Ring-LWE λ_ρ	β indistinguishable from $u \leftarrow [R_q]$ in particular, $\beta \mapsto \alpha$ is hard
Module-LWE λ_μ	β indistinguishable from $u \leftarrow [R_q^n]$ in particular, $\beta \mapsto \alpha$ is hard

Standard-LWE Encryption Scheme
setup phase



$\vec{e_1} \leftarrow [\beta_2]^m$

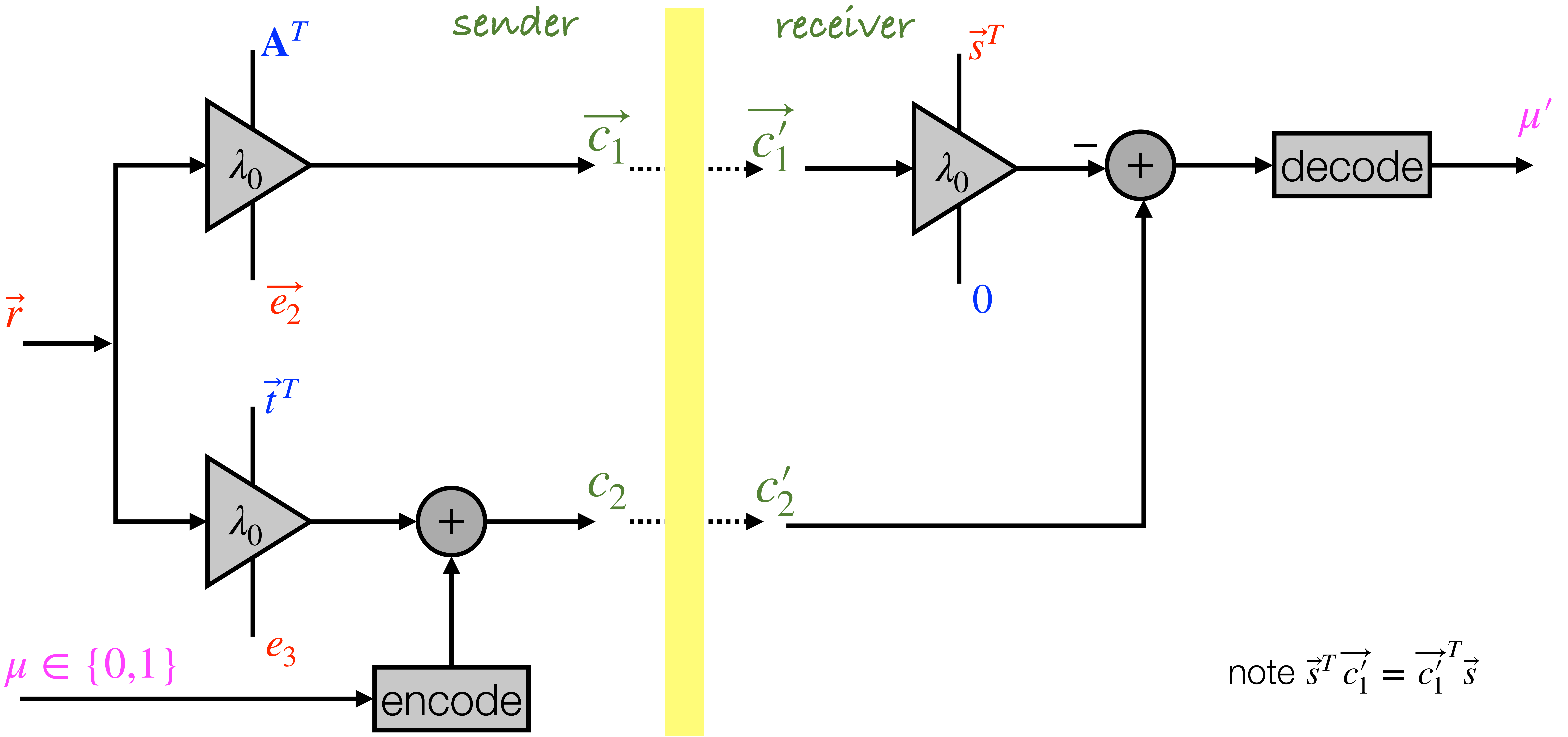
sk: $\vec{s} \leftarrow [\beta_1]^m$

pk: $\mathbf{A} \leftarrow \mathbb{Z}_q^{m \times m}$

pk: $\vec{t} = \mathbf{A}\vec{s} + \vec{e_1}$

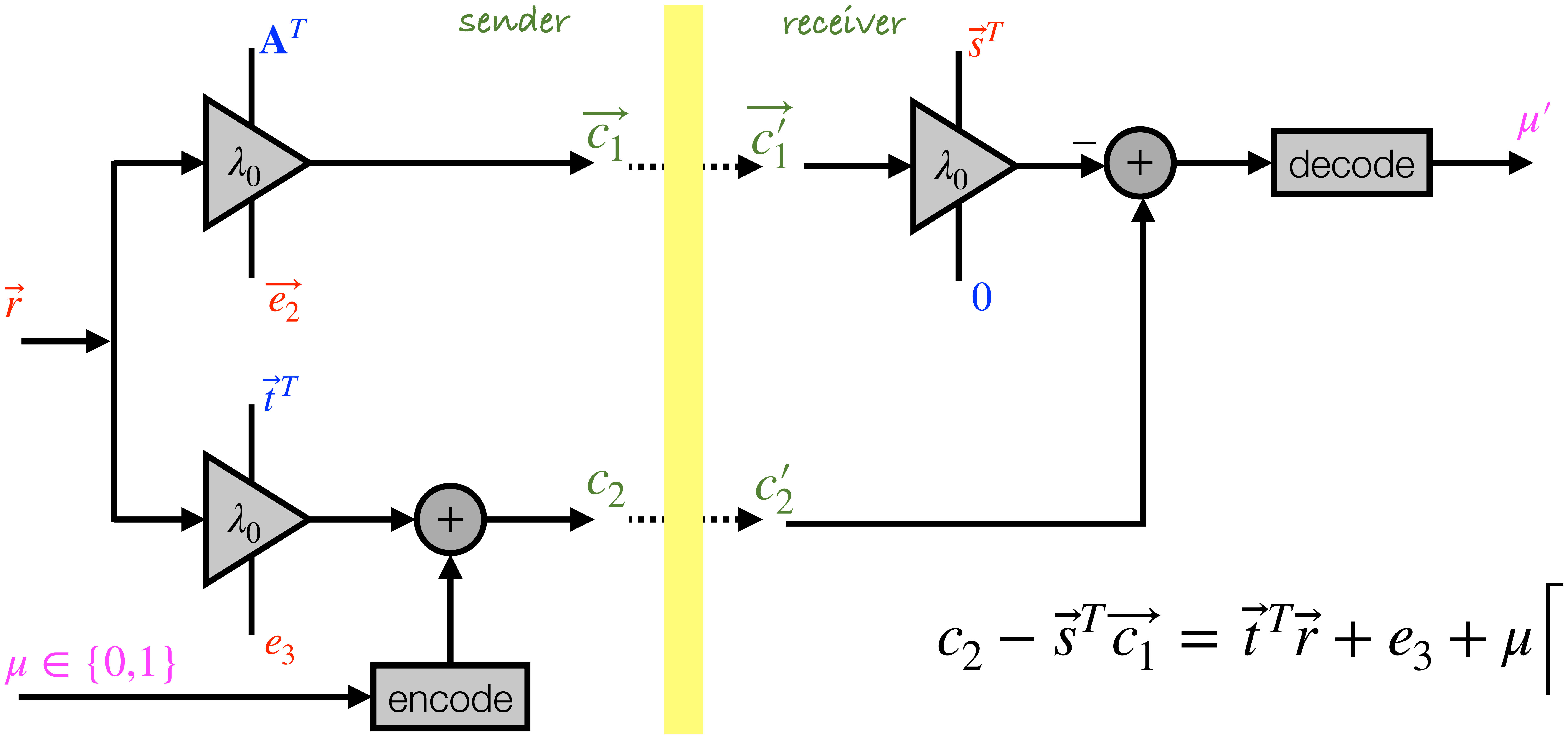
we set $m = n$, for the general LWE gate

Standard-LWE Encryption Scheme
encryption/decryption of one-bit messages



note $\vec{s}^T \vec{c_1'} = \vec{c_1'}^T \vec{s}$

Standard-LWE Encryption Scheme
encryption/decryption of one-bit messages



$$\vec{r} \leftarrow [\beta_1]^m$$

$$\vec{e_2} \leftarrow [\beta_2]^m,\ e_3 \leftarrow [\beta_2]$$

$$\vec{c_1} = \mathbf{A}^T\vec{r} + \vec{e_2}$$

$$c_2 = \vec{t}^T\vec{r} + e_3 + \mu\left\lceil \frac{q}{2} \right\rfloor$$

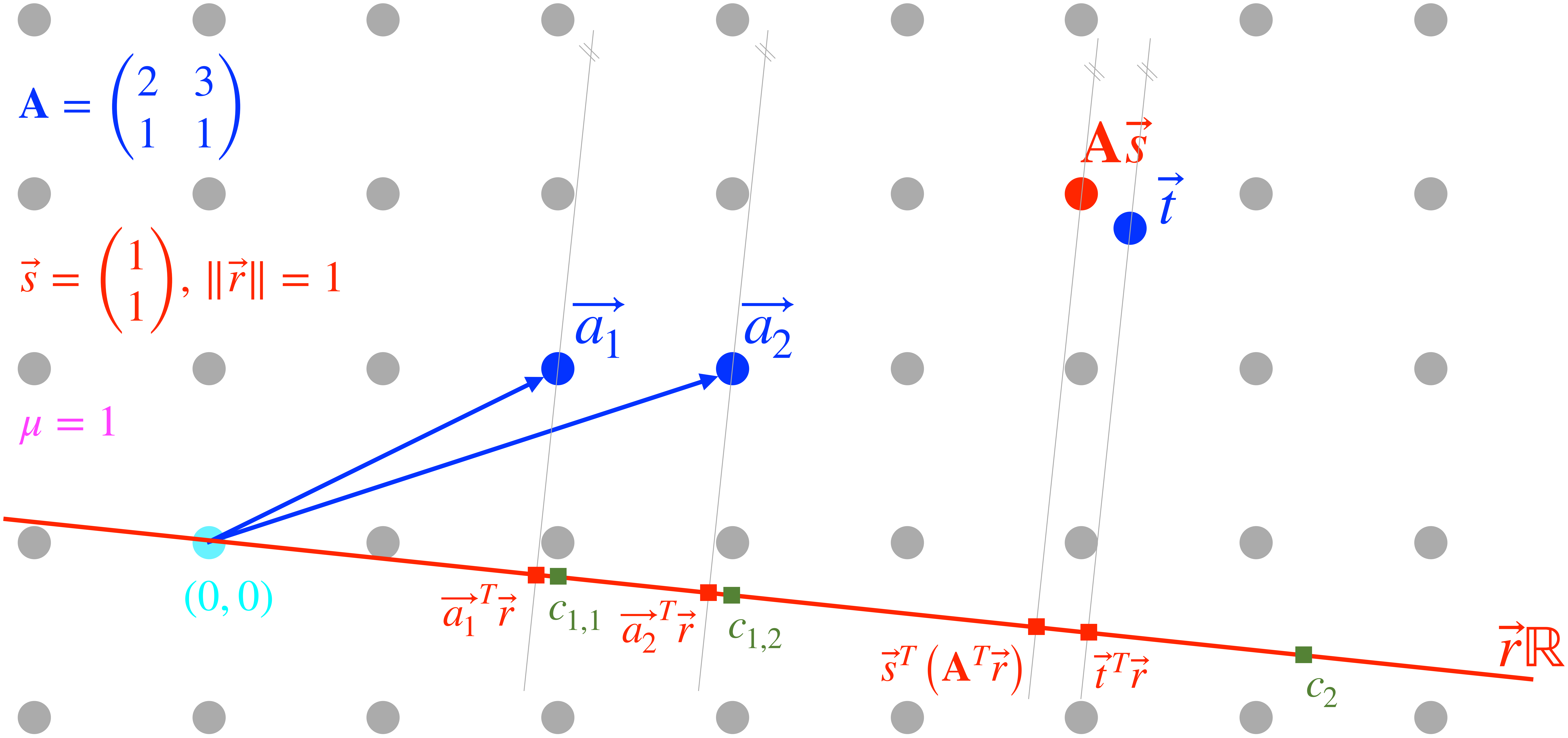
$$c_2 - \vec{s}^T\vec{c_1} = \vec{t}^T\vec{r} + e_3 + \mu\left\lceil \frac{q}{2} \right\rfloor - \vec{t}^T\vec{r} + \vec{e_1}^T\vec{r} - \vec{s}^T\vec{e_2}$$

$$= \vec{e_1}^T\vec{r} - \vec{s}^T\vec{e_2} + e_3 + \mu\left\lceil \frac{q}{2} \right\rfloor$$

sk: $\vec{s} \leftarrow [\beta]^m$, pk: $\left(\mathbf{A} \leftarrow \mathbb{Z}_q^{m\times m},\ \vec{t} = \mathbf{A}\vec{s} + \vec{e_1}\right)$, where: $\vec{e_1} \leftarrow [\beta]^m$

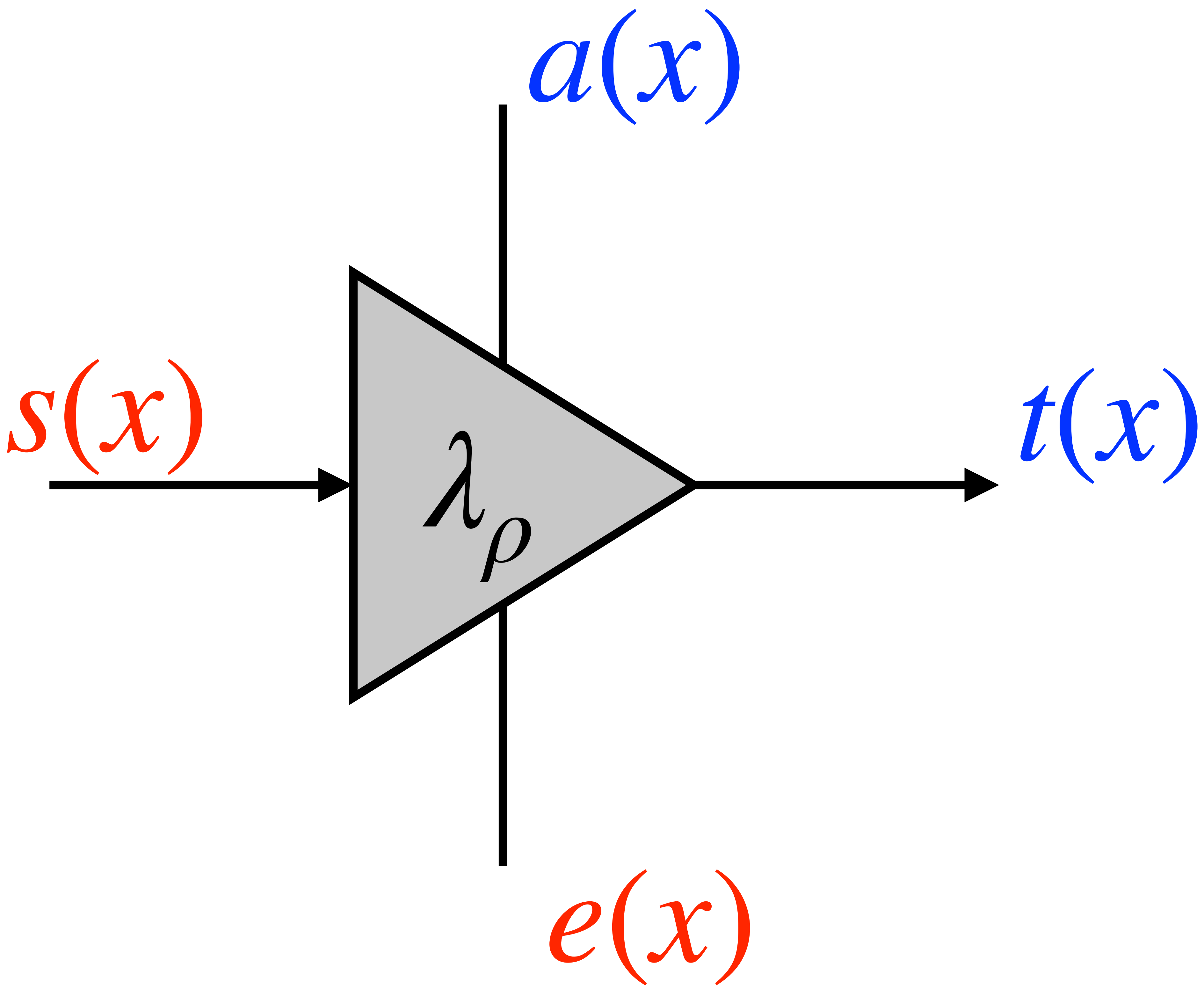
note $\vec{s}^T\mathbf{A}^T = \vec{t}^T - \vec{e_1}^T$

Geometric interpretation invoking adjoint operator mechanics.



we ignore modulo q and stay with simple $\mathbb{Z}^2 \subset \mathbb{R}^2$

Ring-LWE Encryption Scheme
setup phase



$e_1(x) \leftarrow [\beta_2]$

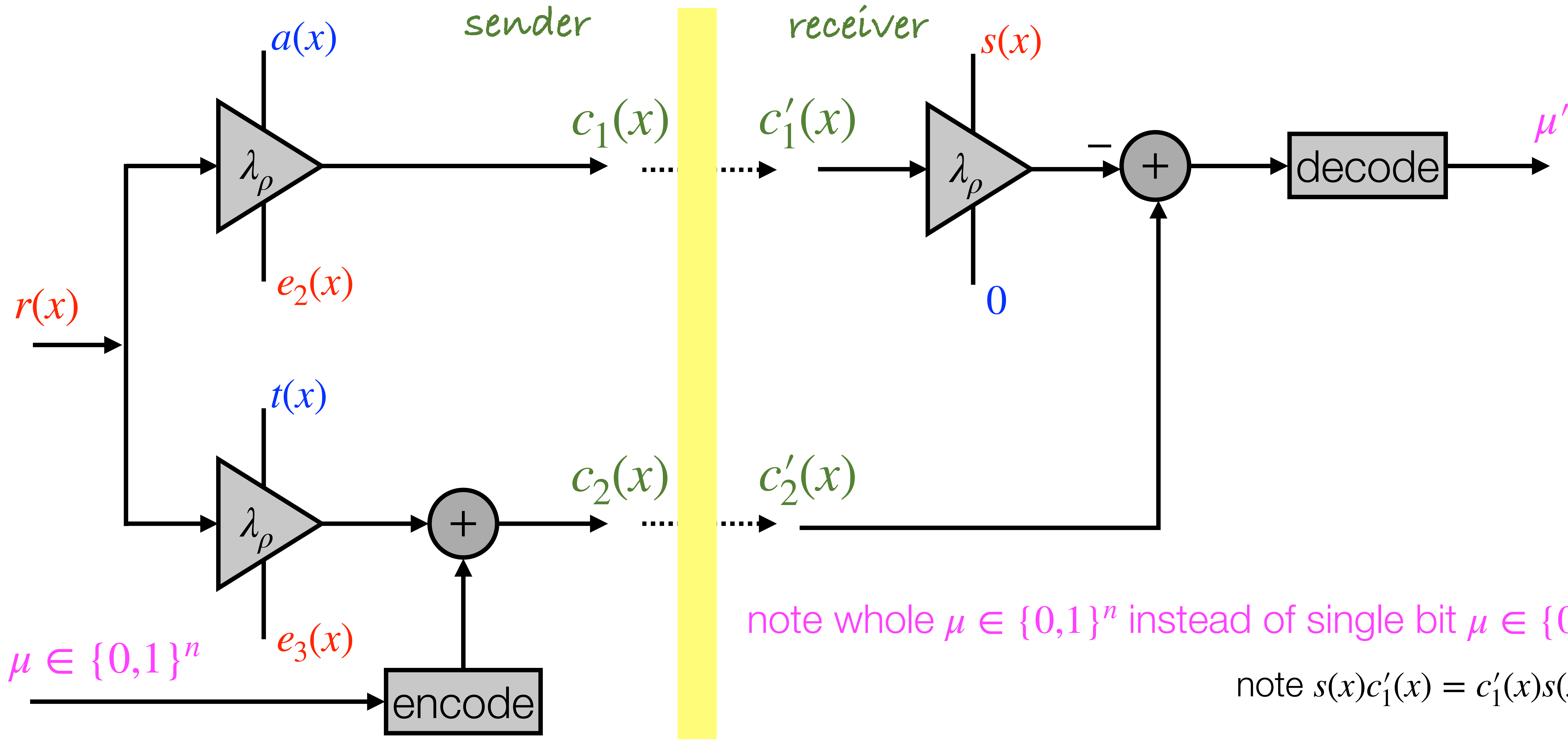
sk: $s(x) \leftarrow [\beta_1]$

pk: $\mathbf{A} \leftarrow R_q \left(= \mathbb{Z}_q[x] / \langle x^n + 1 \rangle \right)$

pk: $t(x) = a(x) \times s(x) + e_1(x)$

$p(x) \leftarrow S$ means that $p(x)$ coefficients are all chosen uniformly at random from the set S

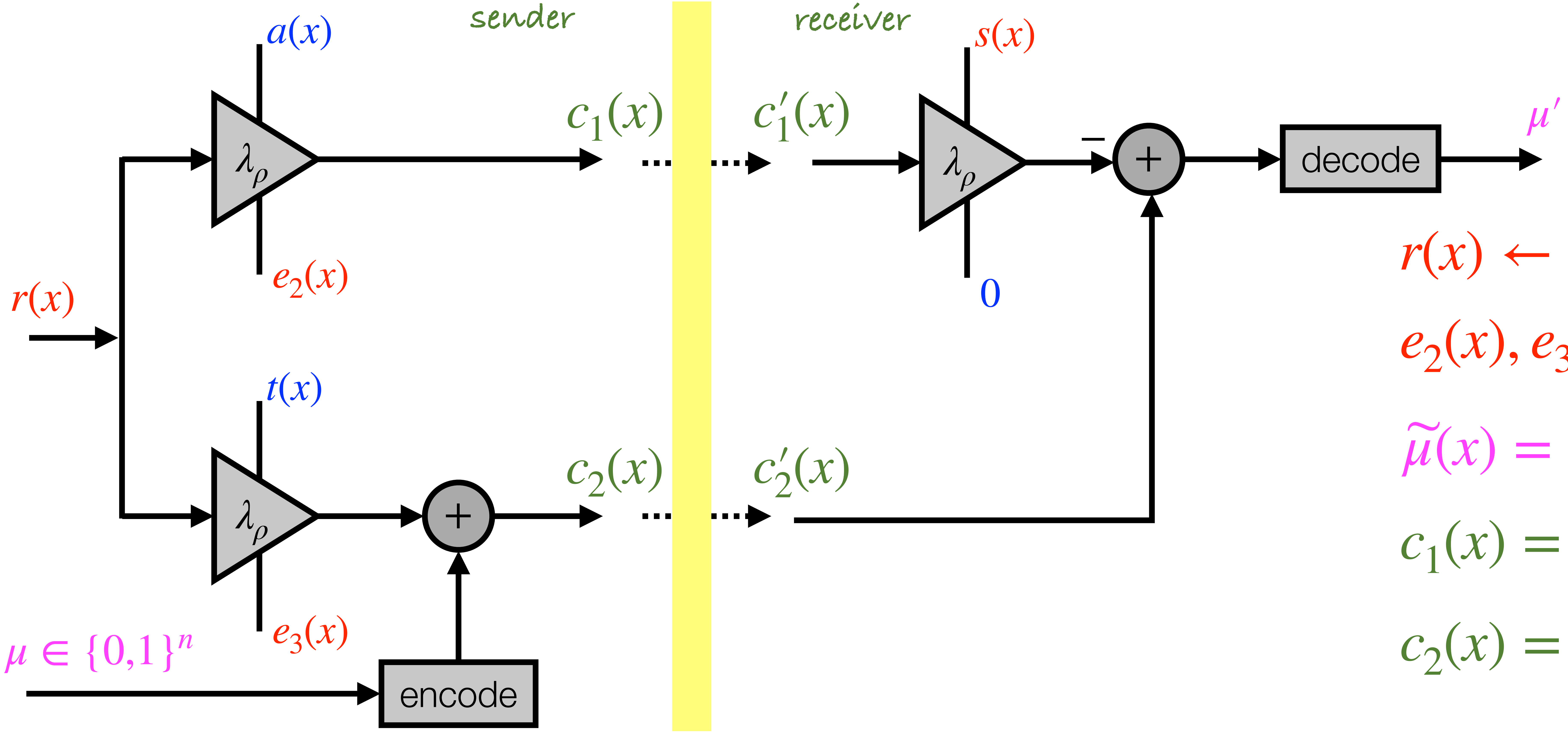
Ring-LWE Encryption Scheme
encryption/decryption of *n*-bit messages



note whole $\mu \in \{0,1\}^n$ instead of single bit $\mu \in \{0,1\}$

note $s(x)c_1'(x) = c_1'(x)s(x)$

Ring-LWE Encryption Scheme
encryption/decryption of *n*-bit messages



$$r(x) \leftarrow [\beta_1]$$

$$e_2(x), e_3(x) \leftarrow [\beta_2]$$

$$\widetilde{\mu}(x) = encode(\mu)$$

$$c_1(x) = a(x)r(x) + e_2(x)$$

$$c_2(x) = t(x)r(x) + e_3(x) + \widetilde{\mu}(x)$$

$$\begin{aligned} c_2(x) - s(x)c_1(x) &= t(x)r(x) + e_3(x) + \widetilde{\mu}(x) - t(x)r(x) + e_1(x)r(x) - s(x)e_2(x) \\ &= e_1(x)r(x) - s(x)e_2(x) + e_3(x) + \widetilde{\mu}(x) \end{aligned}$$

note $s(x)a(x) = t(x) - e_1(x)$

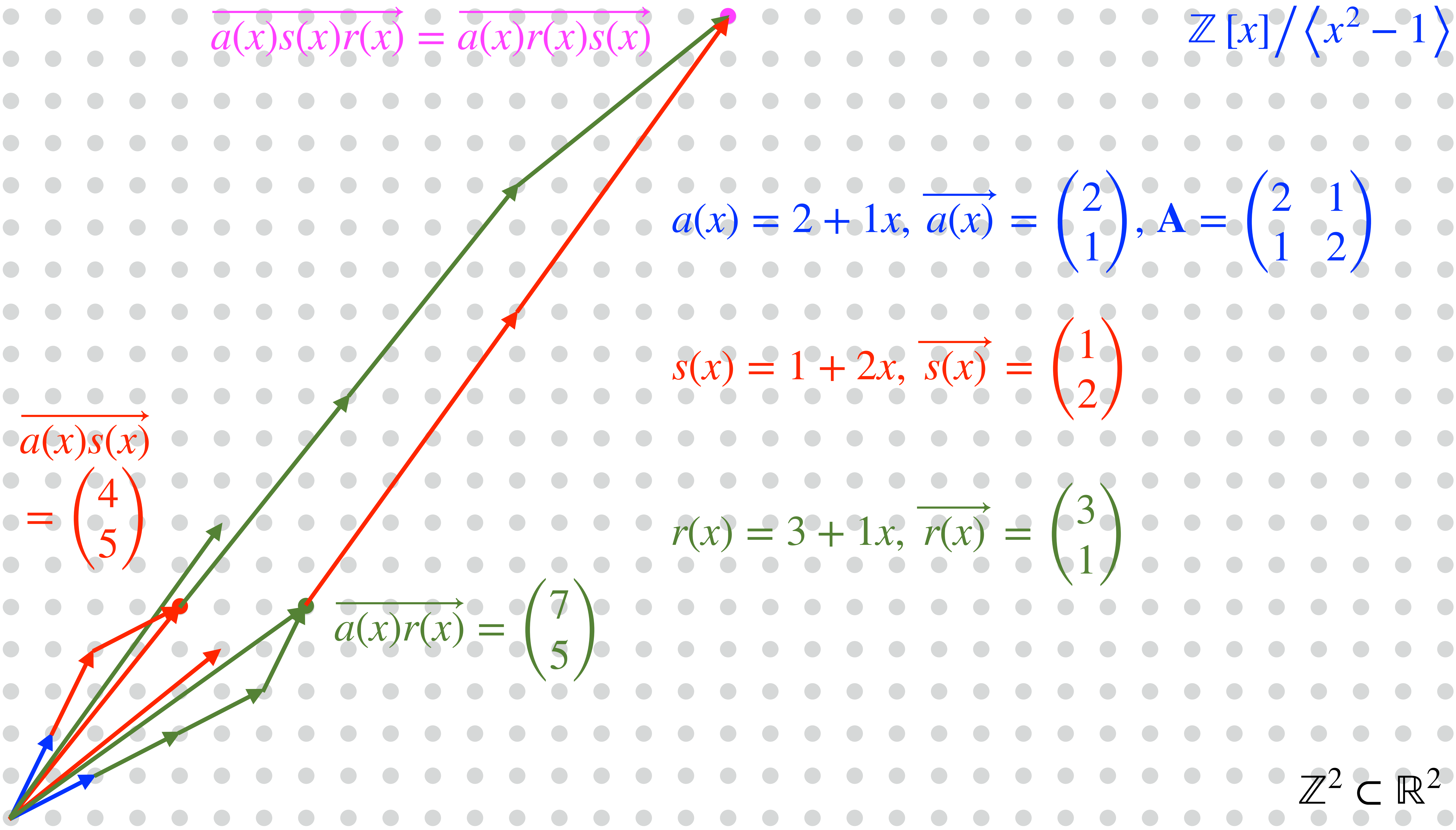
Linear Algebra Viewpoint

Let $a(x), b(x) \in \mathbb{Z}[x] \big/ \langle f(x) \rangle$ and fix $a(x)$, then:

$$a(x)b(x) = a(x) \sum_{i=0}^{d-1} b_i x^i \mod f(x) = \sum_{i=0}^{d-1} b_i \left(a(x)x^i \mod f(x) \right) .$$

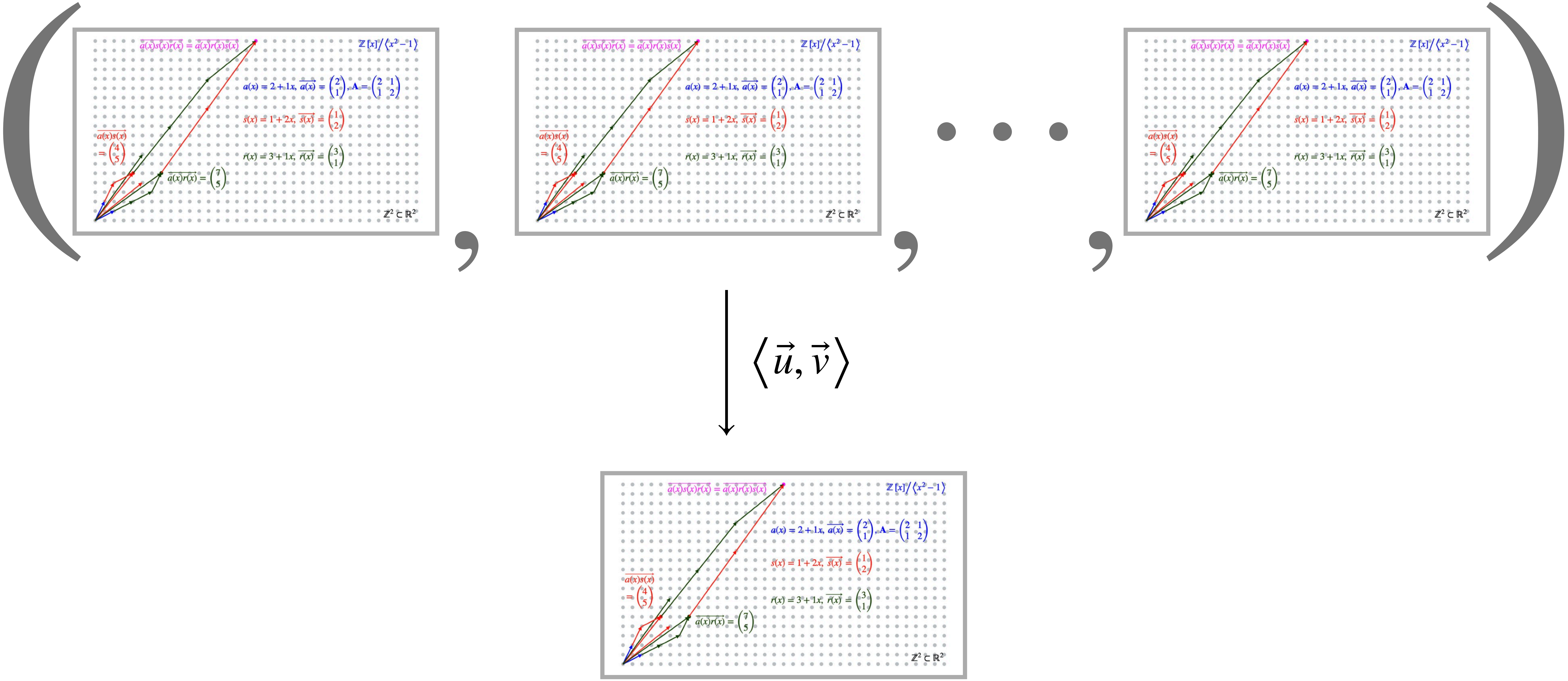
This can be interpreted as: $\overrightarrow{a(x)b(x)} = \mathbf{A}\overrightarrow{b(x)}$, for $\mathbf{A} \in \mathbb{Z}^{d \times d}$ with columns:

$$\mathbf{A} = \left(\overrightarrow{a(x)},\ \overrightarrow{a(x)x \mod f(x)},\ \ldots,\ \overrightarrow{a(x)x^{d-1} \mod f(x)} \right) .$$

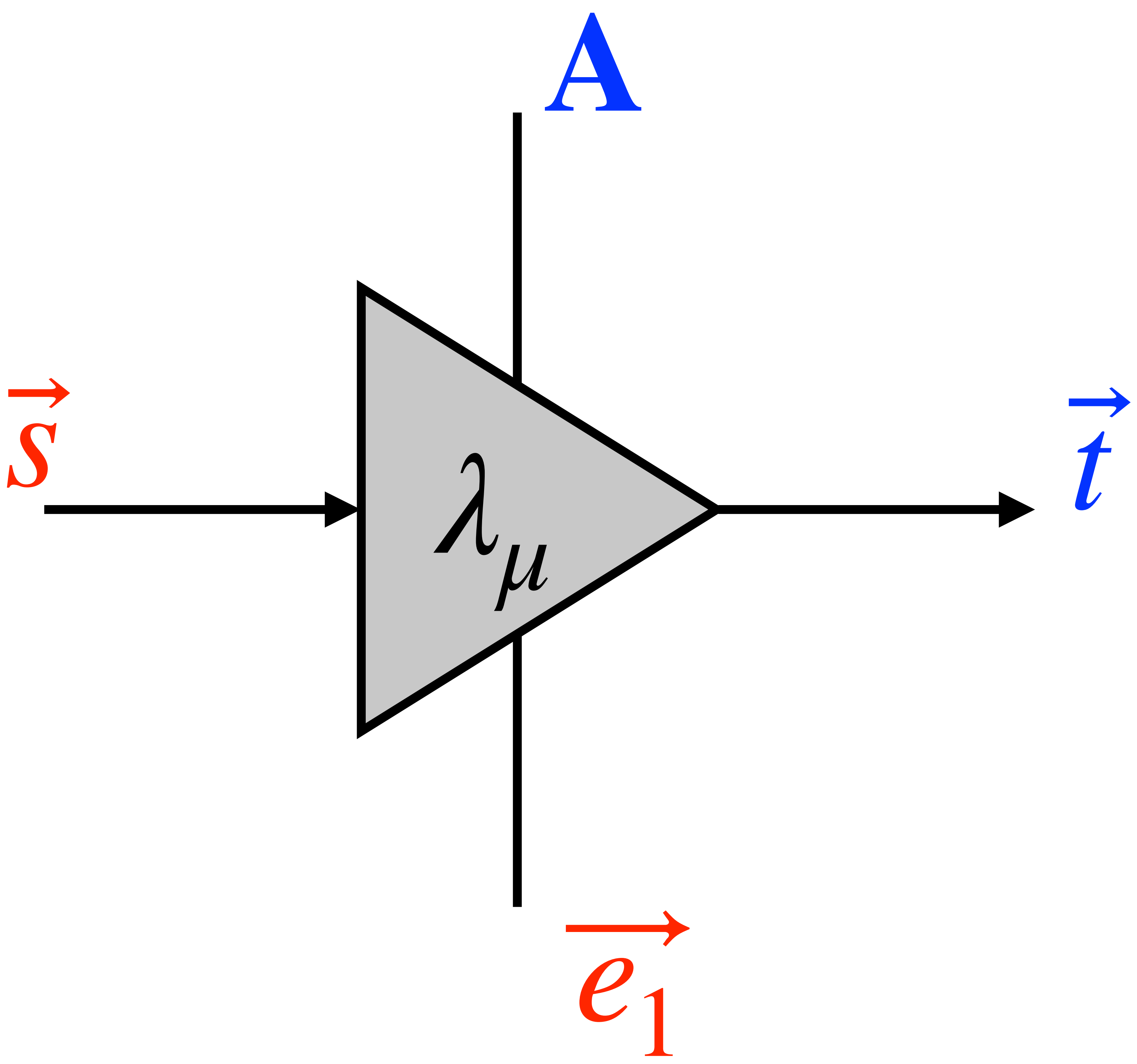


$\overrightarrow{a(x)s(x)r(x)} = \overrightarrow{a(x)r(x)s(x)}$
$\mathbb{Z}[x]/\langle x^2 - 1 \rangle$
$a(x) = 2 + 1x,\ \overrightarrow{a(x)} = \begin{pmatrix} 2 \\ 1 \end{pmatrix}, \mathbf{A} = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}$
$s(x) = 1 + 2x,\ \overrightarrow{s(x)} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$
$r(x) = 3 + 1x,\ \overrightarrow{r(x)} = \begin{pmatrix} 3 \\ 1 \end{pmatrix}$
$\overrightarrow{a(x)s(x)} = \begin{pmatrix} 4 \\ 5 \end{pmatrix}$
$\overrightarrow{a(x)r(x)} = \begin{pmatrix} 7 \\ 5 \end{pmatrix}$
$\mathbb{Z}^2 \subset \mathbb{R}^2$

R-Modules in MLWE: (pseudo) Linear Algebra Viewpoint



Module-LWE Encryption Scheme
setup phase



$\vec{e_1} \leftarrow [\beta_2]^m$

sk: $\vec{s} \leftarrow [\beta_1]^m$

pk: $\mathbf{A} \leftarrow R_q^{m \times m}, \ R_q = \mathbb{Z}_q[x] / \langle x^n + 1 \rangle$

pk: $\vec{t} = \mathbf{A}\vec{s} + \vec{e_1}$

we set $m = n$, for the general LWE gate

Example: R_q^k

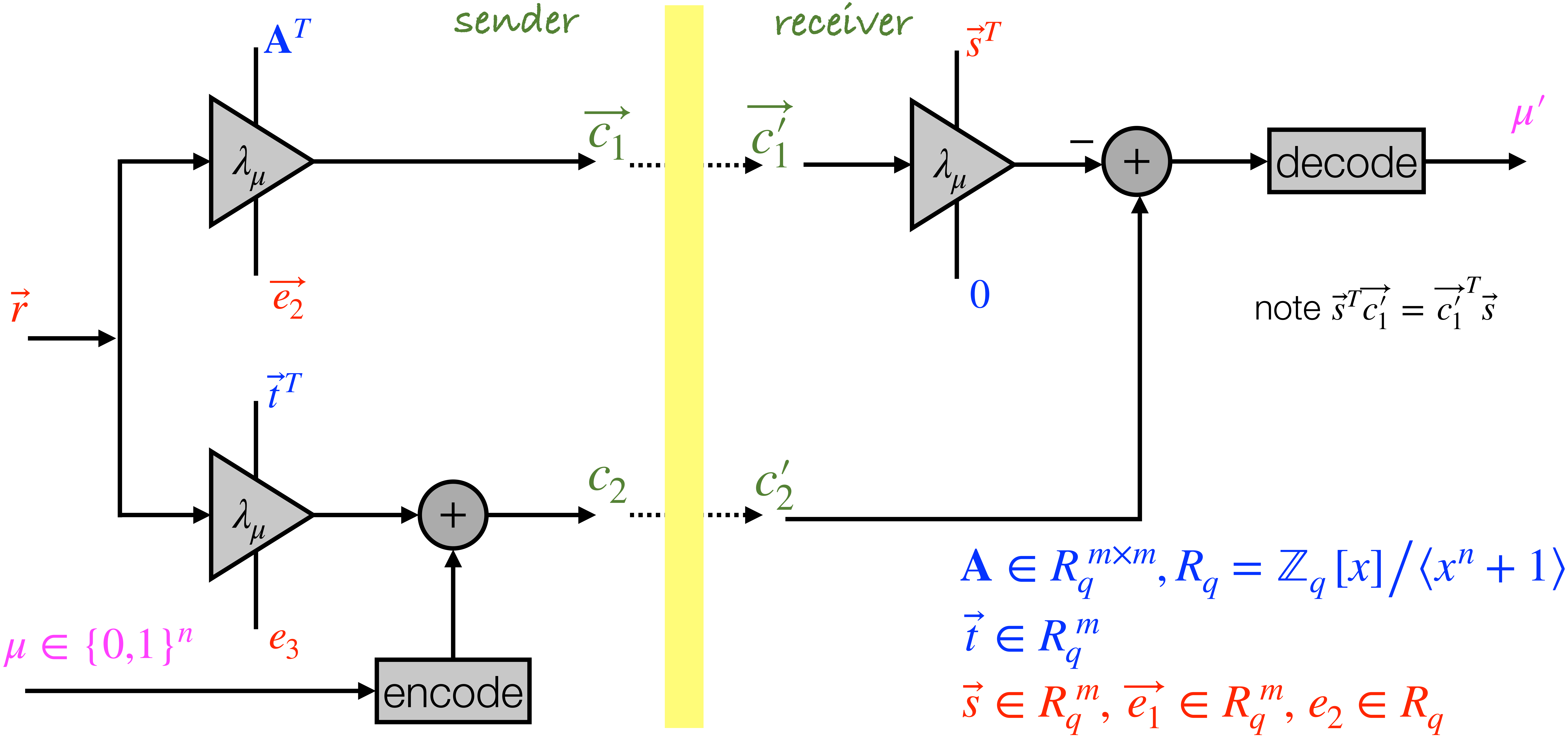
- Let $q = 137,\ n = 4,\ R_q = \mathbb{Z}_{137}[x]/(x^4+1),\ \ k = 3.$

- Let $a = \begin{bmatrix} 93 + 51x + 34x^2 + 54x^3 \\ 27 + 87x + 81x^2 + 6x^3 \\ 112 + 15x + 46x^2 + 122x^3 \end{bmatrix}$ and $b = \begin{bmatrix} 40 + 78x + x^2 + 119x^3 \\ 11 + 31x + 57x^2 + 90x^3 \\ 108 + 72x + 47x^2 + 14x^3 \end{bmatrix} \in R_q^k.$

- Then $a + b = \begin{bmatrix} 133 + 129x + 35x^2 + 36x^3 \\ 38 + 118x + x^2 + 96x^3 \\ 83 + 87x + 93x^2 + 136x^3 \end{bmatrix},\ a - b = \begin{bmatrix} 53 + 110x + 33x^2 + 72x^3 \\ 16 + 56x + 24x^2 + 53x^3 \\ 4 + 80x + 136x^2 + 108x^3 \end{bmatrix},$

  and $a \cdot b^T = a[1]b[1] + a[2]b[2] + a[3]b[3] = 93 + 59x + 44x^2 + 132x^3.$

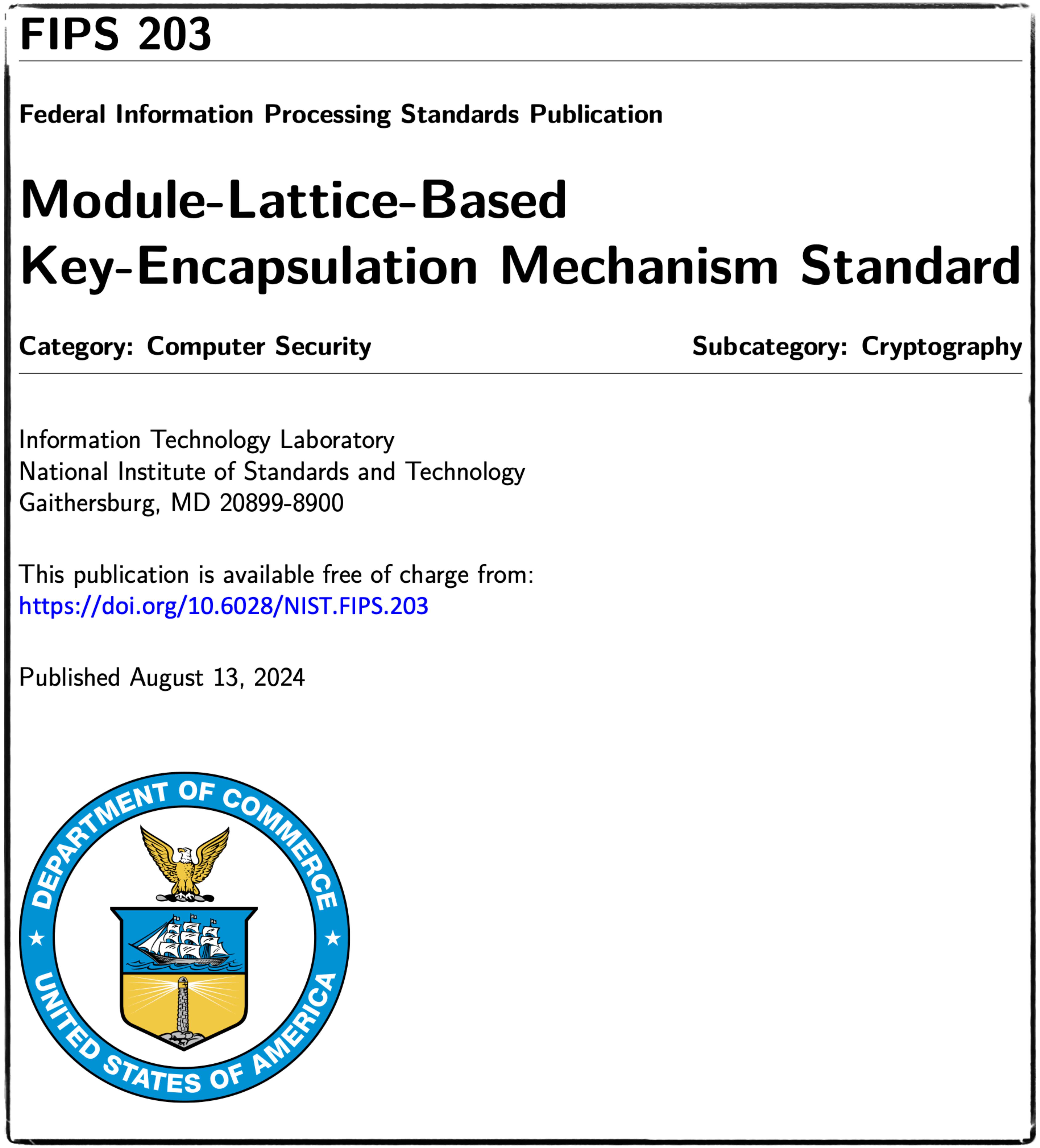
Module-LWE Encryption Scheme
encryption/decryption of *n*-bit messages



$\mathbf{A} \in R_q^{m \times m}, R_q = \mathbb{Z}_q[x] / \langle x^n + 1 \rangle$

$\vec{t} \in R_q^m$

$\vec{s} \in R_q^m, \vec{e_1} \in R_q^m, e_2 \in R_q$



FIPS 203

Federal Information Processing Standards Publication

Module-Lattice-Based Key-Encapsulation Mechanism Standard

Category: Computer Security **Subcategory: Cryptography**

Information Technology Laboratory
National Institute of Standards and Technology
Gaithersburg, MD 20899-8900

This publication is available free of charge from:
https://doi.org/10.6028/NIST.FIPS.203

Published August 13, 2024

- Fujisaki-Okamato extension to convert IND-CPA scheme to CCA2 secure one
- Number Theoretic Transform for faster ring operations
- Mandatory and recommended security checks
- Key and ciphertext data length optimizations
- Precise definition of the three parametric ML-KEM schemes based on M-LWE
    - *__M__odule __L__attice refers to lattices corresponding to certain R-modules*

[https://doi.org/10.6028/NIST.FIPS.203]

Algorithm 17 ML-KEM.Encaps_internal(ek, m)

Uses the encapsulation key and randomness to generate a key and an associated ciphertext.

Input: encapsulation key ek $\in \mathbb{B}^{384k+32}$.
Input: randomness $m \in \mathbb{B}^{32}$.
Output: shared secret key $K \in \mathbb{B}^{32}$.
Output: ciphertext $c \in \mathbb{B}^{32(d_u k+d_v)}$.

1: $(K, r) \leftarrow$ G($m \| $H(ek)) ▷ derive shared secret key K and randomness r
2: $c \leftarrow$ K-PKE.Encrypt(ek, m, r) ▷ encrypt m using K-PKE with randomness r
3: **return** (K, c)

Algorithm 18 ML-KEM.Decaps_internal(dk, c)

Uses the decapsulation key to produce a shared secret key from a ciphertext.

Input: decapsulation key dk $\in \mathbb{B}^{768k+96}$.
Input: ciphertext $c \in \mathbb{B}^{32(d_u k + d_v)}$.
Output: shared secret key $K \in \mathbb{B}^{32}$.

```
dk_PKE ← dk[0 : 384k]                    ▷ extract (from KEM decaps key) the PKE decryption key
ek_PKE ← dk[384k : 768k + 32]            ▷ extract PKE encryption key
h ← dk[768k + 32 : 768k + 64]            ▷ extract hash of PKE encryption key
z ← dk[768k + 64 : 768k + 96]            ▷ extract implicit rejection value
m′ ← K-PKE.Decrypt(dk_PKE, c)            ▷ decrypt ciphertext
(K′, r′) ← G(m′‖h)
K̄ ← J(z‖c)
c′ ← K-PKE.Encrypt(ek_PKE, m′, r′)       ▷ re-encrypt using the derived randomness r′
if c ≠ c′ then
    K′ ← K̄                               ▷ if ciphertexts do not match, “implicitly reject”
end if
return K′
```

Table 2. Approved parameter sets for ML-KEM

	n	q	k	η_1	η_2	d_u	d_v	required RBG strength (bits)
ML-KEM-512	256	3329	2	3	2	10	4	128
ML-KEM-768	256	3329	3	2	2	10	4	192
ML-KEM-1024	256	3329	4	2	2	11	5	256

Table 3. Sizes (in bytes) of keys and ciphertexts of ML-KEM

	encapsulation key	decapsulation key	ciphertext	shared secret key
ML-KEM-512	800	1632	768	32
ML-KEM-768	1184	2400	1088	32
ML-KEM-1024	1568	3168	1568	32

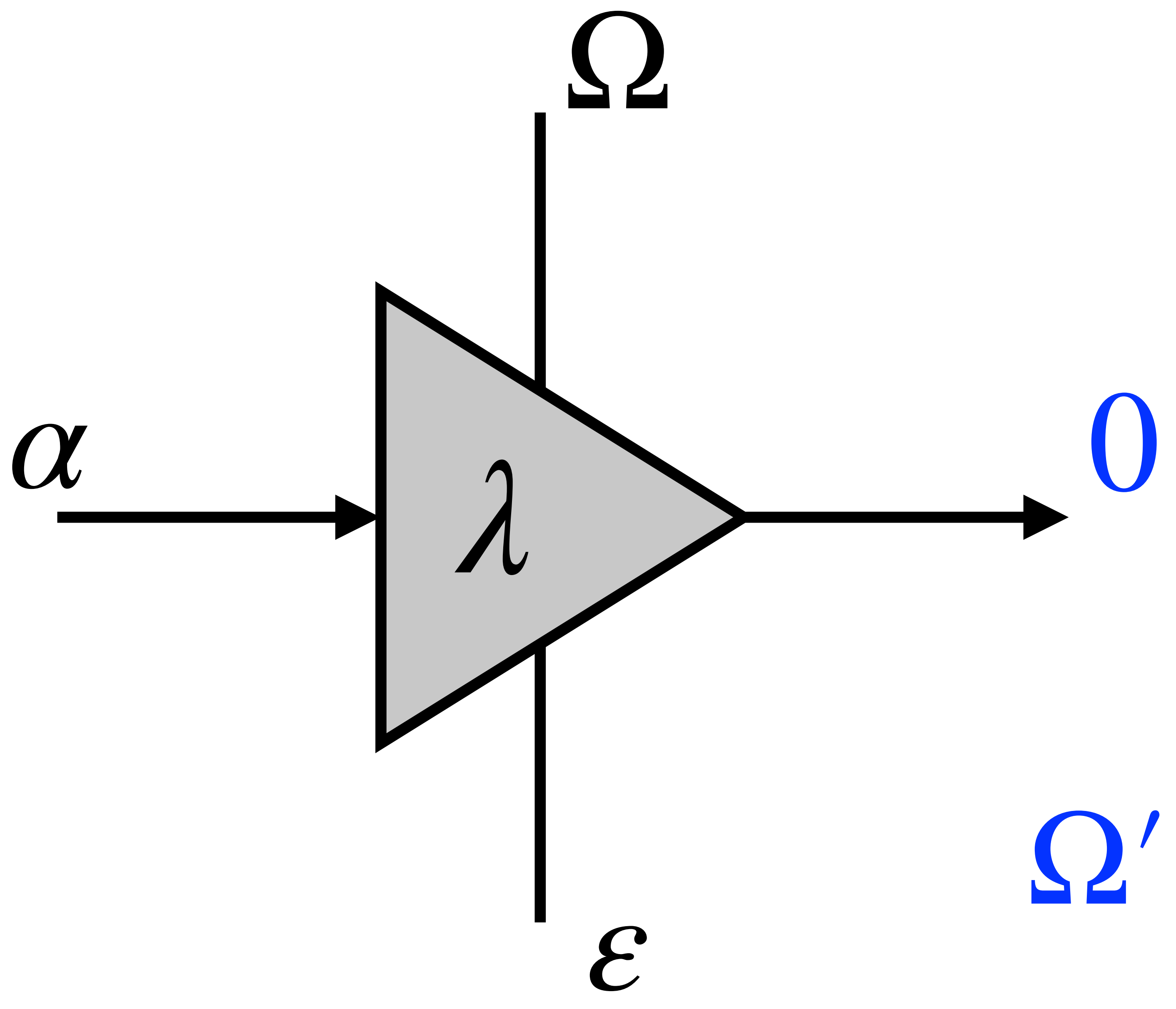
Short Integer Solution (SIS)
- standard, search version

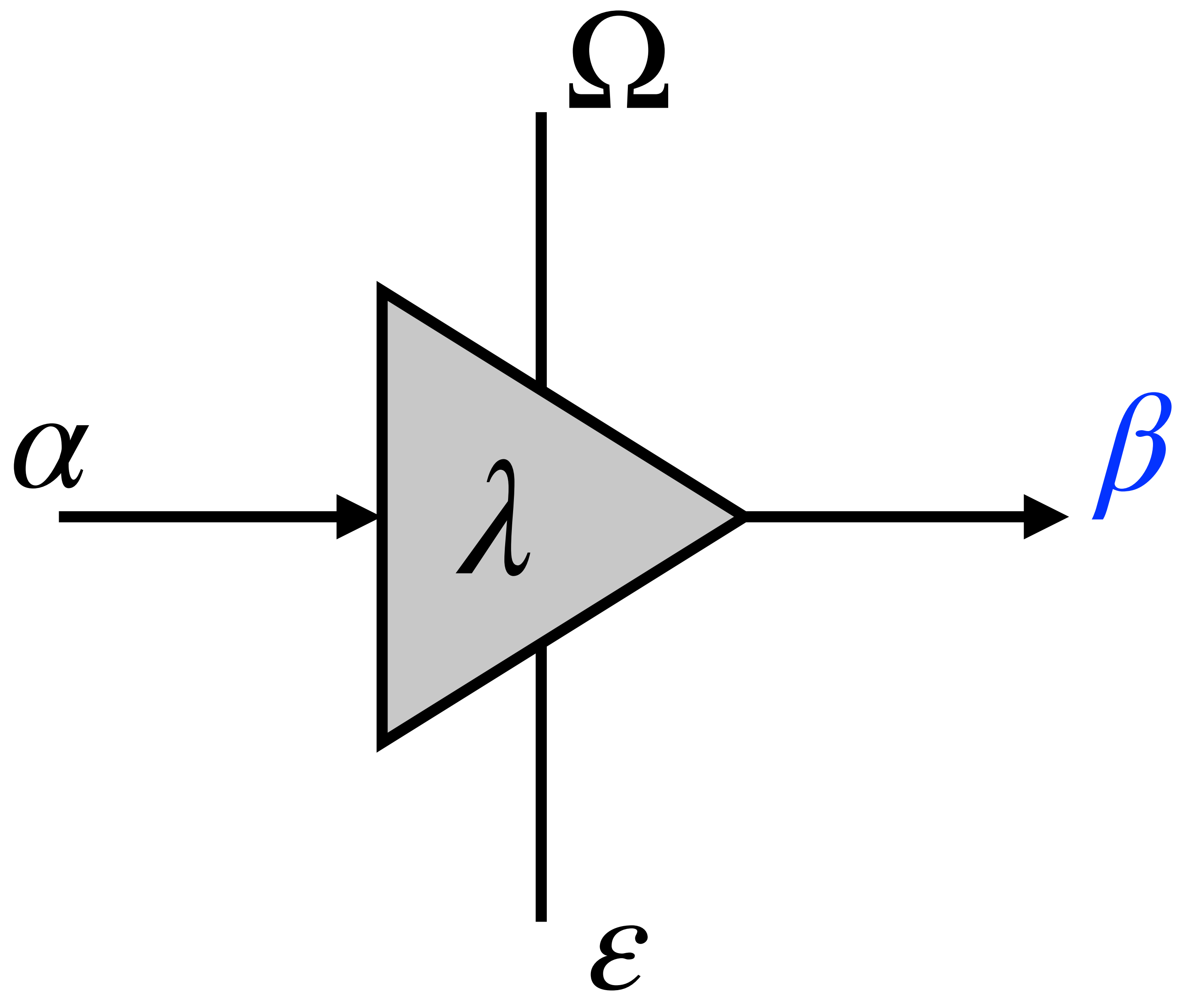
Definition 4.1.1 (Short Integer Solution ($\mathrm{SIS}_{n,q,\beta,m}$)). Given m uniformly random vectors $\mathbf{a}_i \in \mathbb{Z}_q^n$, forming the columns of a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, find a nonzero integer vector $\mathbf{z} \in \mathbb{Z}^m$ of norm $\|\mathbf{z}\| \leq \beta$ such that

$$f_{\mathbf{A}}(\mathbf{z}) := \mathbf{A}\mathbf{z} = \sum_i \mathbf{a}_i \cdot z_i = \mathbf{0} \in \mathbb{Z}_q^n. \tag{4.1.1}$$

[Peikert, https://ia.cr/2015/939]

LWE Gate from the SIS Viewpoint





$$\Omega' = [\Omega \parallel \mathbf{I}]$$

$$\Omega' \times \begin{bmatrix} \alpha \\ \varepsilon \end{bmatrix} = 0$$

homogeneous case

$$\Omega' \times \begin{bmatrix} \alpha \\ \varepsilon \end{bmatrix} = \beta \neq 0$$

inhomogeneous case

The homogeneous and inhomogeneous problems are essentially equivalent for typical params.

[Peikert, https://ia.cr/2015/939]

LWE or SIS - Heuristic Arguments

- Are we searching for **the particular solution** that we know it exists and that was used to setup the problem by opponent? *The noisy vector is primarily just an obstacle.*
  - we view the solution as a short **coordinate vector** for a lattice
  - we apply **Bounded-Distance-Decoding** to find the solution

LWE

- Or, are we searching for **"something like this" instead**, without any a priori hint anything like this was used to setup the problem by opponent? *The noisy vector is a natural part of the solution.*
  - we view the solution as a certain short **lattice vector directly**
  - we apply a sort of a **Short-Vector-Problem** to find the solution

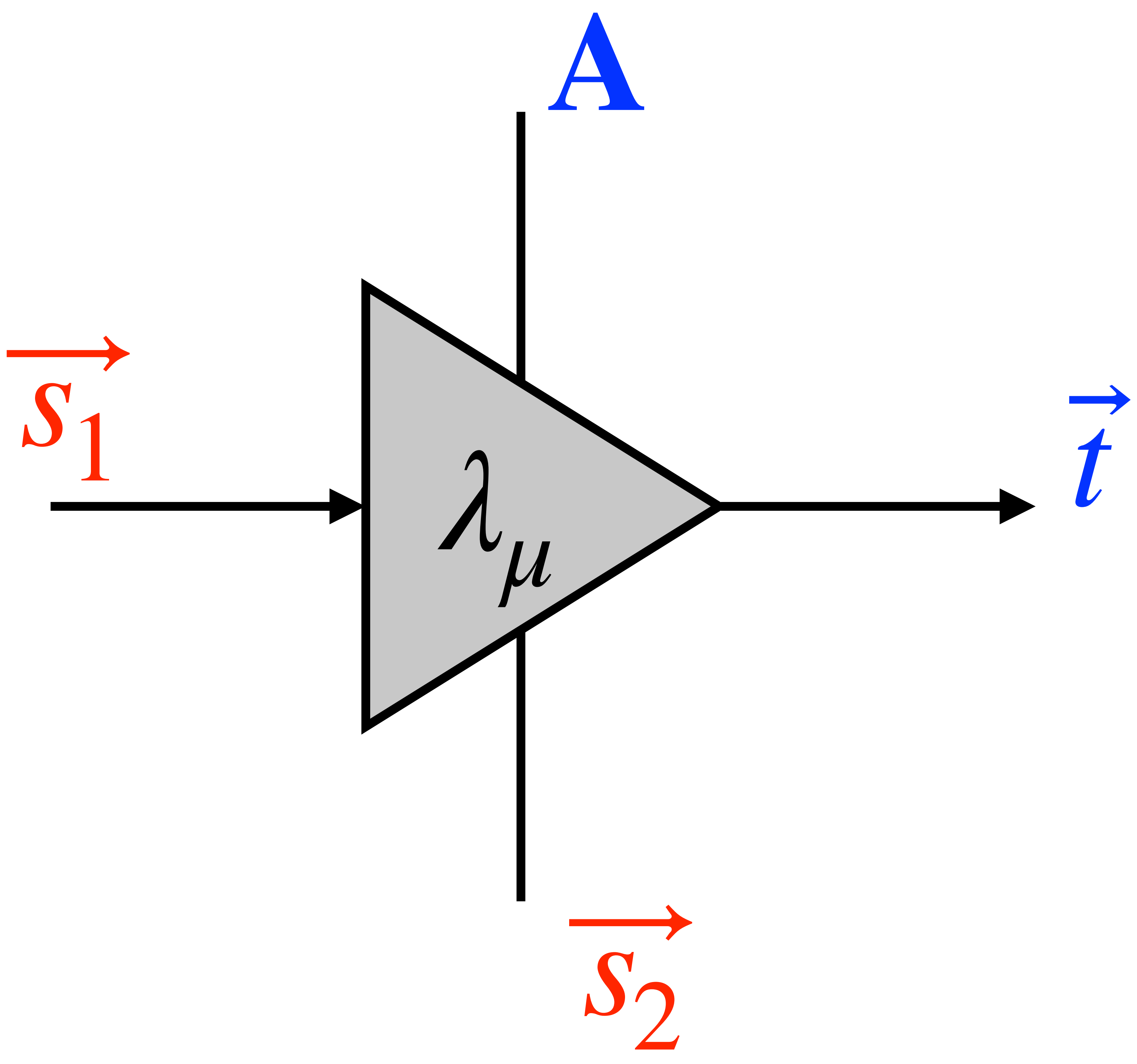
SIS

Up to a scaling factor, the lattices mentioned for LWE and SIS are duals of each other.

[Peikert, https://ia.cr/2015/939]

Module-LWE/SIS Signature Scheme

setup phase



sk: $\vec{s_1} \leftarrow [\beta_1]^l,\ \vec{s_2} \leftarrow [\beta_1]^k$

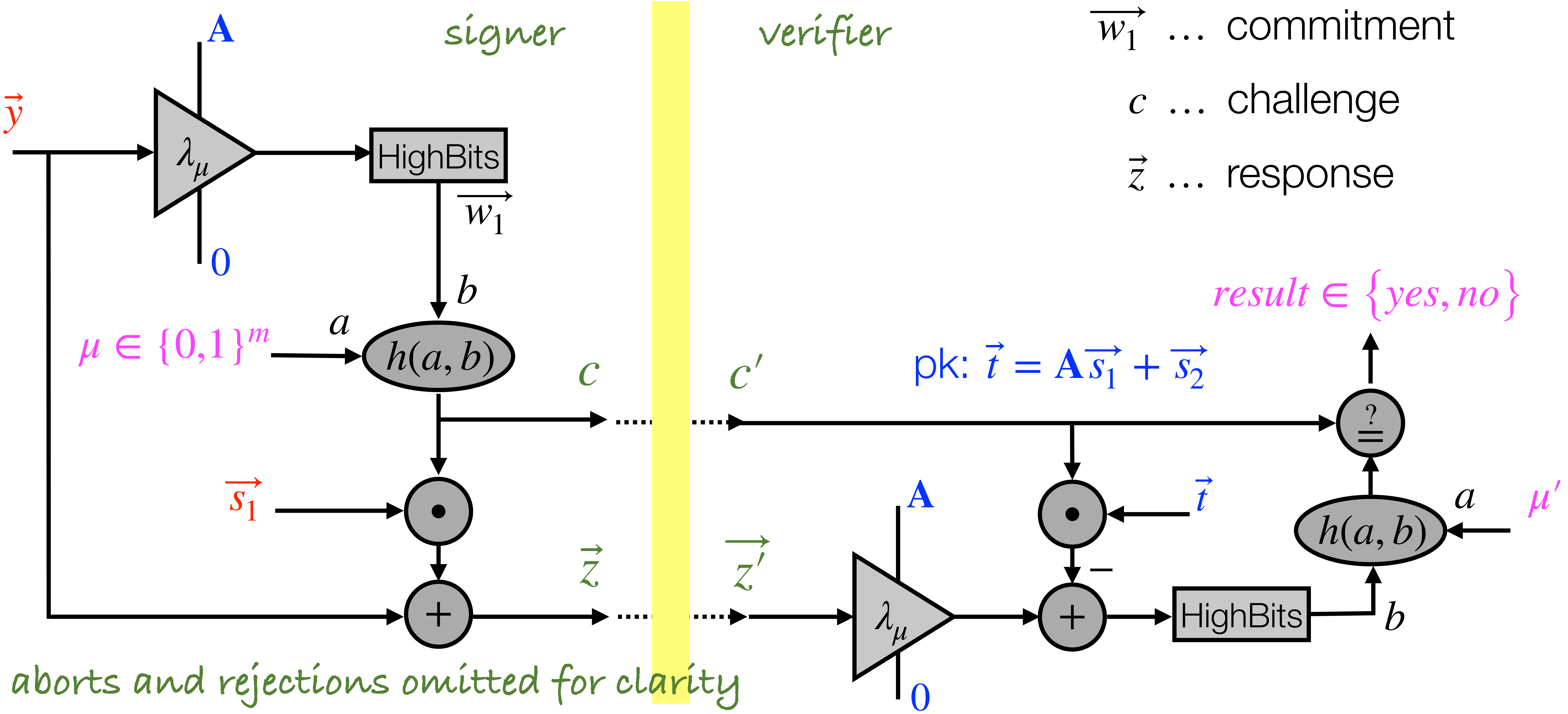
pk: $\mathbf{A} \leftarrow R_q^{k\times l},\ R_q = \mathbb{Z}_q[x]/\langle x^n + 1\rangle$

pk: $\vec{t} = \mathbf{A}\vec{s_1} + \vec{s_2}$

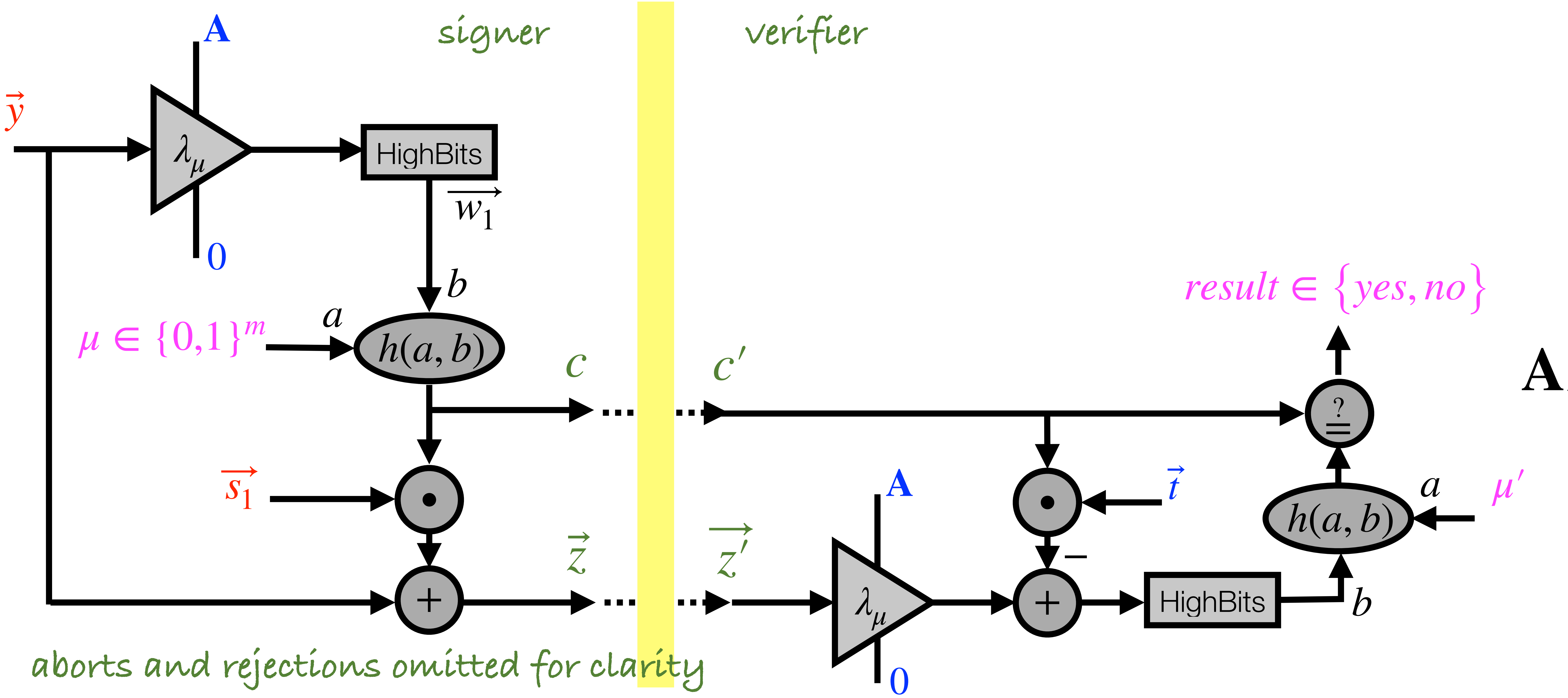
the noise vector $\vec{s_2}$ is a part of the secret private key; it governs Aborts in Fiat-Shamir later on

Module-LWE/SIS Schnorr-Fiat-Shamir Signature Scheme

signature generation/verification



Module-LWE/SIS Schnorr-Fiat-Shamir Signature Scheme signature generation/verification



$$\overrightarrow{w_1} = HighBits\left(\mathbf{A}\vec{z} - c\vec{t}\right)$$

$$c \stackrel{?}{=} h\left(\mu, \overrightarrow{w_1}\right)$$

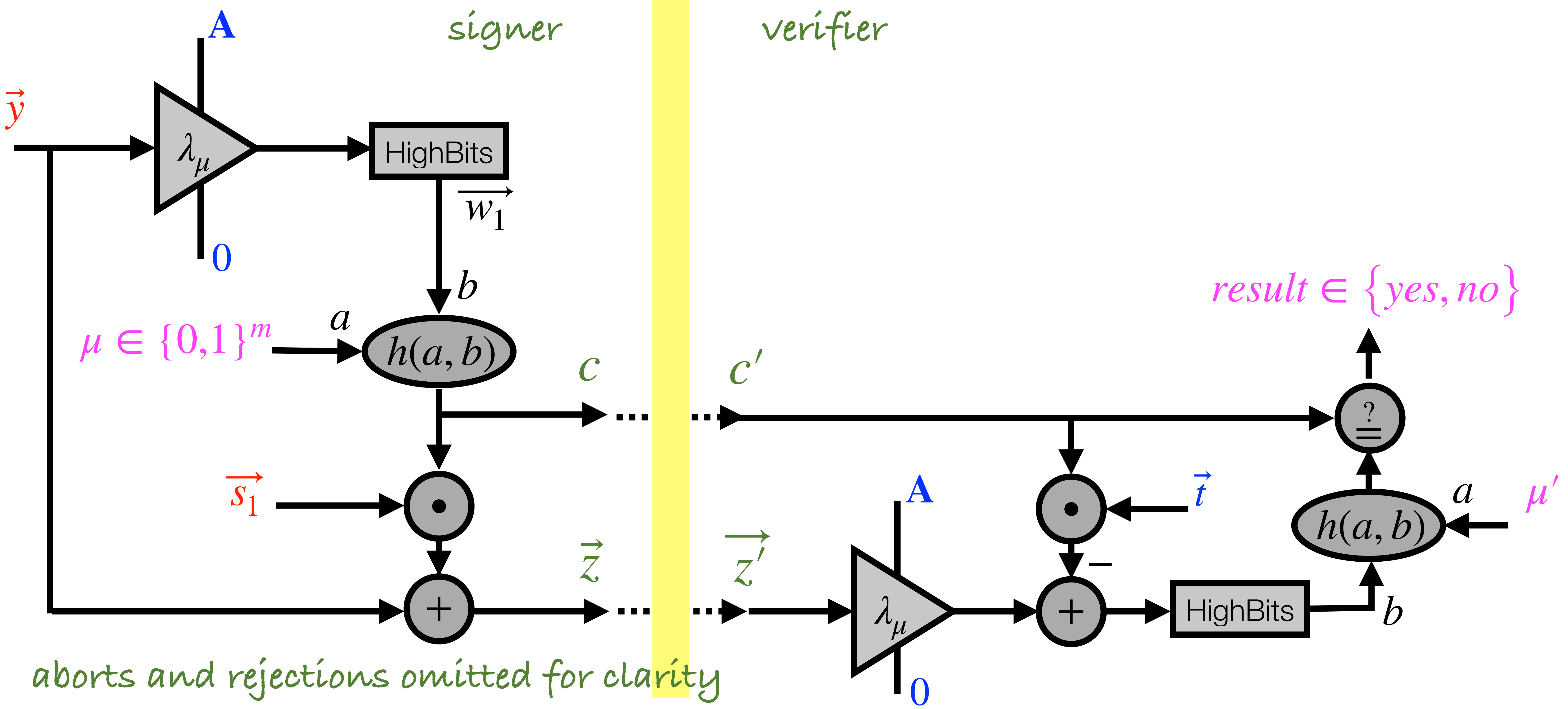
$$\mathbf{A}\vec{z} - c\vec{t} = \mathbf{A}\vec{y} + c\vec{t} - c\overrightarrow{s_2} - c\vec{t}$$
$$= \mathbf{A}\vec{y} - c\overrightarrow{s_2}$$

note $\mathbf{A}\overrightarrow{s_1} = \vec{t} - \overrightarrow{s_2}^T$

pick a small random $\vec{y}$ and compute $\overrightarrow{w_1} = HighBits\left(\mathbf{A}\vec{y}\right)$

$c = h\left(\mu, \overrightarrow{w_1}\right), \vec{z} = \vec{y} + c\overrightarrow{s_1}$

Module-LWE/SIS Schnorr-Fiat-Shamir Signature Scheme forgery through Module-SIS



$$HighBits\left(\mathbf{A}\vec{z} - c\vec{t}\right) \stackrel{!}{=} \overrightarrow{w_1}$$

$$\mathbf{A}\vec{z} - c\vec{t} = 2\gamma_2\overrightarrow{w_1} + \overrightarrow{w_0}$$

$$\mathbf{A}\vec{z} - \overrightarrow{w_0} = c\vec{t} + 2\gamma_2\overrightarrow{w_1}$$

$$[\Omega \parallel \mathbf{I}] \times \begin{bmatrix} \vec{z} \\ \overrightarrow{w_0} \end{bmatrix} = c\vec{t} + 2\gamma_2\overrightarrow{w_1}$$

we pick $\overrightarrow{w_1}$ at random and compute $c = h\left(\mu, \overrightarrow{w_1}\right)$

the remaining task is to find a correct $\vec{z}$

MSIS

FIPS 204

Federal Information Processing Standards Publication

Module-Lattice-Based Digital Signature Standard

Category: Computer Security **Subcategory: Cryptography**

Information Technology Laboratory
National Institute of Standards and Technology
Gaithersburg, MD 20899-8900

This publication is available free of charge from:
https://doi.org/10.6028/NIST.FIPS.204

Published August 13, 2024

- Fiat-Shamir with Aborts extension
- Rejection sampling to minimize private key leakage - transcript attack
- Number Theoretic Transform for faster ring operations
- Key and signature data length optimizations
- Precise definition of the three parametric ML-DSA schemes based on M-LWE and M-SIS
    - *__M__odule __L__attice refers to lattices corresponding to certain R-modules*

[https://doi.org/10.6028/NIST.FIPS.204]

Table 2. Sizes (in bytes) of keys and signatures of ML-DSA

	Private Key	Public Key	Signature Size
ML-DSA-44	2560	1312	2420
ML-DSA-65	4032	1952	3309
ML-DSA-87	4896	2592	4627

ML-DSA parameter sets

	Values assigned by each parameter set		
(see Sections 6.1 and 6.2 of this document)	ML-DSA-44	ML-DSA-65	ML-DSA-87
q - modulus [see §6.1]	8380417	8380417	8380417
ζ - a 512th root of unity in $\mathbb{Z}_q$ [see §7.5]	1753	1753	1753
d - # of dropped bits from $\mathbf{t}$ [see §6.1]	13	13	13
τ - # of ± 1's in polynomial c [see §6.2]	39	49	60
λ - collision strength of $\tilde{c}$ [see §6.2]	128	192	256
γ_1 - coefficient range of $\mathbf{y}$ [see §6.2]	2^{17}	2^{19}	2^{19}
γ_2 - low-order rounding range [see §6.2]	$(q-1)/88$	$(q-1)/32$	$(q-1)/32$
(k,ℓ) - dimensions of $\mathbf{A}$ [see §6.1]	(4,4)	(6,5)	(8,7)
η - private key range [see §6.1]	2	4	2
$\beta = \tau \cdot \eta$ [see §6.2]	78	196	120
ω - max # of 1's in the hint $\mathbf{h}$ [see §6.2]	80	55	75
Challenge entropy $\log_2 \binom{256}{\tau} + \tau$ [see §6.2]	192	225	257
Repetitions (see explanation below)	4.25	5.1	3.85
Claimed security strength	Category 2	Category 3	Category 5

SLH-DSA by NIST FIPS 205 for Comparison

Table 2. SLH-DSA parameter sets

private key size = 2 x public key size

	n	h	d	h'	a	k	lg_w	m	security category	pk bytes	sig bytes
SLH-DSA-SHA2-128s SLH-DSA-SHAKE-128s	16	63	7	9	12	14	4	30	1	32	7 856
SLH-DSA-SHA2-128f SLH-DSA-SHAKE-128f	16	66	22	3	6	33	4	34	1	32	17 088
SLH-DSA-SHA2-192s SLH-DSA-SHAKE-192s	24	63	7	9	14	17	4	39	3	48	16 224
SLH-DSA-SHA2-192f SLH-DSA-SHAKE-192f	24	66	22	3	8	33	4	42	3	48	35 664
SLH-DSA-SHA2-256s SLH-DSA-SHAKE-256s	32	64	8	8	14	22	4	47	5	64	29 792
SLH-DSA-SHA2-256f SLH-DSA-SHAKE-256f	32	68	17	4	9	35	4	49	5	64	49 856

Vulnerabilities we went through before and probably will go again

- Implementation faults, for instance:
  - faulty encryption/decryption
  - faulty signature generation/verification
- Computational faults
  - such as were RSA-CRT vulnerabilities
- Side channels
  - sensitive data leakage

Recent Example - EUCLEAK Attack on YubiKey Series 5

- FIDO2 and EAL5+ certified cryptographic device
- ECDSA implementation broken via EM side channel
- Possibly affects broader area of security microcontrollers by Infineon and **broader protocols area**
- The failure is in **radiating modular inversion procedure**
- There is a modular inversion in PACE-CAM(*) involving chip private key z_A

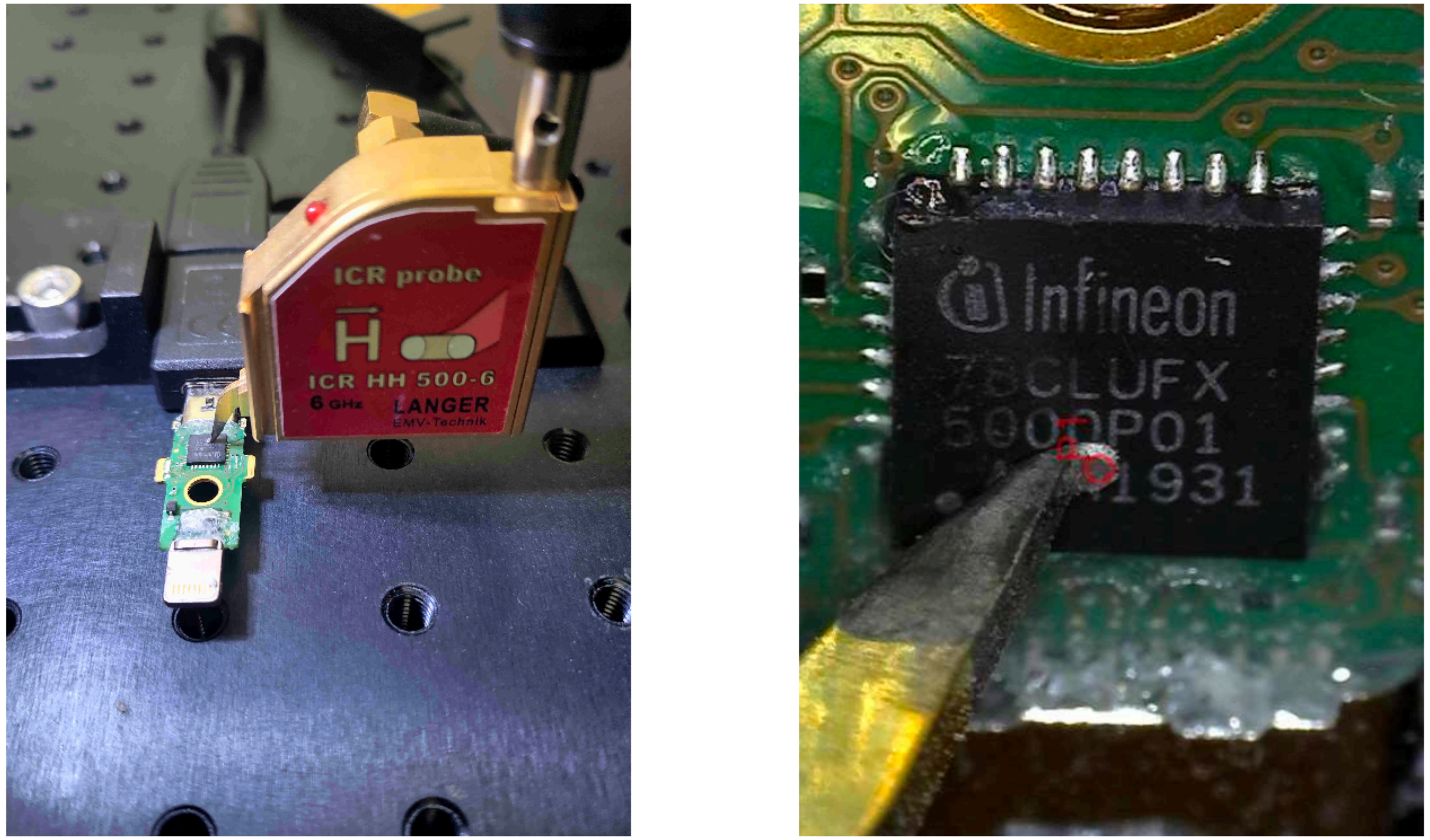


Figure 1.4: YubiKey 5Ci – EM Acquisition Setup

— https://ninjalab.io/eucleak/

*) PACE-CAM employed in NFC passports and ID cards (SK)

NTT - Number Theoretic Transform

- Specialized discrete Fourier transform to speed up multiplication in certain rings of convolution polynomials
- Can be also interpreted as a sort of Chinese Remainder Theorem machinery
- Is a vital core of LWE based algorithms ML-KEM and ML-DSA
- Is a fruitful target of **fault and side channel attacks**

$$R_q := \mathbb{Z}_q[X]/(X^{256}+1) \qquad T_q := \bigoplus_{i=0}^{127} \mathbb{Z}_q[X]/\left(X^2 - \zeta^{2\mathsf{BitRev}_7(i)+1}\right)$$

$$\hat{f} := \left(f \bmod (X^2 - \zeta^{2\mathsf{BitRev}_7(0)+1}), \ldots, f \bmod (X^2 - \zeta^{2\mathsf{BitRev}_7(127)+1})\right)$$

$$f \times_{R_q} g = \mathsf{NTT}^{-1}(\hat{f} \times_{T_q} \hat{g})$$

— NIST FIPS 203: ML-KEM, August 13th, 2024

Floating Point FFT in FALCON (FN-DSA)

- Automatic offloading of sensitive computation to a Floating Point Unit (FPU) naturally invokes side-channels that are uneasy to predict and prevent

4.1 Floating-Point

Signature generation, and also part of key pair generation, involve the use of complex numbers. These can be approximated with standard IEEE 754 floating-point numbers ("binary64" format, commonly known as "double precision"). Each such number is encoded over 64 bits, that split into the following elements:

- a sign $s = \pm 1$ (1 bit);
- an exponent e in the -1022 to $+1023$ range (11 bits);
- a mantissa m such that $1 \leq m < 2$ (52 bits).

In general, the represented value is $sm2^e$. The mantissa is encoded as $2^{52}(m-1)$; it has 53 bits of precision, but its top bit, of value 1 by definition, is omitted in the encoding.

[Falcon: Fast-Fourier Lattice-based Compact Signatures over NTRU, v1.2]

Thank you for your attention

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Cybersecurity Competence Centre. Neither the European Union nor the European Cybersecurity Competence Centre can be held responsible for them.

Funded by the European Union

ECCC

EUROPEAN CYBERSECURITY COMPETENCE CENTRE

History (year-month-day format)

- 2025-01-18, version 1 release
- 2024-12-12, version 0.9999 beta - better annotation towards adjoint operator
- 2024-11-14, version 0.999 beta - clarification note on adjoint operator
- 2024-11-14, version 0.99 beta - bunch of typos corrected, mainly captions
- 2024-11-13, version 0.9 beta - typos may occur(!)