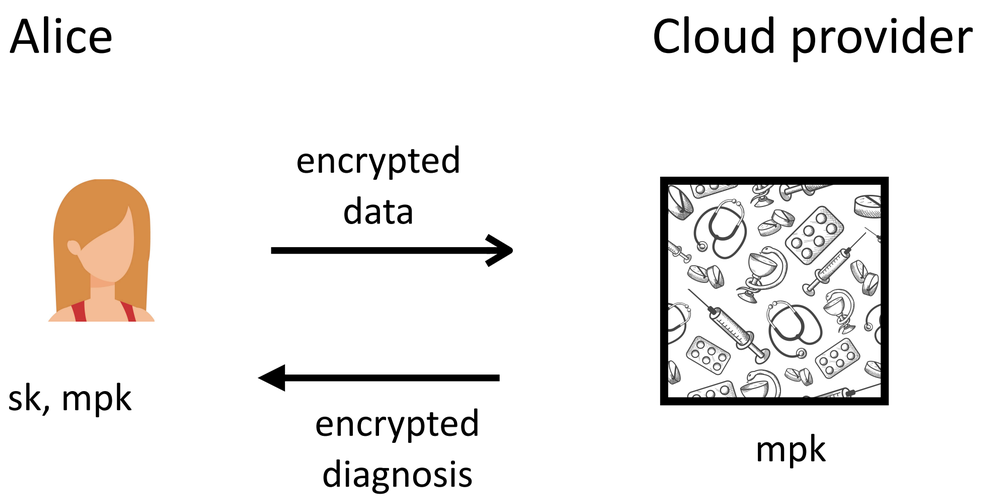
Plug-and-Play Sanitization for TFHE

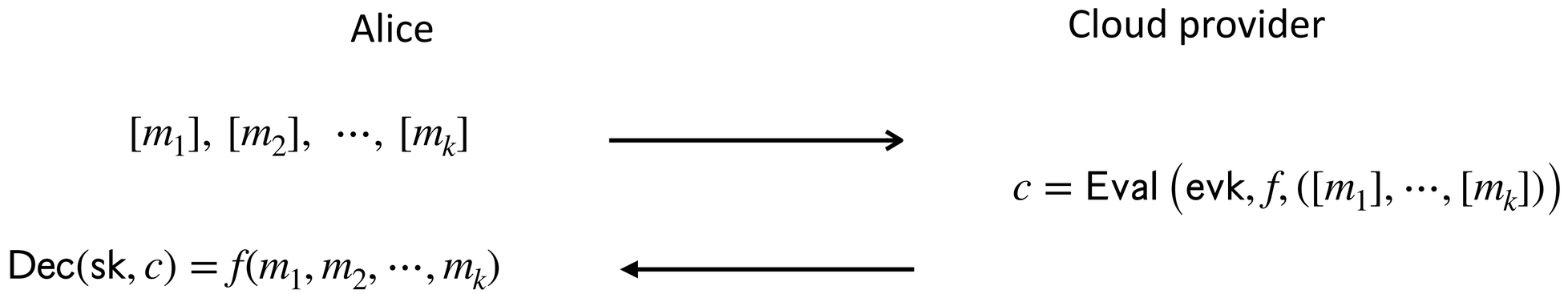
Malika Izabachène,
joint work with Florian Bourse

ECO Seminar
5, April 2024

Privacy preserving diagnosis

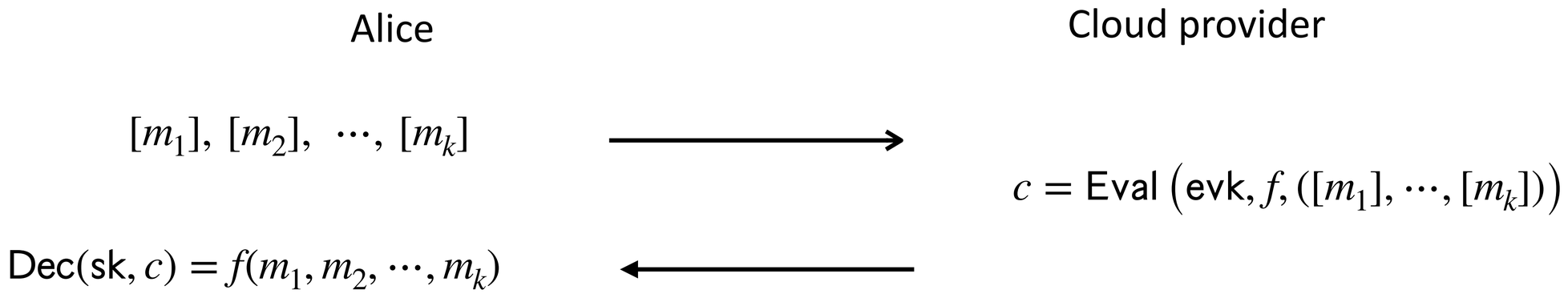


Fully Homomorphic Encryption (FHE)



1. Message privacy: Alice messages are kept unknown to the cloud provider;

Fully Homomorphic Encryption (FHE)



1. Message privacy: Alice messages are kept unknown to the cloud provider;
2. Circuit privacy: Eval reveals nothing about f, except $f(m_1, m_2, \cdots, m_k)$, even knowing sk.

Example: f is an algorithm from which a service provider makes profit.

Outline

- Definitions
- Circuit privacy and sanitization notions
- Previous approaches
- Randomization tools
- TFHE bootstrapping
- FHE sanitization algorithm
- Comparison with previous approach and performance

Fully Homomorphic Encryption (FHE)

- KeyGen(1^λ): evk, sk
- Enc(sk, m): c (also, written as $[m]$ in this talk)
- Dec(sk, c): m
- Eval(evk, $f, [m_1], \cdots, [m_k]$): c

Fully Homomorphic Encryption (FHE)

- KeyGen(1^λ): evk, sk

- Enc(sk, m): c (also, written as $[m]$ in this talk)

- Dec(sk, c): m

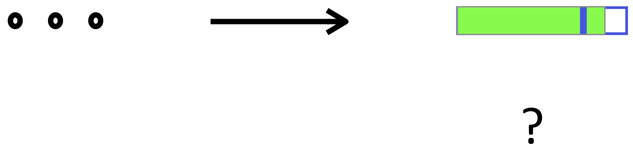
- Eval(evk, $f, [m_1], \cdots, [m_k]$): c

Correctness:

1) if $c = \text{Enc}(\text{sk}, m)$, $\text{Dec}(\text{sk}, c) = m$;
2) if $c = \text{Eval}(\text{evk}, f, [m_1], .. [m_k])$, $\text{Dec}(\text{sk}, c) = f(m_1, \cdots, m_k)$

FHE ciphertexts (noisy ciphertexts)

$$[m_0] + [m_1] \longrightarrow [m_0 \oplus m_1]$$



?

$$[m_0] \times [m_1] \longrightarrow [m_0 \wedge m_1]$$

Bootstrapping, [Gentry09]

[Gentry09] Craig Gentry. Fully homomorphic encryption using ideal lattices. STOC2009.

By definition: [m] = Encrypt(sk,m) , Decrypt(sk, [m]) = m

Define $\text{Decrypt}_{[m]}(\cdot) := \text{Decrypt}(\cdot, [m])$

Bootstrapping, [Gentry09]

[Gentry09] Craig Gentry. Fully homomorphic encryption using ideal lattices. STOC2009.

By definition: [m] = Encrypt(sk,m) , Decrypt(sk, [m]) = m

Define $\text{Decrypt}_{[m]}(\cdot) := \text{Decrypt}(\cdot , [m])$

Let [sk] = Encrypt(sk,sk)

Bootstrapping, [Gentry09]

[Gentry09] Craig Gentry. Fully homomorphic encryption using ideal lattices. STOC2009.

By definition: $[m] = \text{Encrypt}(sk, m)$, $\text{Decrypt}(sk, [m]) = m$

Define $\text{Decrypt}_{[m]}(\cdot) := \text{Decrypt}(\cdot , [m])$

Let $[sk] = \text{Encrypt}(sk, sk)$

$$\text{Decrypt}_{[m]}([sk]) = \text{Decrypt}([sk], [m])$$

Bootstrapping, [Gentry09]

[Gentry09] Craig Gentry. Fully homomorphic encryption using ideal lattices. STOC2009.

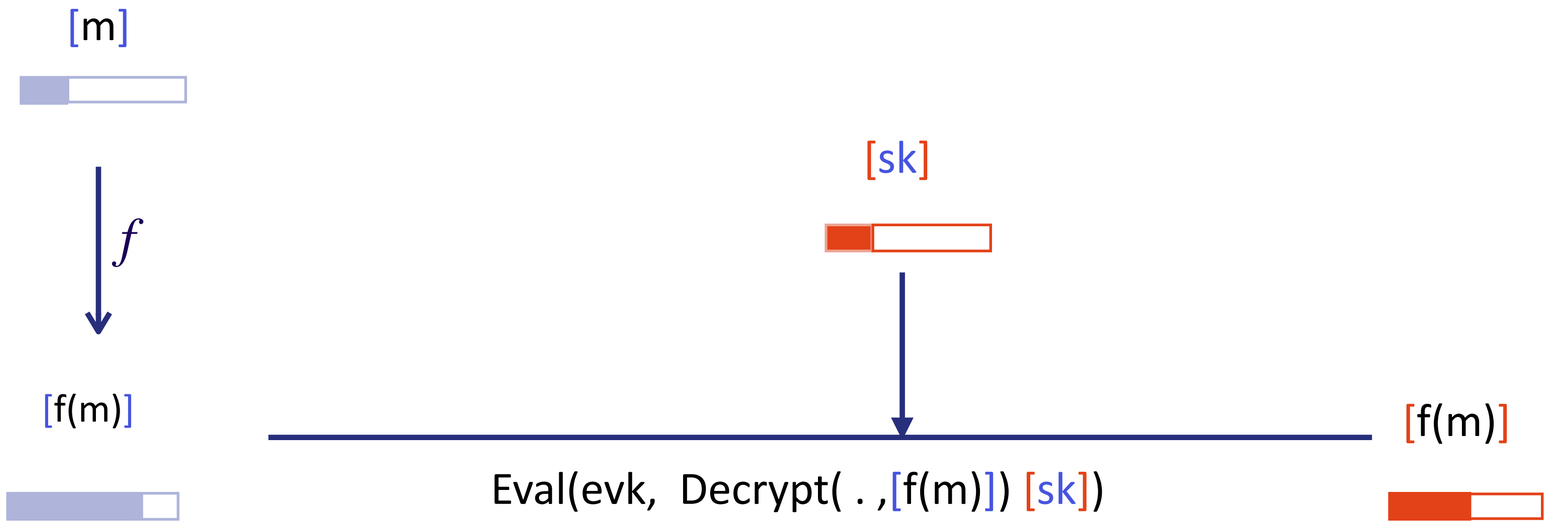
By definition: [m] = Encrypt(sk,m) , Decrypt(sk, [m]) = m

Define $\text{Decrypt}_{[m]}(\cdot) := \text{Decrypt}(\cdot , [m])$

Let [sk] = Encrypt(sk,sk)

$$\text{Decrypt}_{[m]}([sk]) = \text{Decrypt}([sk], [m]) = [m]$$

Bootstrapping noise growth, [Gentry09]



Circuit-privacy and Sanitization

Warm-up example: GSW encryption

[GSW13] Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. CRYPTO 2013.

$$q = 2^\ell$$

$$\mathbf{G}^t = \begin{pmatrix} 1 & 2 & \dots & 2^{\ell-1} & 0 & & \dots & & & \dots & & 0 \\ 0 & & \dots & 0 & 1 & 2 & \dots & 2^{\ell-1} & 0 & \dots & & 0 \\ \vdots & & & & & & & & \ddots & & & \vdots \\ 0 & & \dots & & & \dots & & & 0 & 1 & 2 & \dots & 2^{\ell-1} \end{pmatrix}$$

$$\mathrm{GSW}_{\mathbf{s}}(m) = (\mathbf{A} \mid \mathbf{s}\mathbf{A} + e) + m\mathbf{G} \in \mathbb{Z}_q^{n\ell \times n}$$

Decomposition

For all V, $\mathbf{G}^{-1}(V) \in \mathbb{Z}^{n\ell \times n\ell}$ is small and $\mathbf{G}^{-1}(V) \cdot \mathbf{G} = V$

Addition

$$C_1 + C_2 = (\mathbf{A}_1 + \mathbf{A}_2 \mid \mathbf{s}(\mathbf{A}_1 + \mathbf{A}_2) + e_1 + e_2) + m_1\mathbf{G} + m_2\mathbf{G}$$

Product

$$\mathbf{G}^{-1}(C_2) \cdot C_1$$

GSW encryption

$$C_1 = (\mathbf{A}_1 \mid \mathbf{s}\mathbf{A}_1 + e_1) \qquad C_2 = (\mathbf{A}_2 \mid \mathbf{s}\mathbf{A}_2 + e_2)$$

Given **s** and a ciphertext C computed as combined operations from (known) C_1, C_2, one can distinguish whether C is $C_1 + C_2$ or $\mathbf{G}^{-1}(C_1) \cdot C_1$;

and also recover (1,2) or (1,1);

Sanitization

Any distribution

Canonical distribution

Sanitization ⇒ Circuit Privacy

$\mathscr{C}_{\mathbf{x}}$: ciphertexts that decrypt to x

If $C_1, C_2 \in \mathscr{C}_{\mathbf{x}}$, $\left(\mathsf{pk}, \mathsf{sk}, \mathsf{Sanitize}(C_1)\right) \approx \left(\mathsf{pk}, \mathsf{sk}, \mathsf{Sanitize}(C_2)\right)$

Any distribution

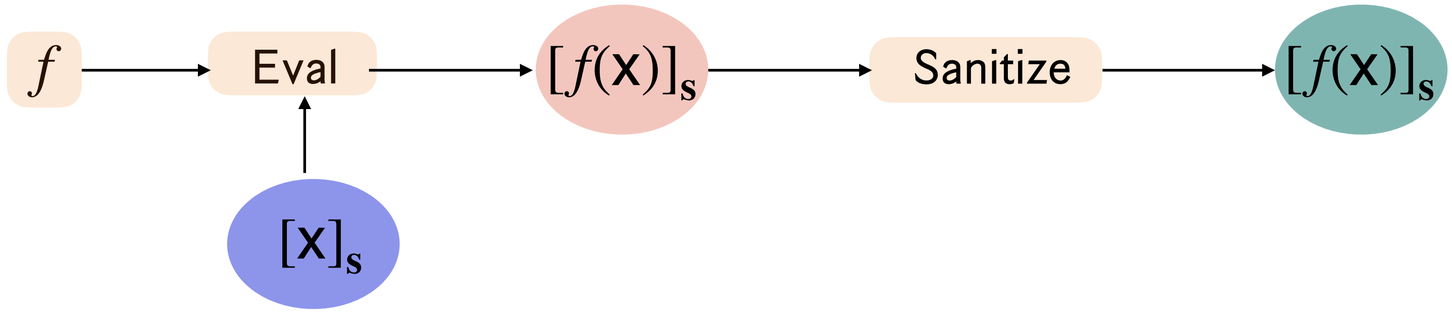
Canonical distribution

Sanitization ⇒ Circuit Privacy

Circuit Privacy for the class of functions $\mathcal{F}$ $(f_1, f_2 \in \mathcal{F})$:

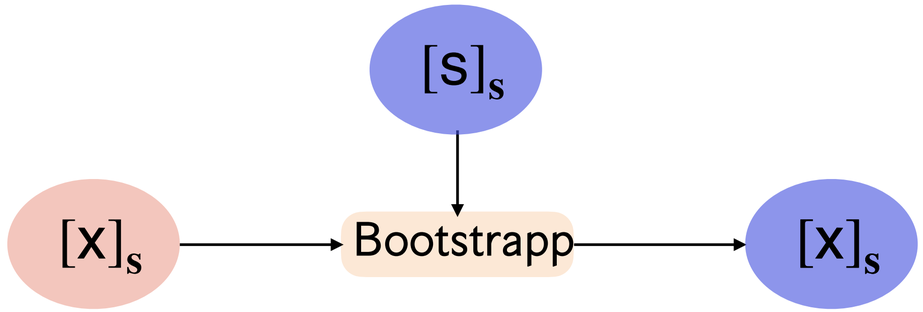
$\mathcal{C}_{\mathbf{x}}$: ciphertexts that decrypt to x

If $C_1 = \mathsf{Eval}(f_1, \star) \in \mathcal{C}_{\mathbf{x}}$, $C_2 = \mathsf{Eval}(f_2, \star) \in \mathcal{C}_{\mathbf{x}}$, $\left(\mathsf{pk}, \mathsf{sk}, C_1\right) \approx \left(\mathsf{pk}, \mathsf{sk}, C_2\right)$



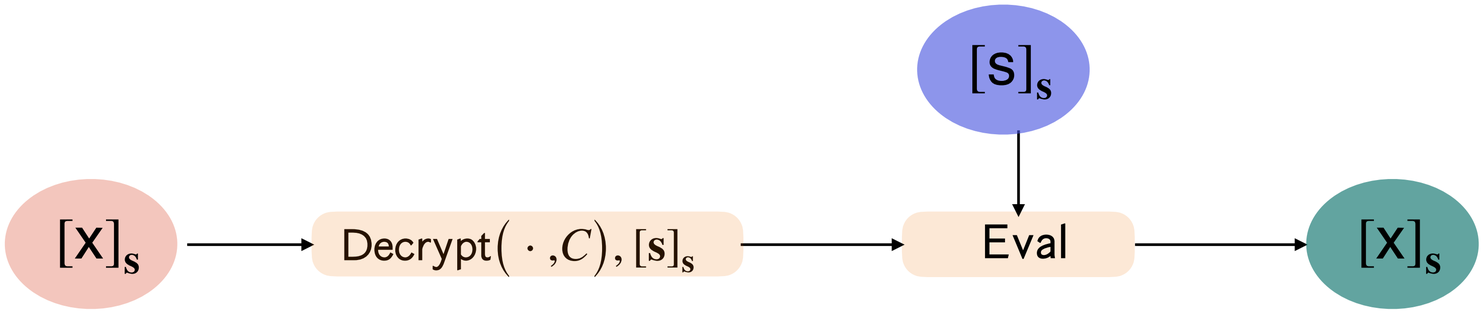
Circuit Privacy for Bootstrapping ⇒ Sanitization

$$\text{Bootstrapp}(C) = \text{Eval}\Big(\text{Decrypt}(\,\cdot\,, C), [\mathbf{s}]_{\mathbf{s}}\Big)$$



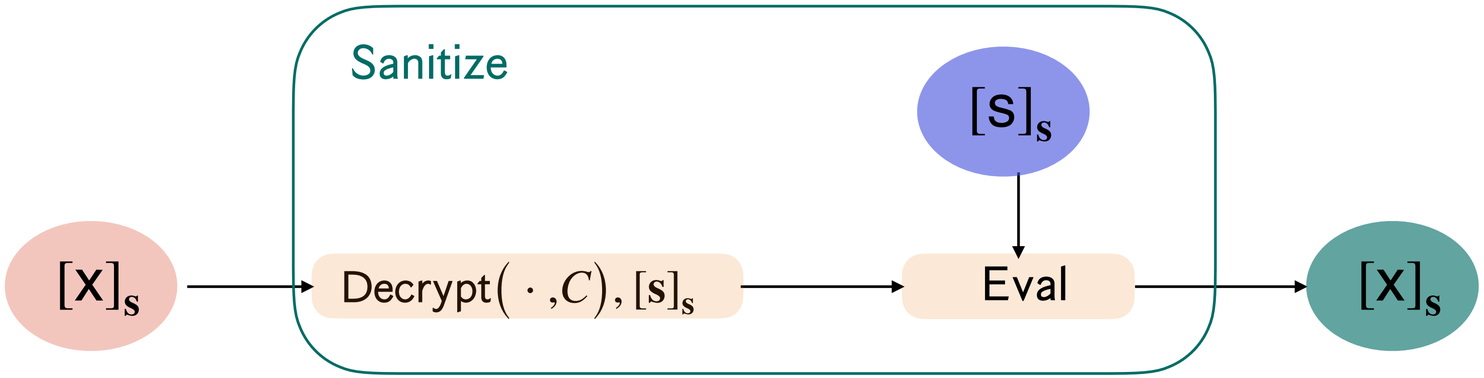
Circuit Privacy for Bootstrapping ⇒ Sanitization

Circuit Privacy of the class of functions $\mathsf{Eval}\left(\mathsf{Decrypt}(\,\cdot\,, C), [\mathbf{s}]_{\mathbf{s}}\right)$:



Circuit Privacy for Bootstrapping ⇒ Sanitization

Circuit Privacy of the class of functions $\mathsf{Eval}\left(\mathsf{Decrypt}(\,\cdot\,, C), [\mathbf{s}]_{\mathbf{s}}\right)$:



Sanitization

$\mathscr{C}_{\mathsf{x}}$: set of all the ciphertexts that decrypt to x

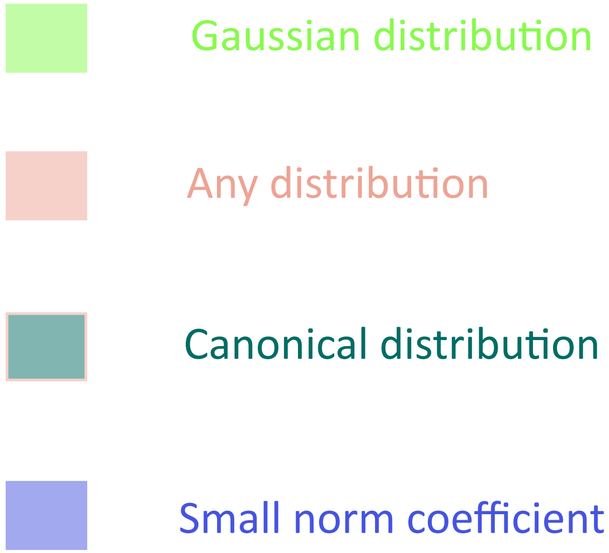
A Sanitize algorithm is a PPT algorithm s.t.:

1) For all ciphertext $C \in \mathscr{C}_{\mathsf{x}}$, $\mathsf{Sanitize}(\mathsf{PK}_s, C) \in \mathscr{C}_{\mathsf{x}}$

2) $\exists$ Sim s.t for any x, $C \in \mathscr{C}_{\mathsf{x}}$, $\left(\mathsf{Sim}(1^\lambda, \mathsf{PK}_s, \mathsf{x}), \mathsf{sk}\right) \approx_s \left(\mathsf{Sanitize}(\mathsf{PK}_s, C), \mathsf{sk}\right)$

Previous Approaches

Notations - color code



Based on Noise flooding, [Gentry09]

[Gentry09] Craig Gentry. A fully homomorphic encryption scheme. PhD Manuscript 2009, Stanford University.

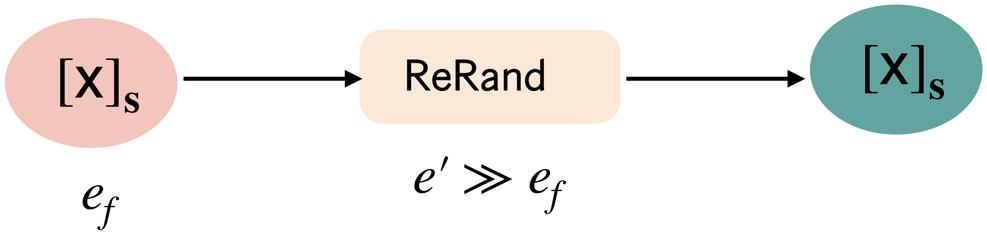
$$C_f = \mathsf{Eval}(\mathsf{evk}, f, C_1, \cdots, C_k)$$

e_f

Based on Noise flooding, [Gentry09]

[Gentry09] Craig Gentry. A fully homomorphic encryption scheme. PhD Manuscript 2009, Stanford University.

$$[x]_{\mathbf{s}} = \mathsf{Eval}(\mathsf{evk}, f, C_1, \cdots, C_k)$$



Add a fresh encryption of 0 with super polynomially larger noise
(Needs to carefully analyse distribution to avoid trivial encryption of 0)

- Add super-polynomial noise —> large modulus
- FHE scheme needs to support large noise
- Strong LWE assumption

Soak-and-spin-and-repeat, [DS16]

[DS16] Léo Ducas and Damien Stehlé. Sanitization of FHE ciphertexts. Eurocrypt 2016.

$$[x]_{\mathbf{s}} = \mathsf{Eval}(\mathsf{evk}, f, C_1, \cdots, C_k)$$

1 **Randomization step :**

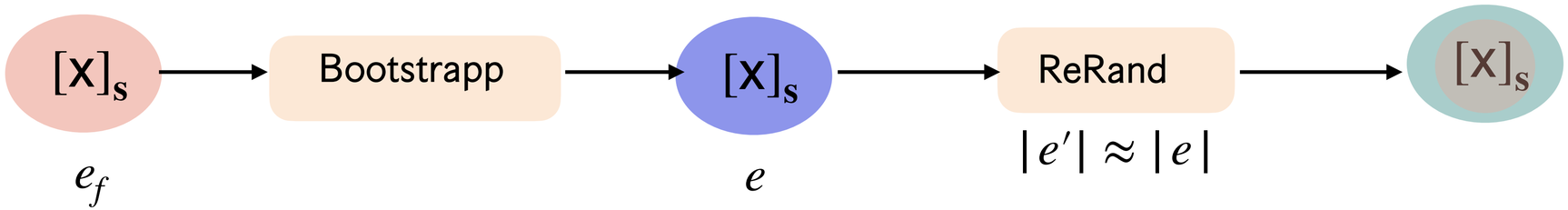
Soak-and-spin-and-repeat, [DS16]

[DS16] Léo Ducas and Damien Stehlé. Sanitization of FHE ciphertexts. Eurocrypt 2016.

$$[x]_{\mathbf{s}} = \mathsf{Eval}(\mathsf{evk}, f, C_1, \cdots, C_k)$$

1 **Randomization step :**

2 **Refreshing step (bootstrapping)**



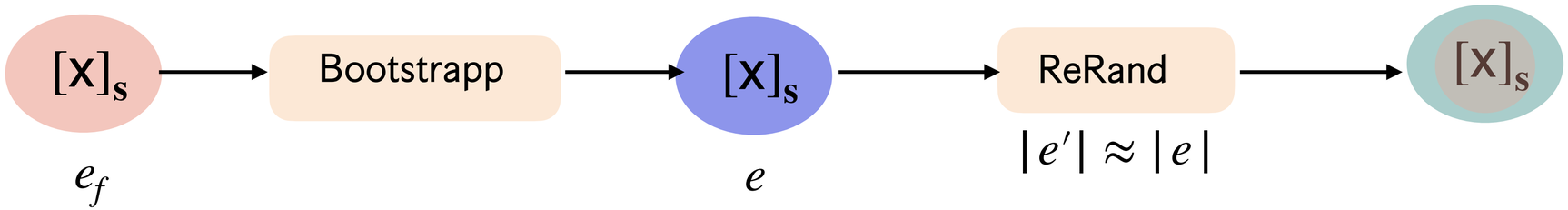
Soak-and-spin-and-repeat, [DS16]

[DS16] Léo Ducas and Damien Stehlé. Sanitization of FHE ciphertexts. Eurocrypt 2016.

$$[x]_{\mathbf{s}} = \mathsf{Eval}(\mathsf{evk}, f, C_1, \cdots, C_k)$$

1 **Randomization step :**

2 **Refreshing step (bootstrapping)**



3 **repeat**

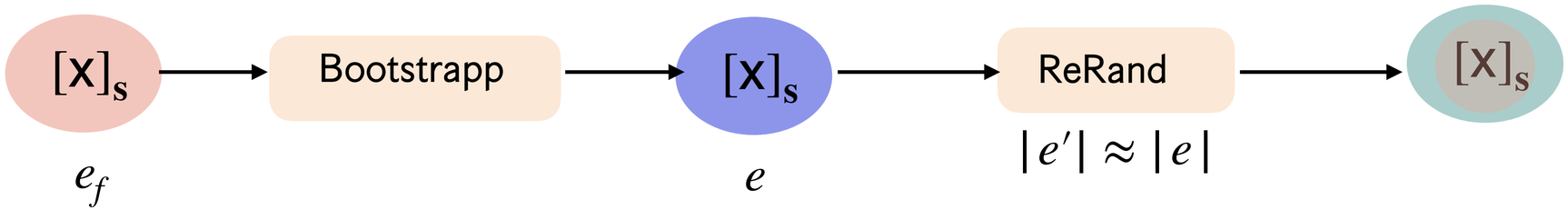
Soak-and-spin-and-repeat, [DS16]

[DS16] Léo Ducas and Damien Stehlé. Sanitization of FHE ciphertexts. Eurocrypt 2016.

$$[x]_{\mathbf{s}} = \mathsf{Eval}(\mathsf{evk}, f, C_1, \cdots, C_k)$$

1 **Randomization step :**

2 **Refreshing step (bootstrapping)**



3 **repeat**

- requires bootstrapping (circular security)
- parameters should be adapted for correctness

Circuit-private Branching Program, [BdMW16]

[BdMW16]. Florian Bourse, Rafa¨el del Pino, Michele Minelli, and Hoeteck Wee. FHE circuit privacy almost for free. CRYPTO 2016.

GSW encryption scheme

$$C_f = \text{Eval}\,(\text{evk}, f, C_1, \cdots, C_k)$$

Branching Program evaluation

Circuit-private Branching Program, [BdMW16]

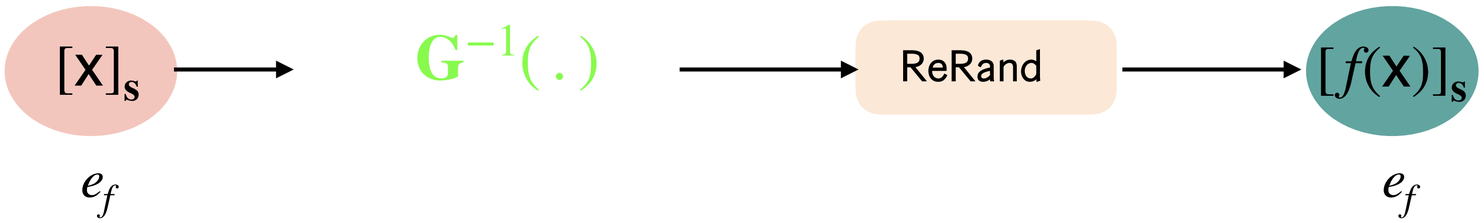
[BdMW16]. Florian Bourse, Rafa¨el del Pino, Michele Minelli, and Hoeteck Wee. FHE circuit privacy almost for free. CRYPTO 2016.

GSW encryption scheme

$$C_f = \text{Eval}\,(\text{evk}, f, C_1, \cdots, C_k)$$

Branching Program evaluation

[GPV08,Pei10,MP12,AP14]



Circuit-private Branching Program, [BdMW16]

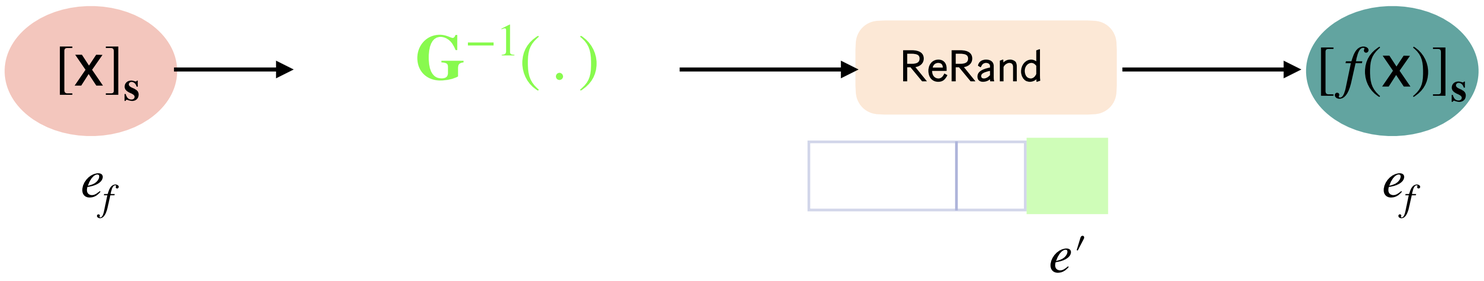
[BdMW16]. Florian Bourse, Rafa¨el del Pino, Michele Minelli, and Hoeteck Wee. FHE circuit privacy almost for free. CRYPTO 2016.

GSW encryption scheme

$$C_f = \text{Eval}\,(\text{evk}, f, C_1, \cdots, C_k)$$

Branching Program evaluation

[GPV08,Pei10,MP12,AP14]



- Pros: No circular security assumption
- Cons: f is a BP-like evaluation and applies to GSW ciphertexts

Choosing ReRand

- Public Key: $0(n \log q)$ LWE encryptions of 0;
- ReRand: random linear combinations of encryptions from the public key + noise;
- Statistical security.

In our work:

- Public Key: one RLWE encryption of 0;
- ReRand: one RLWE encryption of 0 + noise;
- Computational security (under the decisional RLWE).

R(LWE), R(GSW) encryptions and homomorphic operations

LWE encryption

[Regev05] On lattices, learning with errors, random linear codes, and cryptography

Encryption

secret $\mathbf{s} = (s_1, \cdots, s_n) \in \mathbb{Z}^n$, m is a message in $\mathbb{Z}_t$, $\Delta = \lfloor \frac{q}{t} \rceil$

$$\mu^\star = \Delta m + e \quad \text{and} \quad \mathsf{LWE}_{\mathbf{s}}(m) = (\mathbf{a}, b = \sum_i a_i s_i + \mu^\star) \in \mathbb{Z}_q^n \times \mathbb{Z}_q,$$

where $\mathbf{a}$ is random in $\mathbb{Z}_q^n$ and e is a discrete Gaussian of stdev σ (and variance σ^2)

Decryption

1) compute : $\bar{\mu}^\star = b - \sum_i a_i s_i = \Delta m + e$

2) decode the noisy encoding : $\lfloor \frac{\bar{\mu}^*}{\Delta} \rceil$

RingLWE encryption

$q := 2^{\ell}$ or products of primes

N power of 2

$$\mathbb{Z}[X] := \mathbb{Z}(X) \mod X^N + 1$$

$$\mathbb{Z}_q[X] := \mathbb{Z}_q(X) \mod X^N + 1$$

Encryption

$$m(X) \in \mathbb{Z}[X]$$

$$s(X) = s_0 + s_1 X + \cdots + s_{N-1} X^{N-1} \text{ with } s_i \in \{0,1\}$$

$a(X)$

$b(X) = a(X) \cdot s(X) + \Delta \cdot m(X) + e(X)$

Decryption

1 compute $\text{encode}^*(m(X)) = b(X) - a(X)s(X) = \Delta \cdot m(X) + e(X)$

2 round the result : $\lfloor \frac{\Delta \cdot m(X) + e(X)}{\Delta} \rceil$

RingGSW encryption

$q := 2^{\ell}$ or products of primes

N power of 2

$\mathbb{Z}[X] := \mathbb{Z}(X) \mod X^N + 1$

$\mathbb{Z}_q[X] := \mathbb{Z}_q(X) \mod X^N + 1$

Encryption

$m(X) \in \mathbb{Z}[X]$

$s(X) = s_0 + s_1 X + \cdots + s_{N-1} X^{N-1}$ with $s_i \in \{0,1\}$

$$\mathsf{RGSW}(s(X), m(X)) := \overbrace{\begin{bmatrix} a_1(X) & a_1(X)\cdot s(X) + e_1(X) \\ \vdots & \vdots \\ a_\ell(X) & a_\ell(X)\cdot s(X) + e_\ell(X) \\ a'_1(X) & a'_1(X)\cdot s(X) + e'_1(X) \\ \vdots & \vdots \\ a'_\ell(X) & a'_\ell(X)\cdot s(X) + e'_\ell(X) \end{bmatrix}}^{\mathsf{RLWE}(s(X),\mathbf{0})} + m(X)\cdot \overbrace{\begin{bmatrix} 1 & 0 \\ \vdots & \vdots \\ 2^{\ell-1} & 0 \\ 0 & 1 \\ \vdots & \vdots \\ 0 & 2^{\ell-1} \end{bmatrix}}^{\mathbf{G}}$$

RingGSW encryption

$$\mathsf{RGSW}(s(X), m(X)) := \overbrace{\begin{bmatrix} a_1(X) & a_1(X)\cdot s(X) + e_1(X) \\ \vdots & \vdots \\ a_\ell(X) & a_\ell(X)\cdot s(X) + e_\ell(X) \\ a'_1(X) & a'_1(X)\cdot s(X) + e'_1(X) \\ \vdots & \vdots \\ a'_\ell(X) & a'_\ell(X)\cdot s(X) + e'_\ell(X) \end{bmatrix}}^{\mathsf{RLWE}(\mathbf{s}(X),\mathbf{0})} + m(X)\cdot \overbrace{\begin{bmatrix} 1 & 0 \\ \vdots & \vdots \\ 2^{\ell-1} & 0 \\ 0 & 1 \\ \vdots & \vdots \\ 0 & 2^{\ell-1} \end{bmatrix}}^{\mathbf{G}}$$

Addition

multiplication by a small constant

internal multiplication

$$\overbrace{\mathsf{RGSW}(s(X), m_1(X))}^{C_1} \times \overbrace{\mathsf{RGSW}(s(X), m_2(X))}^{C_2} =^\star \overbrace{\mathbf{G}^{-1}(C_2)}^{\in R^{2\ell\times 2}} \times C_1$$

$$=^\star \mathsf{RGSW}(s(X), m_1(X)\cdot m_2(X))$$

RingGSW encryption

$$\mathsf{RGSW}(s(X), m(X)) := \overbrace{\begin{bmatrix} a_1(X) & a_1(X)\cdot s(X) + e_1(X) \\ \vdots & \vdots \\ a_\ell(X) & a_\ell(X)\cdot s(X) + e_\ell(X) \\ a'_1(X) & a'_1(X)\cdot s(X) + e'_1(X) \\ \vdots & \vdots \\ a'_\ell(X) & a'_\ell(X)\cdot s(X) + e'_\ell(X) \end{bmatrix}}^{\mathsf{RLWE}(\mathbf{s}(X),\mathbf{0})} + m(X)\cdot \overbrace{\begin{bmatrix} 1 & 0 \\ \vdots & \vdots \\ 2^{\ell-1} & 0 \\ 0 & 1 \\ \vdots & \vdots \\ 0 & 2^{\ell-1} \end{bmatrix}}^{\mathbf{G}}$$

Addition

multiplication by a small constant

internal multiplication

external multiplication

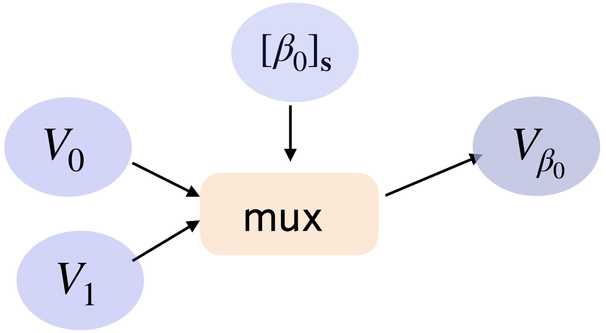
$$\overbrace{\mathsf{RGSW}(s(X), m_1(X))}^{C_1} \odot \overbrace{\mathsf{RLWE}(s(X), m_2(X))}^{C_2} =^\star \mathbf{G}^{-1}(C_2)\cdot C_1$$

B' B $O(B') + |m_1(X)|_1 B$

$$\ldots. =^\star \mathsf{RLWE}(s(X), m_1(X)\cdot m_2(X))$$

Multiplexer randomization

Homomorphic multiplexer

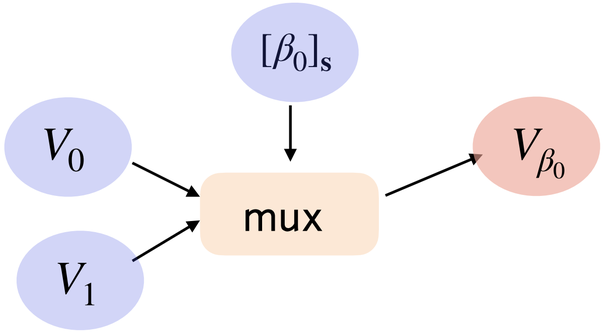


$$\mathsf{MUX}(\beta_0, v_0, v_1) = v_1 + \beta_0(v_0 - v_1), \quad \beta_0 \in \{0,1\}$$

$$V_0 := [v_0] \text{ and } V_1 := [v_1]$$

$$V_{\beta_0} = V_1 + \mathbf{G}^{-1}(V_0 - V_1) \cdot [\beta_0]$$

Homomorphic multiplexer

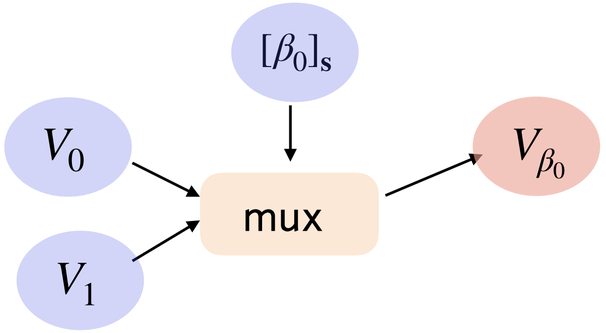


$$\mathsf{MUX}(\beta_0, v_0, v_1) = v_1 + \beta_0(v_0 - v_1), \quad \beta_0 \in \{0,1\}$$

$$V_0 := [v_0] \text{ and } V_1 := [v_1]$$

$$V_{\beta_0} = V_1 + \mathbf{G}^{-1}(V_0 - V_1) \cdot [\beta_0]$$

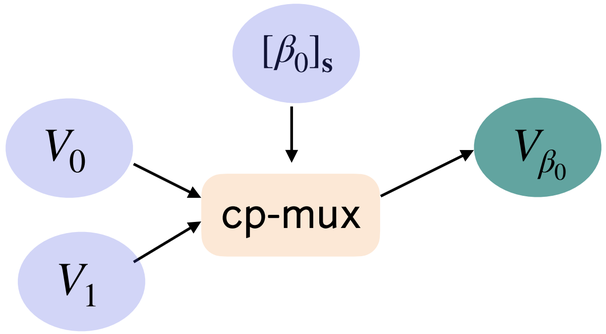
Homomorphic multiplexer



$$\mathsf{MUX}(\beta_0, v_0, v_1) = v_1 + \beta_0(v_0 - v_1), \quad \beta_0 \in \{0,1\}$$

$$V_0 := [v_0] \text{ and } V_1 := [v_1]$$

$$V_{\beta_0} = V_1 + \mathbf{G}^{-1}(V_0 - V_1) \cdot [\beta_0]$$



Using [BdMW16]'s result:

$$V_{\text{next}} = V_1 + \mathbf{G}^{-1}(V_0 - V_1) \cdot [\beta_0] + (\ \mathbf{0} \mid y\ )$$

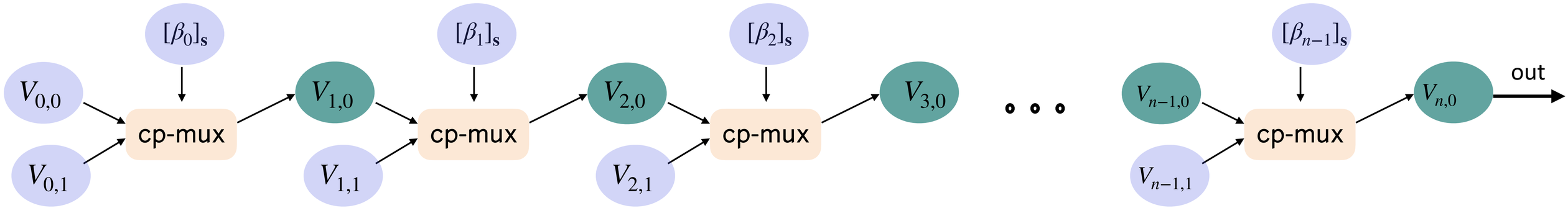
$$= V_{\beta_0} + \mathbf{G}^{-1}(V_0 - V_1) \cdot [0] + (\ \mathbf{0} \mid y\ )$$

$$\approx_s V_{\beta_0} + V_{\mathsf{f}}$$

fresh encryption of 0

does not depend on $V_0 - V_0'$

Circuit-private Branching program-like evaluation, [BdMW16]



TFHE bootstrapping

TFHE building blocks

Extract

RLWE → LWE

$$\mathsf{RLWE}(s(X), m_0 + m_1 X + \cdots + m_{N-1} X^{N-1}) \rightarrow \mathsf{LWE}(\tilde{s}, m_i)$$

TFHE building blocks

Extract

RLWE → LWE

$$\mathsf{RLWE}(s(X), m_0 + m_1 X + \cdots + m_{N-1} X^{N-1}) \rightarrow \mathsf{LWE}(\tilde{s}, m_i)$$

BlindRotate

RGSW × RLWE → RLWE

$$\mathsf{RGSW}(s(X), X^i) \odot \mathsf{RLWE}(s(X), m(X)) \longrightarrow \mathsf{RLWE}(s(X), m(X) \cdot X^i)$$

TFHE building blocks

Extract

RLWE → LWE

$$\mathsf{RLWE}(s(X), m_0 + m_1 X + \cdots + m_{N-1} X^{N-1}) \rightarrow \mathsf{LWE}(\tilde{s}, m_i)$$

BlindRotate

RGSW × RLWE → RLWE

$$\mathsf{RGSW}(s(X), X^i) \odot \mathsf{RLWE}(s(X), m(X)) \longrightarrow \mathsf{RLWE}(s(X), m(X) \cdot X^i)$$

KeySwitch

LWE × LWE → LWE

$$\mathsf{KeySwitch}\left(\mathsf{KSK}_{\mathbf{s}\rightarrow\mathbf{s}'}\ ,\ \mathsf{LWE}_{\mathbf{s}}(m)\right) \longrightarrow \mathsf{LWE}_{\mathbf{s}'}(m)$$

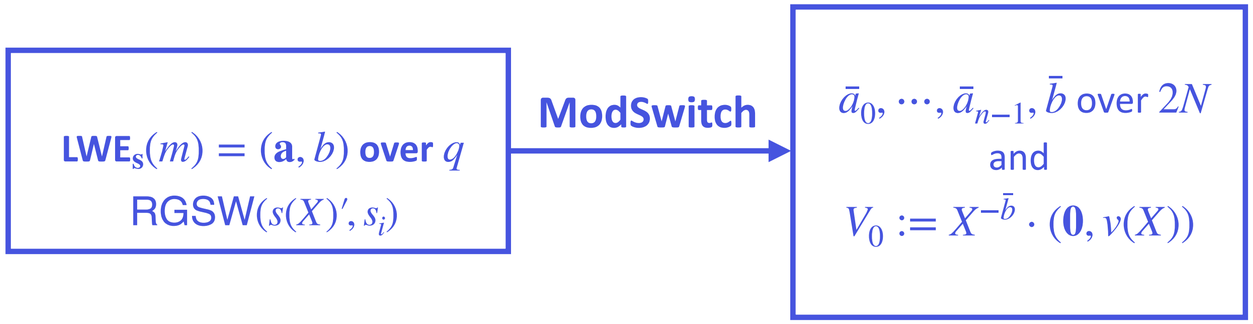
(also works: $\mathsf{LWE}_{\mathbf{s}}(m) \longrightarrow \mathsf{LWE}_{\mathbf{s}'}\left(f(m)\right)$)

TFHE bootstrapping

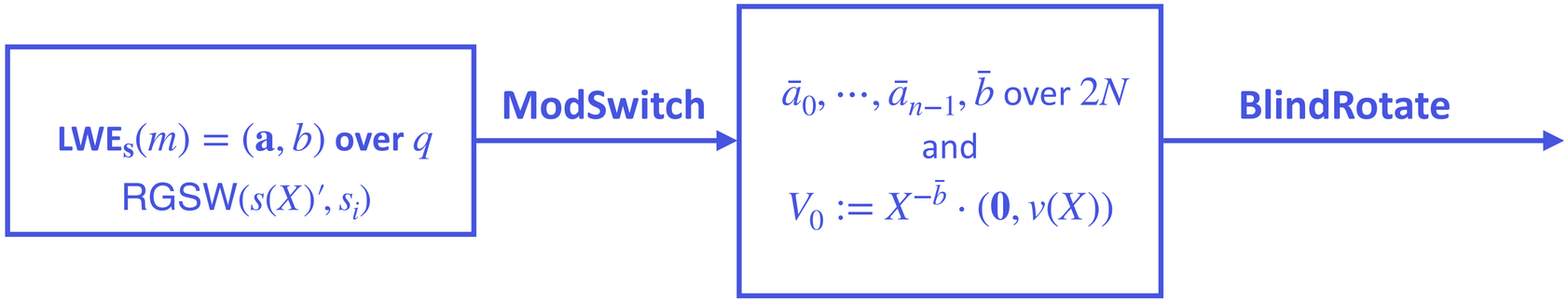
$\mathbf{LWE_s}(m) = (\mathbf{a}, b)$ **over** q

RGSW$(s(X)', s_i)$

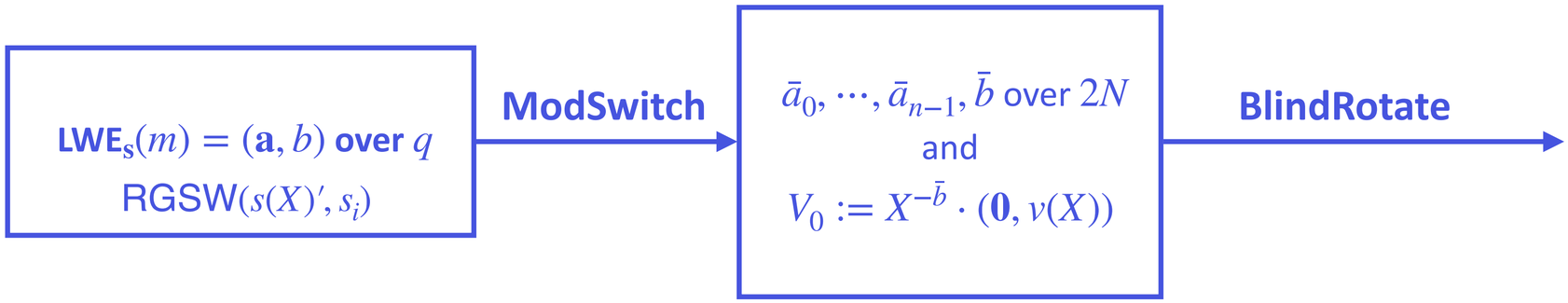
TFHE bootstrapping

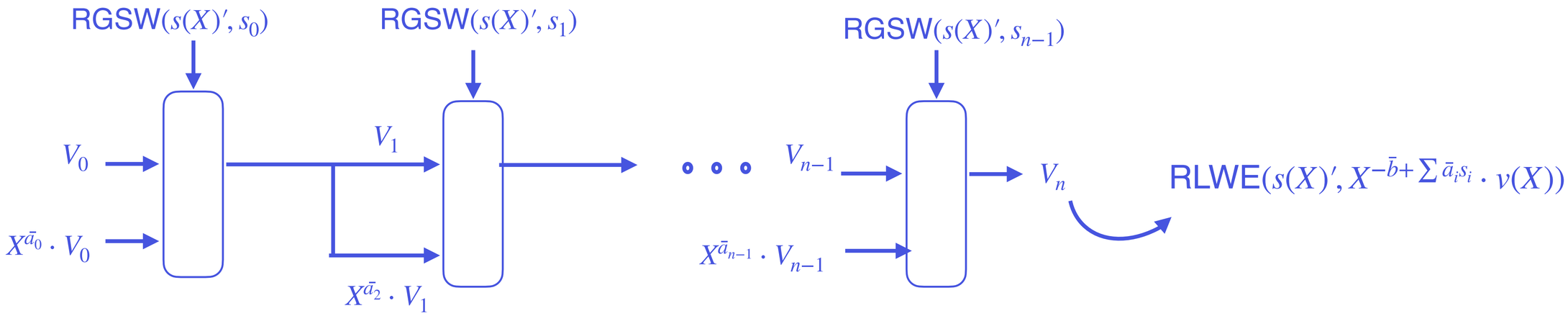


TFHE bootstrapping

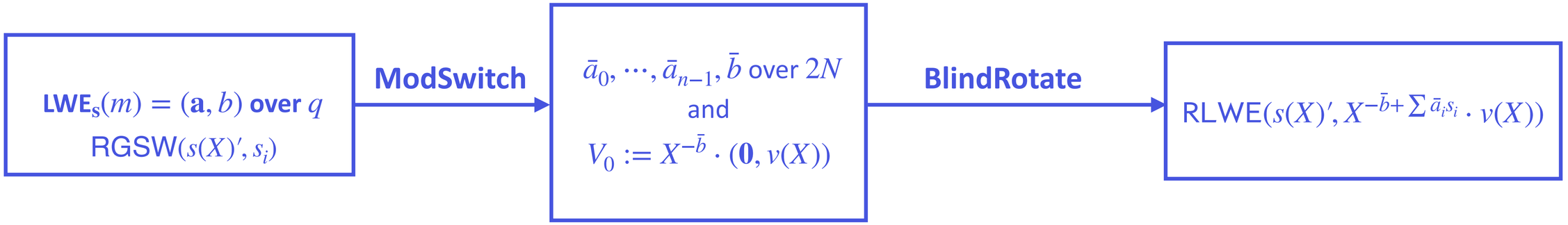


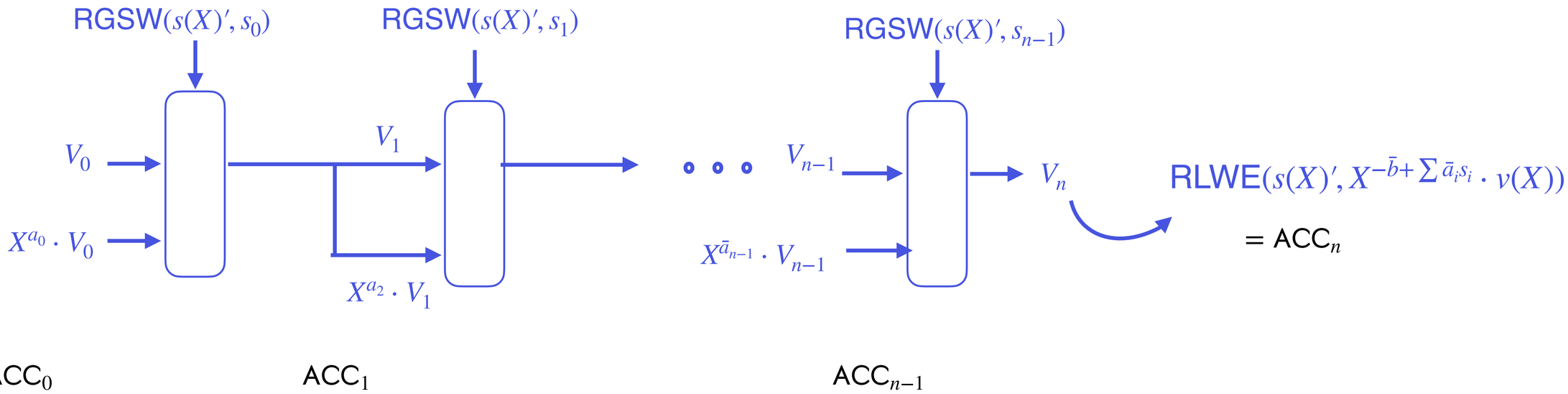
TFHE bootstrapping



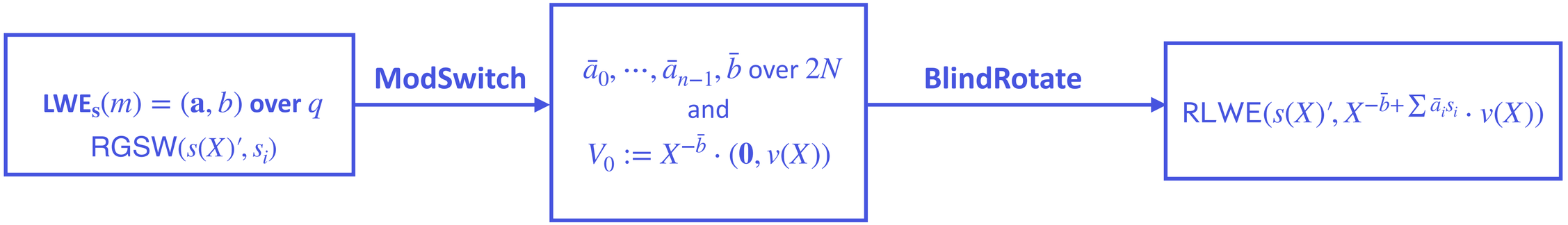


TFHE bootstrapping

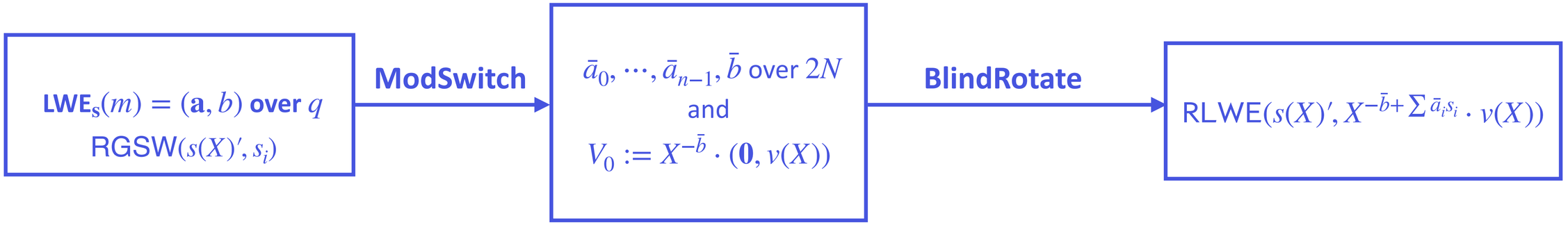




TFHE bootstrapping

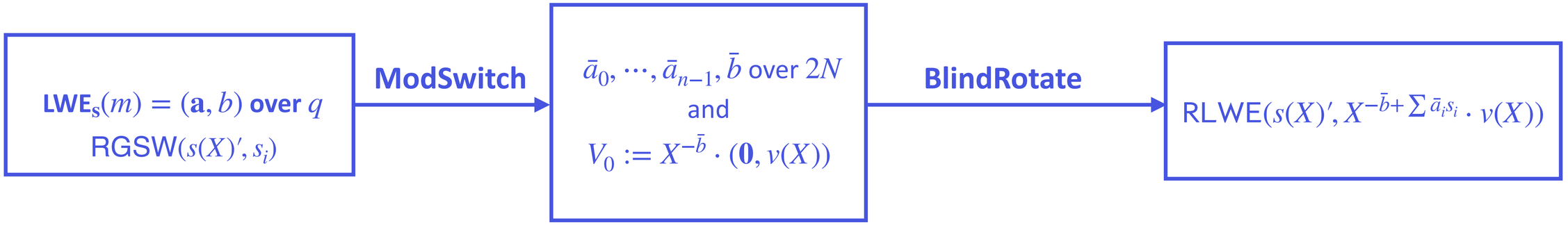


TFHE bootstrapping



$$X^{-j} \cdot v(X) = \begin{cases} v_j + \cdots & \text{if } 0 \leq j \leq N \\ -v_j + \cdots & \text{otherwise} \end{cases}$$

TFHE bootstrapping

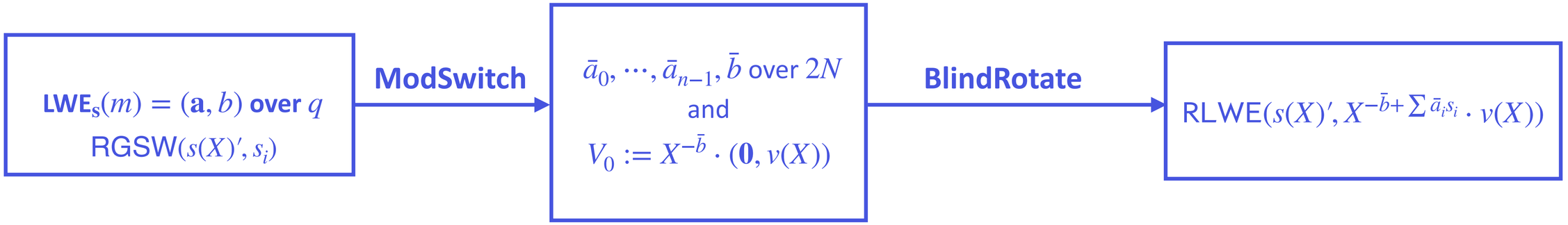


$$X^{-j} \cdot v(X) = \begin{cases} v_j + \cdots & \text{if } 0 \le j \le N \\ -v_j + \cdots & \text{otherwise} \end{cases}$$

If $v_j \in \mathbb{Z}_p$ is defined as the rounding of noisy messages $j \in \mathbb{Z}_{2N}$

i.e. $v_j := \dfrac{\lfloor \frac{pj}{2N} \rceil \mod p}{p}$ and taking $j = \bar{b} - \bar{\mathbf{a}} \cdot \mathbf{s}$:

TFHE bootstrapping



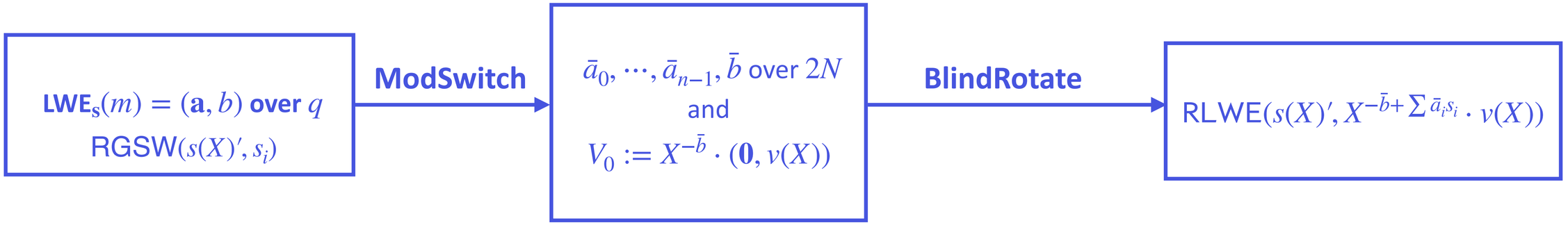
$$X^{-j} \cdot v(X) = \begin{cases} v_j + \cdots & \text{if } 0 \le j \le N \\ -v_j + \cdots & \text{otherwise} \end{cases}$$

If $v_j \in \mathbb{Z}_p$ is defined as the rounding of noisy messages $j \in \mathbb{Z}_{2N}$

i.e. $v_j := \dfrac{\lfloor \frac{pj}{2N} \rceil \mod p}{p}$ and taking $j = \bar{b} - \bar{\mathbf{a}} \cdot \mathbf{s}$:

$$\mathrm{coeff}_0\left(X^{-j} \cdot (v_{N-1}X^{N-1} + \cdots + v_1 X + v_0)\right) = v_j$$

TFHE bootstrapping



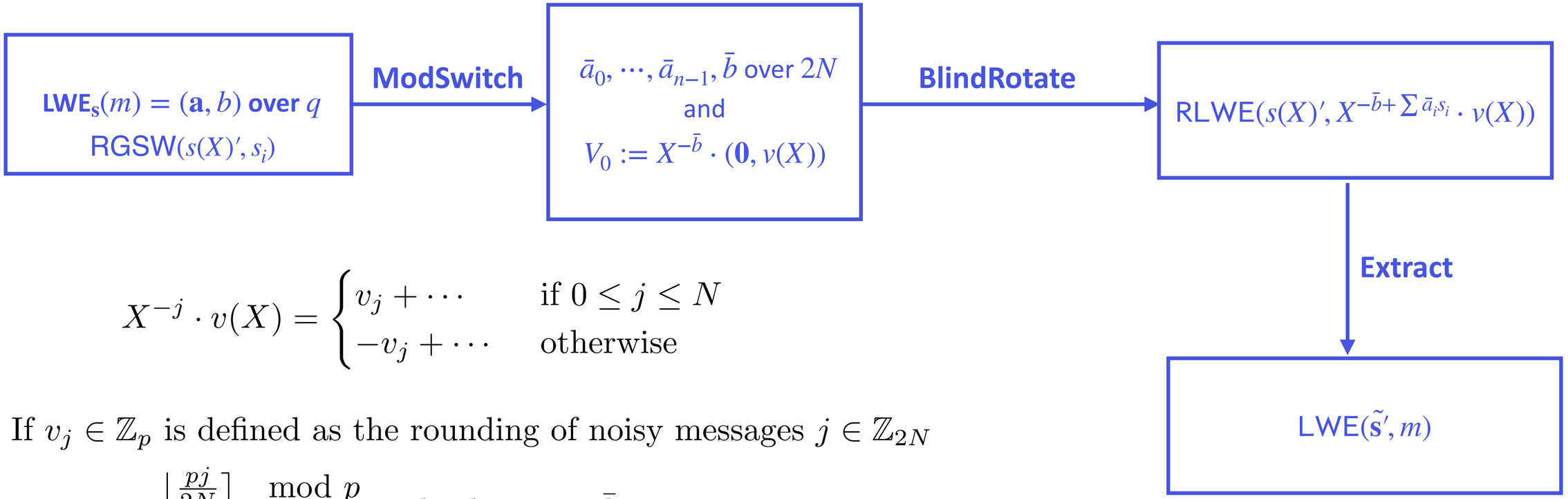
$$X^{-j} \cdot v(X) = \begin{cases} v_j + \cdots & \text{if } 0 \le j \le N \\ -v_j + \cdots & \text{otherwise} \end{cases}$$

If $v_j \in \mathbb{Z}_p$ is defined as the rounding of noisy messages $j \in \mathbb{Z}_{2N}$

i.e. $v_j := \dfrac{\lfloor \frac{pj}{2N} \rceil \mod p}{p}$ and taking $j = \bar{b} - \bar{\mathbf{a}} \cdot \mathbf{s}$:

$$\text{coeff}_0\left(X^{-j} \cdot (v_{N-1}X^{N-1} + \cdots + v_1 X + v_0)\right) = v_j = m$$

TFHE bootstrapping



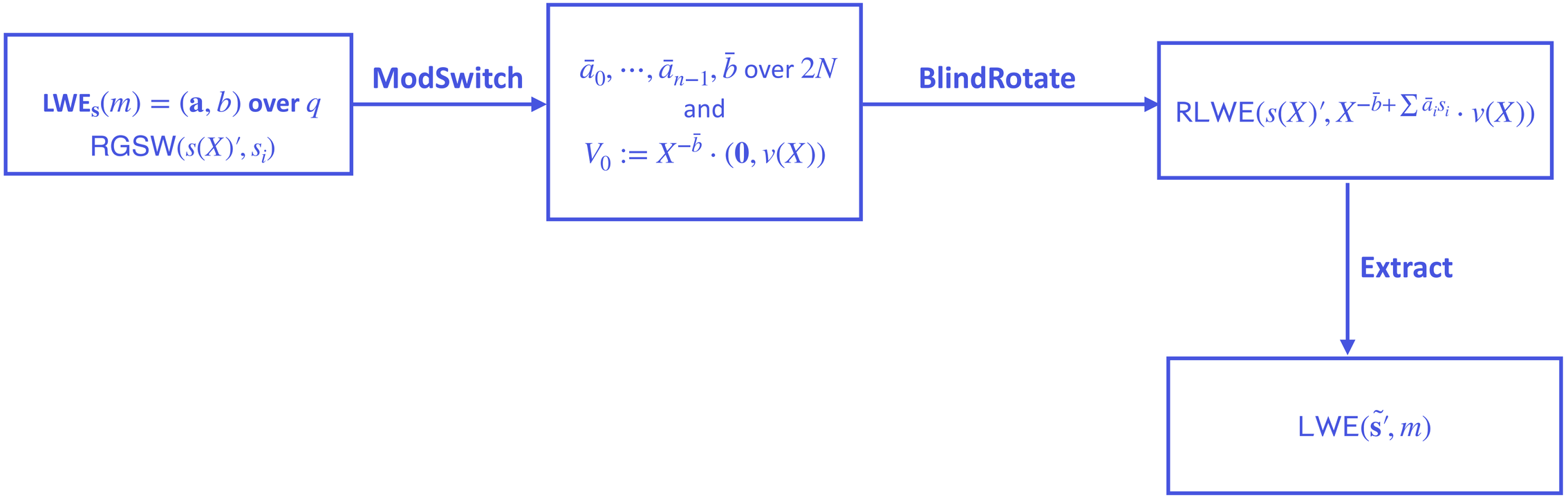
$$X^{-j} \cdot v(X) = \begin{cases} v_j + \cdots & \text{if } 0 \le j \le N \\ -v_j + \cdots & \text{otherwise} \end{cases}$$

If $v_j \in \mathbb{Z}_p$ is defined as the rounding of noisy messages $j \in \mathbb{Z}_{2N}$

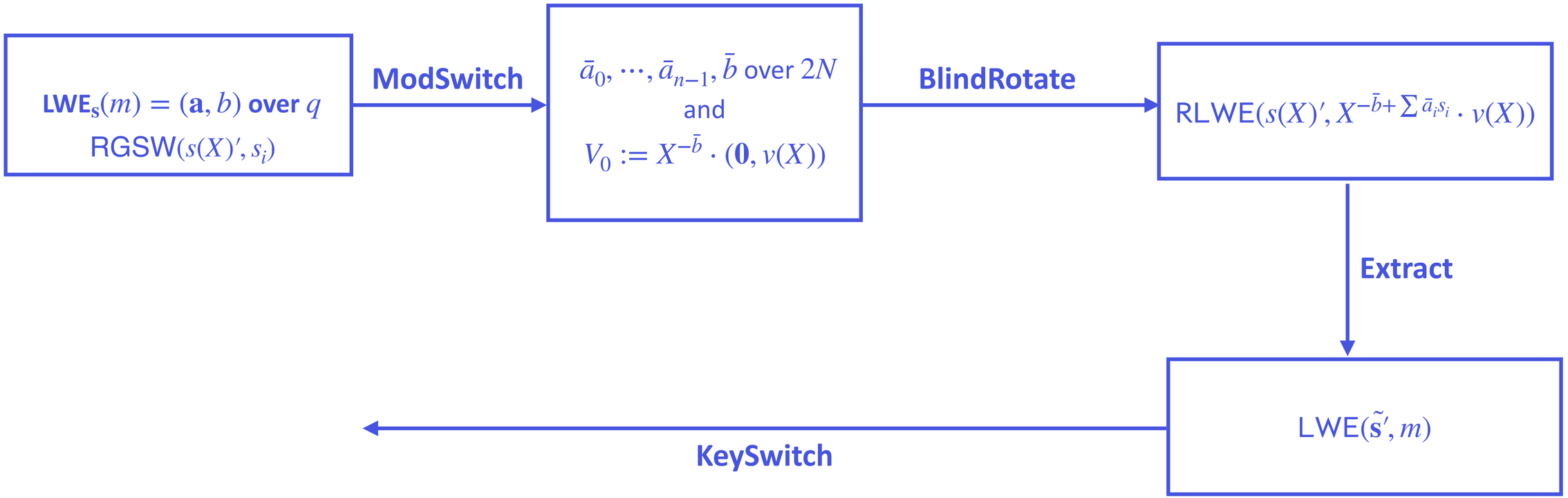
i.e. $v_j := \dfrac{\lfloor \frac{pj}{2N} \rceil \mod p}{p}$ and taking $j = \bar{b} - \bar{\mathbf{a}} \cdot \mathbf{s}$:

$$\mathrm{coeff}_0\left(X^{-j} \cdot (v_{N-1}X^{N-1} + \cdots + v_1 X + v_0)\right) = v_j = m$$

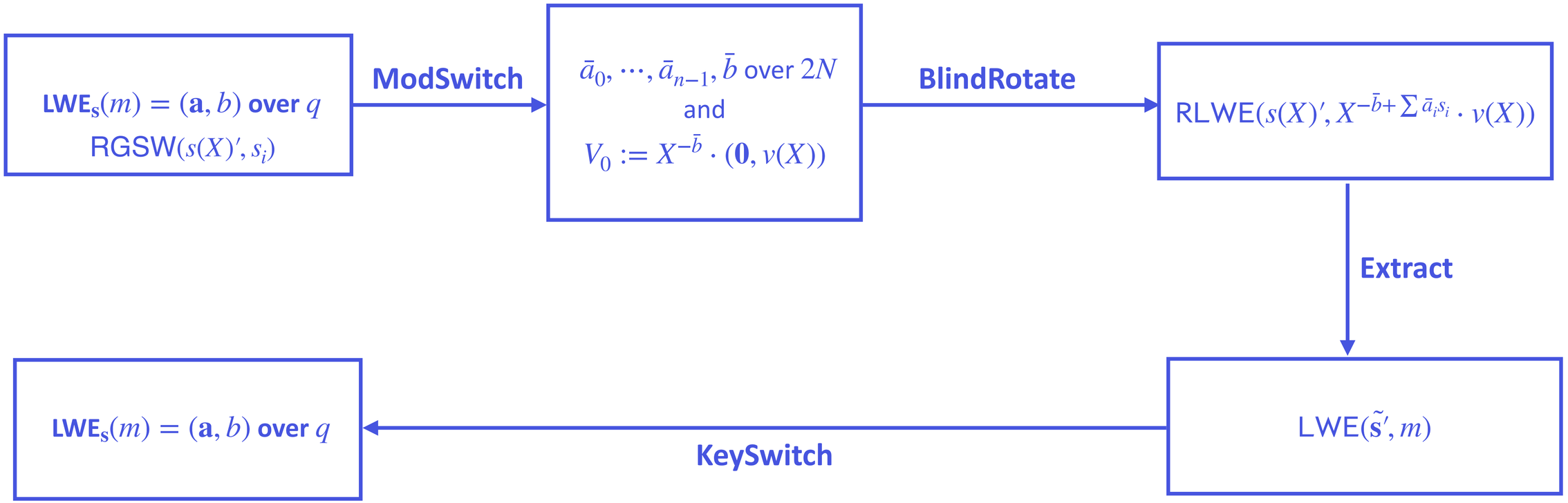
TFHE bootstrapping



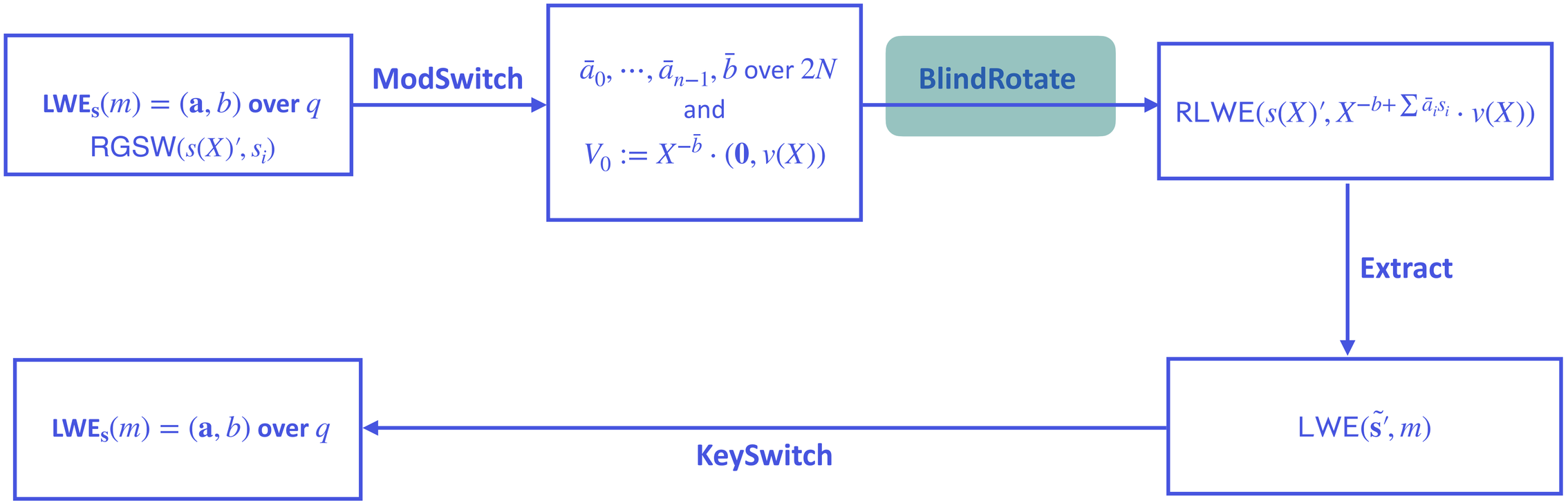
TFHE bootstrapping



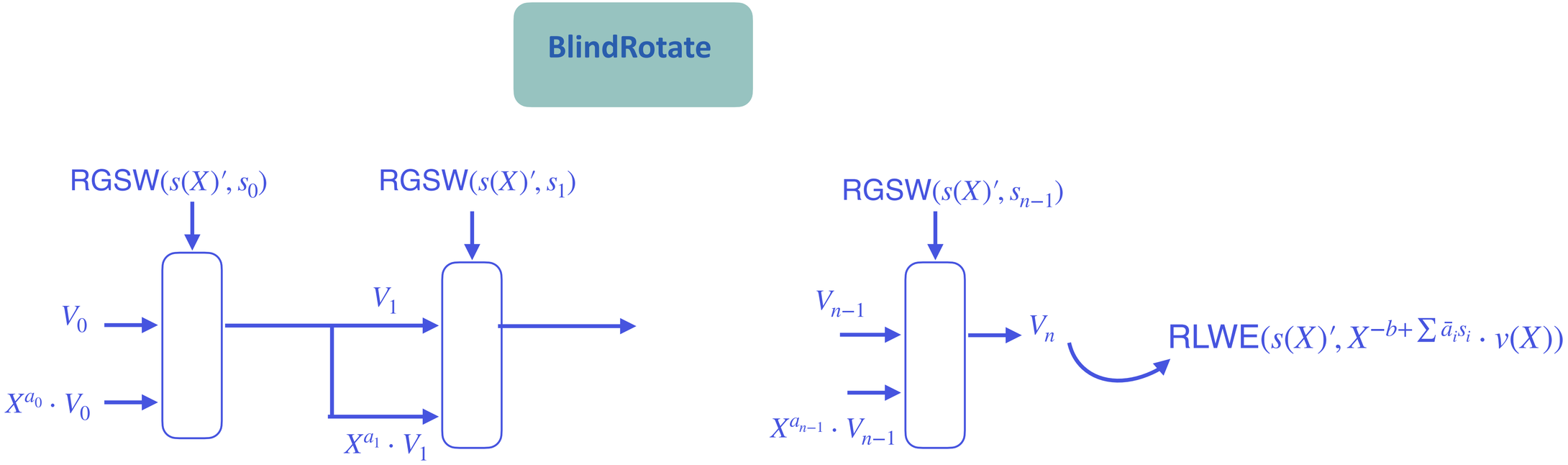
TFHE bootstrapping



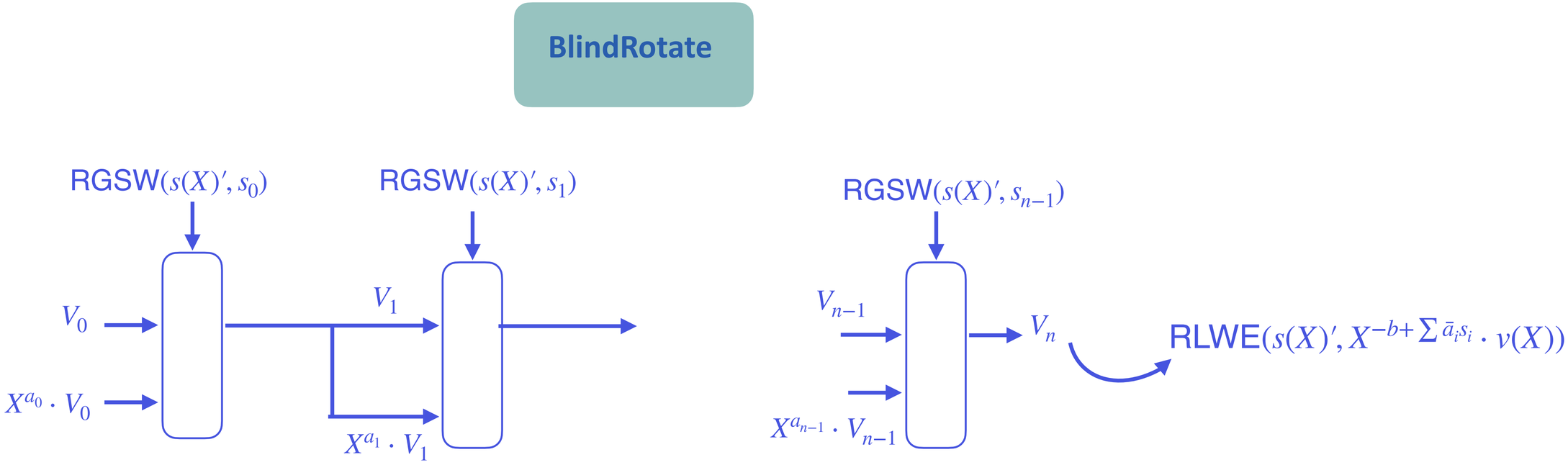
TFHE bootstrapping



TFHE bootstrapping



First Strategy: applying [BdMW16] over rings



$$V_t = V_{b_{t-1}} + \mathbf{G}^{-1}(V_{t-1} - V'_{t-1}) \cdot \mathsf{RGSW}(0) + \left(\begin{array}{c|c} \mathbf{0} & y \end{array} \right)$$

fresh GSW encryption of 0

$$\approx_s V_{b_{t-1}} + C_{\mathsf{f}}$$

Circuit-private BlindRotate

Uniformity over R_q

$$V_{b_{t-1}} + \overbrace{\mathbf{G}^{-1}_{\text{rand}}(V_{t-1} - V'_{t-1})}^{x} \cdot \mathsf{RGSW}(0) + \left(\begin{array}{c|c} \mathbf{0} & y \end{array} \right) \approx_s V_{b_{t-1}} + V_{\mathsf{f}}$$

$$x \cdot \left(\begin{array}{c|c} B & \mathbf{s}B + e \end{array} \right) + \left(\begin{array}{c|c} \mathbf{0} & y \end{array} \right) \approx_s V_{\mathsf{f}}$$

Needs to show :

- LEFT PART: uniformity of $x \cdot B$
- RIGHT PART: noise is Gaussian of parameter independent of that of V_{t-1}, V'_{t-1}: $x \cdot (\mathbf{s} \cdot B + e + y)$

Uniformity over R_q

$$V_{b_{t-1}} + \overbrace{\mathbf{G}^{-1}_{\text{rand}}(V_{t-1} - V'_{t-1})}^{x} \cdot \mathsf{RGSW}(0) + \left(\begin{array}{c|c} \mathbf{0} & y \end{array} \right) \approx_s V_{b_{t-1}} + V_{\mathsf{f}}$$

$$x \cdot \left(\begin{array}{c|c} B & \mathbf{s}B + e \end{array} \right) + \left(\begin{array}{c|c} \mathbf{0} & y \end{array} \right) \approx_s V_{\mathsf{f}}$$

Needs to show :

- LEFT PART: uniformity of $x \cdot B \overset{?}{\approx}_s \mathbf{u}$, with $\mathbf{u}$ uniform
- RIGHT PART: noise is Gaussian of parameter independent of that of V_{t-1}, V'_{t-1}: $x \cdot (\mathbf{s} \cdot B + e + y)$

Uniformity over R_q

$$V_{b_{t-1}} + \overbrace{\mathbf{G}^{-1}_{\text{rand}}(V_{t-1} - V'_{t-1})}^{x} \cdot \text{RGSW}(0) + \left(\begin{array}{c|c} \mathbf{0} & y \end{array} \right) \approx_s V_{b_{t-1}} + V_{\mathsf{f}}$$

$$x \cdot \left(\begin{array}{c|c} B & \mathbf{s}B + e \end{array} \right) + \left(\begin{array}{c|c} \mathbf{0} & y \end{array} \right) \approx_s V_{\mathsf{f}}$$

Needs to show :

- LEFT PART: uniformity of $x \cdot B \overset{?}{\approx}_s \mathbf{u}$
- RIGHT PART: noise is Gaussian of parameter independent of that of V_{t-1}, V'_{t-1}:

$$\begin{aligned} & x \cdot (\mathbf{s} \cdot B + e + y) \\ = \; & x \cdot B \cdot \mathbf{s} + x \cdot e + y \\ \approx_s \; & \mathbf{u} \cdot \mathbf{s} + x \cdot e + y \end{aligned}$$

$x \cdot e + y \approx_s e'$, for any V_{t-1}, V'_{t-1}

Uniformity over R_q, q power of 2

$$V_{b_{t-1}} + \overbrace{\mathbf{G}^{-1}_{\text{rand}}(V_{t-1} - V'_{t-1})}^{x} \cdot \text{RGSW}(0) + \left(\mathbf{0} \mid y \right) \approx_s V_{b_{t-1}} + V_{\mathsf{f}}$$

$$x \cdot \left(B \mid sB + e \right) + \left(\mathbf{0} \mid y \right) \approx_s V_{\mathsf{f}}$$

Needs to show :

- LEFT PART: uniformity of $x \cdot B$
- RIGHT PART: noise is Gaussian of parameter independent of that of V_{t-1}, V'_{t-1}: $x \cdot (\mathbf{s} \cdot B + e + y)$

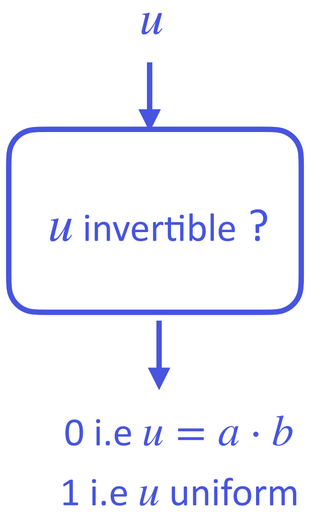
What happens over R_q,
q a power of 2 ?

Uniformity over R_q, q power of 2

Take a and $b \in \mathbb{Z}_2$, is the product $a \cdot b$ uniform over $\mathbb{Z}_2$? ($q = 2,\ N = 2^0$)

Uniformity over R_q, q power of 2

Take a and $b \in \mathbb{Z}_2$, is the product $a \cdot b$ uniform over $\mathbb{Z}_2$? $(q = 2,\ N = 2^0)$



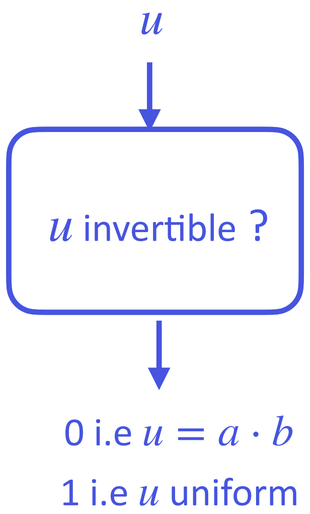
a	b	$u = a \wedge b$	u unif
0	0	0	0
0	1	0	0
1	0	0	1
1	1	1	1

$$\Pr[u \text{ non invertible} \mid u \text{ uniform}] = \frac{1}{2}$$

$$\Pr[u \text{ non invertible} \mid u = a \cdot b] = \underbrace{\frac{1}{2}}_{a \text{ inv}} + \underbrace{\frac{1}{2} \cdot \frac{1}{2}}_{a \text{ non inv}}$$

Uniformity over R_q, q power of 2

Take a and $b \in \mathbb{Z}_2$, is the product $a \cdot b$ uniform over $\mathbb{Z}_2$? $(q = 2,\ N = 2^0)$



a	b	$u = a \wedge b$	u unif
0	0	0	0
0	1	0	0
1	0	0	1
1	1	1	1

$$\Pr[u \text{ non invertible} \mid u \text{ uniform}] = \frac{1}{2}$$

$$\Pr[u \text{ non invertible} \mid u = a \cdot b] = \underbrace{\frac{1}{2}}_{a \text{ inv}} + \underbrace{\frac{1}{2} \cdot \frac{1}{2}}_{a \text{ non inv}}$$

Needs to make this proba. small
Decrease with the number of components

Uniformity over R_q, q power of 2

$$x \cdot \underbrace{\left(B \mid sB + e \right)}_{\in R_q^{(d+1)\ell \times d \ + \ (d+1)\ell}} + \overbrace{\left(\mathbf{0} \mid y \right)}^{= \text{RERAND}} \approx_s V_{\mathsf{f}}$$

Needs to show :

- LEFT PART: uniformity of $x \cdot B$
- RIGHT PART: noise is Gaussian of parameter independent of that of V_{t-1}, V'_{t-1}: $x \cdot (sB + e + y)$

Our version for cp-mux

Public key: $\left(\mathbf{a}, \mathbf{a} \cdot \mathbf{s} + e\right)$

ReRand $:= \left(r \cdot \mathbf{a} + e', r \cdot \left(\mathbf{a} \cdot \mathbf{s} + e\right) + e''\right)$

cp-mux$([\beta_t]_{\mathbf{s}}, V_t, V_t') := V_t + \mathbf{G}_{\text{rand}}^{-1}\left(V_t' - V_t\right) \cdot [\beta_t] +$ ReRand

cp-mux$([\beta_t]_{\mathbf{s}}, V_t, V_t') \approx_c V_{\beta_t} + V_f$

$\approx_c$ cp-mux$([\beta_t]_{\mathbf{s}}, V_{\beta_t}, V_{\beta_t})$

$\approx_c V_{\beta_t} + \mathbf{G}_{\text{rand}}^{-1}\left(\mathbf{0}\right) \cdot [\beta_t] +$ ReRand

Our version for cp-mux

New proof technique compared to [BdMW16]: reuse ReRand

Unfolding the loop:

$$
\begin{aligned}
& X^{b-\mathbf{a}\cdot\mathbf{s}} + \\
& \quad + \mathsf{G}_{\mathrm{rand}}^{-1}\left(\mathsf{ACC}_0(X^{a_0}-1)\right)\cdot[s_0]_{\mathbf{s}} \\
& \qquad\qquad \vdots \\
& \quad + \mathsf{G}_{\mathrm{rand}}^{-1}\left(\mathsf{ACC}_{n-2}(X^{a_{n-2}}-1)\right)\cdot[s_{n-2}]_{\mathbf{s}} \\
& \quad + \mathsf{G}_{\mathrm{rand}}^{-1}\left(\mathsf{ACC}_{n-1}(X^{a_{n-1}}-1)\right)\cdot[s_{n-1}]_{\mathbf{s}} \\
& \quad + \mathsf{ReRand}
\end{aligned}
$$

Our version for cp-mux

New proof technique compared to [BdMW16]: reuse ReRand

Unfolding the loop:

$$
\begin{aligned}
& X^{b-\mathbf{a}\cdot\mathbf{s}} + \\
& \qquad + \mathsf{G}_{\mathrm{rand}}^{-1}\left(\mathsf{ACC}_0(X^{a_0}-1)\right)\cdot[s_0]_{\mathbf{s}} \\
& \qquad\qquad \vdots \\
& \qquad + \mathsf{G}_{\mathrm{rand}}^{-1}\left(\mathsf{ACC}_{n-2}(X^{a_{n-2}}-1)\right)\cdot[s_{n-2}]_{\mathbf{s}} \\
\text{ReRand} \quad & + \mathsf{G}_{\mathrm{rand}}^{-1}\left(\mathsf{ACC}_{n-1}(X^{a_{n-1}}-1)\right)\cdot[s_{n-1}]_{\mathbf{s}}
\end{aligned}
$$

Our version for cp-mux

New proof technique compared to [BdMW16]: reuse ReRand

Unfolding the loop:

$$
\begin{aligned}
& X^{b-\mathbf{a}\cdot\mathbf{s}} + \\
& \qquad +\mathsf{G}_{\mathrm{rand}}^{-1}\left(\mathsf{ACC}_0(X^{a_0}-1)\right)\cdot[s_0]_{\mathbf{s}} \\
& \qquad\qquad \vdots \\
& \qquad +\mathsf{G}_{\mathrm{rand}}^{-1}\left(\mathsf{ACC}_{n-2}(X^{a_{n-2}}-1)\right)\cdot[s_{n-2}]_{\mathbf{s}} \\
\mathsf{ReRand} \quad & +\mathsf{G}_{\mathrm{rand}}^{-1}\left(\mathbf{0}\right)\cdot[s_{n-1}]_{\mathbf{s}}
\end{aligned}
$$

Our version for cp-mux

New proof technique compared to [BdMW16]: reuse ReRand

Unfolding the loop:

$$
\begin{aligned}
& X^{b-\mathbf{a}\cdot\mathbf{s}} + \\
& \quad + \mathsf{G}_{\mathrm{rand}}^{-1}\left(\mathsf{ACC}_0(X^{a_0}-1)\right)\cdot[s_0]_{\mathbf{s}} \\
& \qquad \vdots \\
\mathsf{ReRand} \quad & + \mathsf{G}_{\mathrm{rand}}^{-1}\left(\mathsf{ACC}_{n-2}(X^{a_{n-2}}-1)\right)\cdot[s_{n-2}]_{\mathbf{s}} \\
& \quad + \mathsf{G}_{\mathrm{rand}}^{-1}\left(\mathbf{0}\right)\cdot[s_{n-1}]_{\mathbf{s}}
\end{aligned}
$$

Our version for cp-mux

New proof technique compared to [BdMW16]: reuse ReRand

Unfolding the loop:

$$
\begin{aligned}
& X^{b-\mathbf{a}\cdot\mathbf{s}} + \\
& \qquad +\mathsf{G}_{\mathrm{rand}}^{-1}\left(\mathsf{ACC}_0(X^{a_0}-1)\right)\cdot[s_0]_{\mathbf{s}} \\
& \qquad\qquad \vdots \\
\mathsf{ReRand} \quad & +\mathsf{G}_{\mathrm{rand}}^{-1}\left(\mathbf{0}\right)\cdot[s_{n-2}]_{\mathbf{s}} \\
& \qquad +\mathsf{G}_{\mathrm{rand}}^{-1}\left(\mathbf{0}\right)\cdot[s_{n-1}]_{\mathbf{s}}
\end{aligned}
$$

Our version for cp-mux

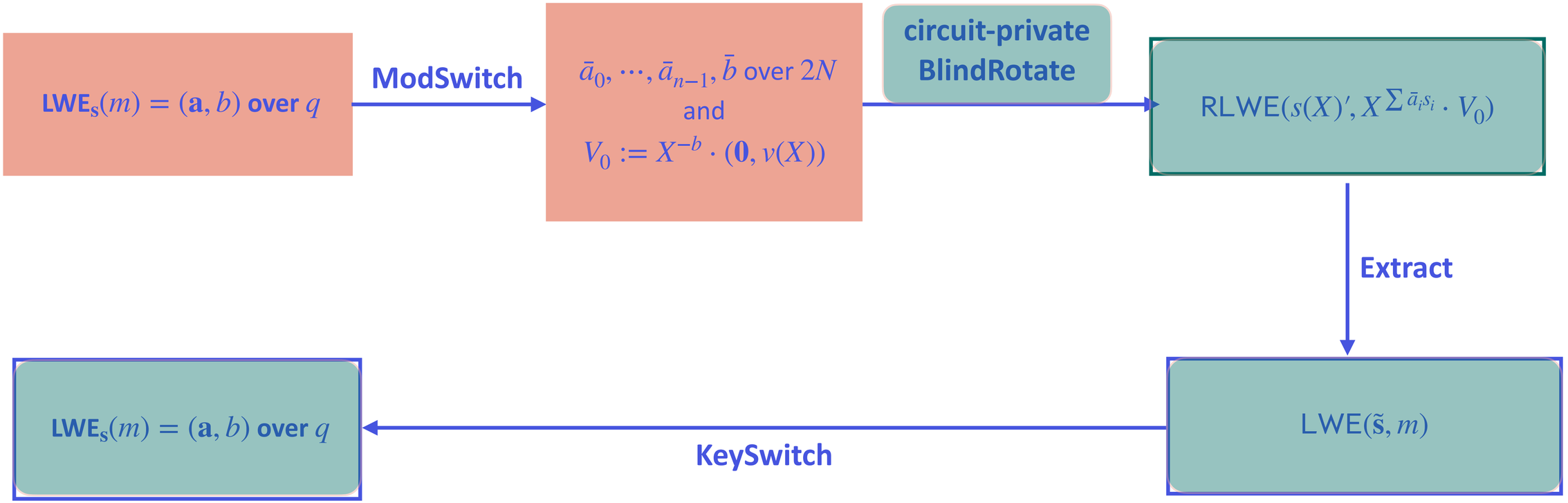
New proof technique compared to [BdMW16]: reuse ReRand

Unfolding the loop:

$$
\begin{aligned}
& X^{b-\mathbf{a}\cdot\mathbf{s}} + \\
\mathsf{ReRand}\ & + \mathsf{G}_{\mathrm{rand}}^{-1}(\mathbf{0}) \cdot [s_0]_{\mathbf{s}} \\
& \vdots \\
& + \mathsf{G}_{\mathrm{rand}}^{-1}(\mathbf{0}) \cdot [s_{n-2}]_{\mathbf{s}} \\
& + \mathsf{G}_{\mathrm{rand}}^{-1}(\mathbf{0}) \cdot [s_{n-1}]_{\mathbf{s}}
\end{aligned}
$$

Sanitized TFHE bootstrapping

FHE Sanitization - our approach



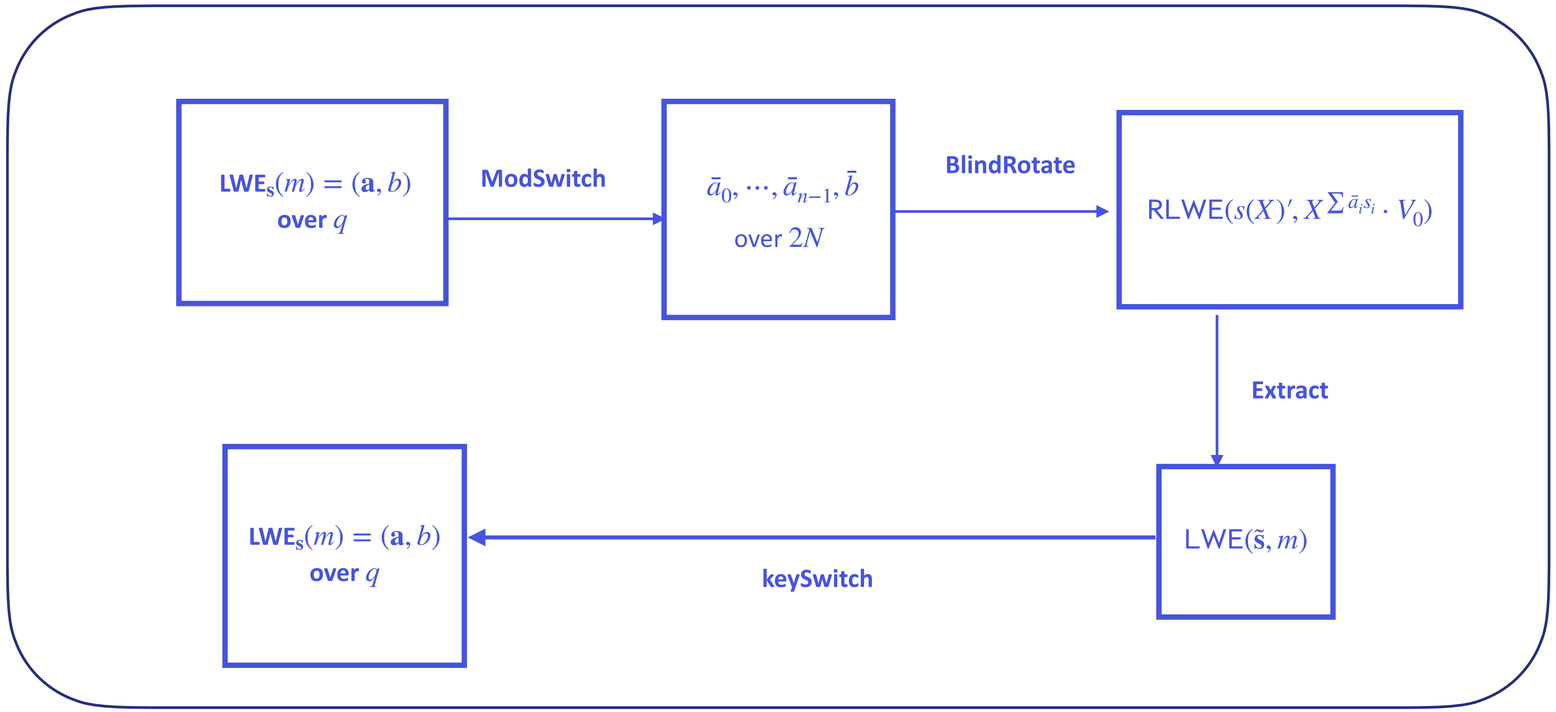
Parameters

- LWE: $n = 538$, $\vartheta_{\mathsf{lwe}} = 2^{-27.2}$, $B_{\mathsf{ks}} = 4$, $\ell_{\mathsf{ks}} = 7$
- RLWE: $N = 1024$, $\vartheta_{\mathsf{rlwe}} = 2^{-67.6}$, $\vartheta_{\mathsf{pk}} = 2^{-65.6}$, $2^{-49.5}$
- RGSW: $B_{\mathsf{G}} = 32$, $\ell_{\mathbf{G}} = 7$, $\vartheta_{\mathbf{G}} = 2^{-39.4}$
- With 100 bits of security:

Bootstrap -	Sanitize w/ pre-computation	Sanitize w/o pre-computation	PkEnc(PK) -
0.42 s	0.52 s	7.5 s	1 ms

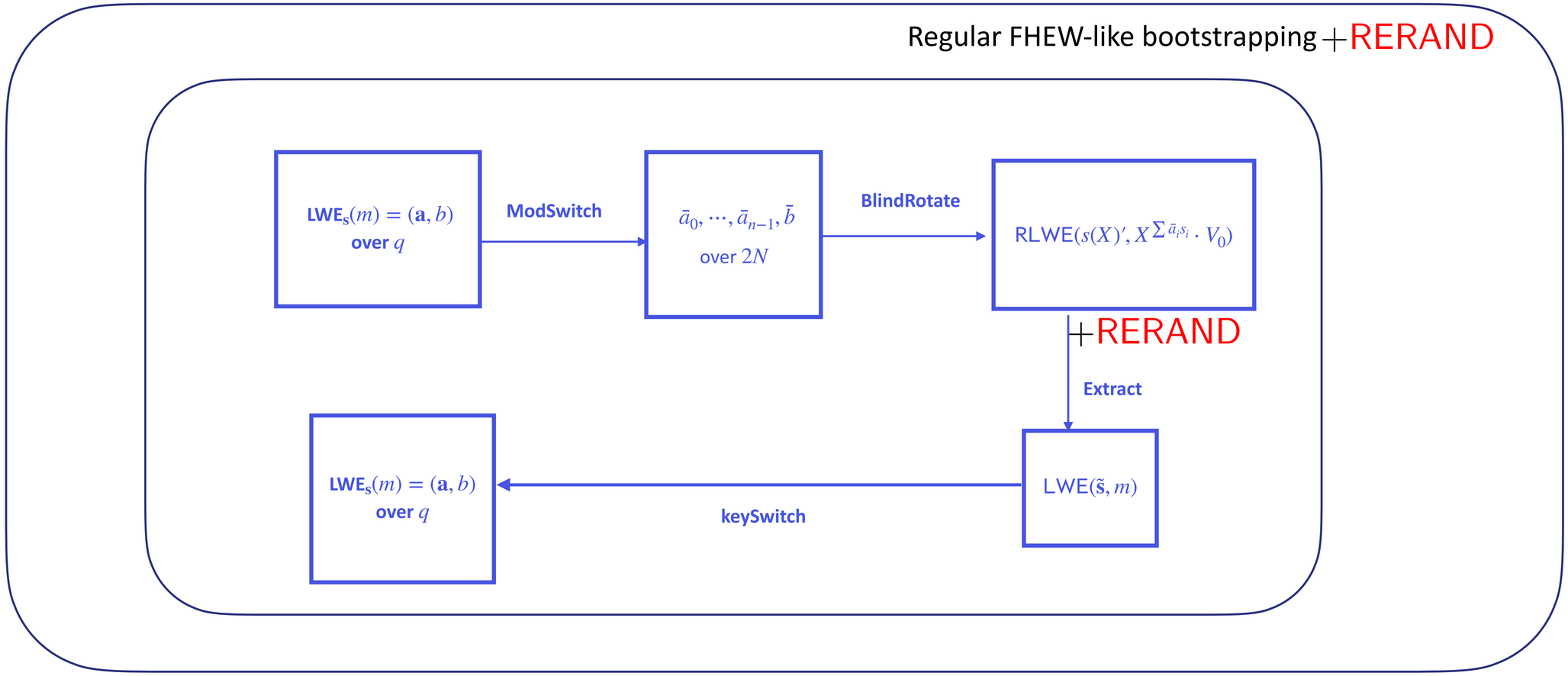
FHE Sanitization - [DS16] approach

Regular FHEW-like bootstrapping



With our parameters, $\sigma_{\text{boot}} \approx 2^{-17.4}$

FHE Sanitization - [DS16] approach

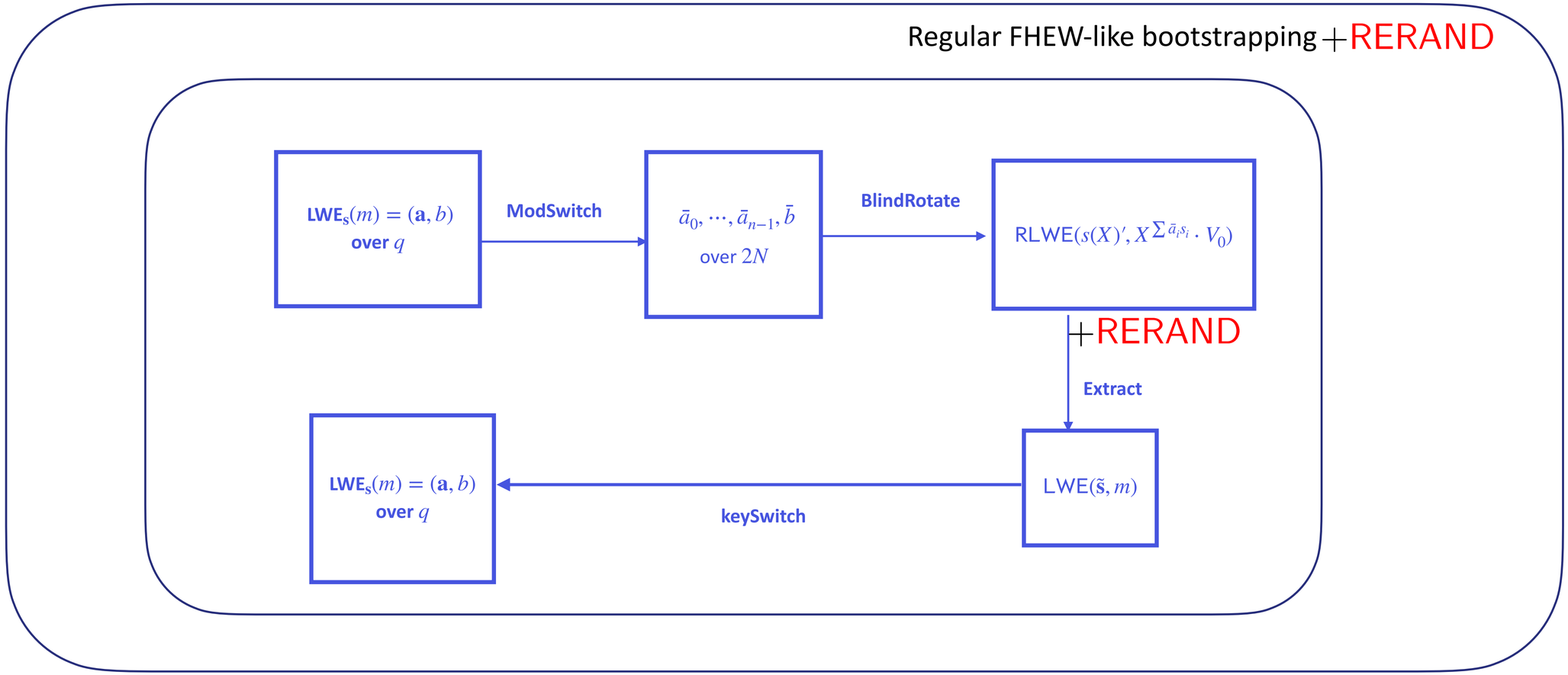


One [DS16] cycle

$$\sigma_{\text{boot}} + \sigma_{\text{PkEnc}(PK_s)} + |\text{soaking noise}| + \sigma_{\text{KS}} \approx 0.016$$

One [DS16] cycle → distance of 2^{-14} between two ciphertexts

FHE Sanitization - [DS16] approach



One [DS16] cycle

≈ 6 such cycles

Conclusion

- [BdMW16] over rings: multi-hop circuit privacy for branching programs for RGSW without circular security;
- Improved induction for [BdMW16], improved noise propagation and reduced parameter improve efficiency compared to [BdMW16], even on GSW;
- Alternative Rerand implementation:
  greatly reduces the size of public key, applicable to RLWE and LWE;
- Sanitization with one round of bootstrapping:
  works with power-of-two modulus, adaptable to other modulus;

Thank you !