Plug-and-Play Sanitization for TFHE

Florian Bourse and Malika Izabachène

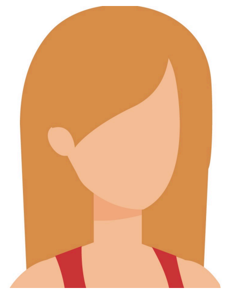
(Independent Scholar) (Cosmian)

Séminaire C2, 7 octobre 2022

Privacy preserving diagnosis

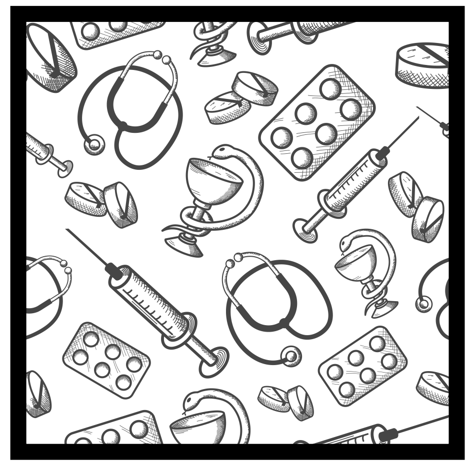
Alice

Cloud provider



encrypted data

encrypted diagnosis



sk, pk

pk

Fully Homomorphic Encryption (FHE)

Alice

Cloud provider

$[m_1],\ [m_2],\ \cdots,\ [m_k] \longrightarrow$

$c = \mathsf{Eval}\left(\mathsf{evk}, f, ([m_1], \cdots, [m_k])\right)$

$\mathsf{Dec}(\mathsf{sk}, c) = f(m_1, m_2, \cdots, m_k) \longleftarrow$

Message privacy: Alice's messages are kept unknown to the cloud provider.

Fully Homomorphic Encryption (FHE)

Alice

Cloud provider

$[m_1],\ [m_2],\ \cdots,\ [m_k] \longrightarrow$

$c = \mathsf{Eval}\left(\mathsf{evk}, f, ([m_1], \cdots, [m_k])\right)$

$\mathsf{Dec}(\mathsf{sk}, c) = f(m_1, m_2, \cdots, m_k) \longleftarrow$

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 and needs to be protected.

Outline

- Definitions
- Circuit privacy and sanitization
- Previous approaches
- TFHE bootstrapping
- Sanitizing TFHE

Fully Homomorphic Encryption

- KeyGen(1^λ): evk, sk
- Enc(sk, μ): c (also, written as $[\mu]$ in this talk)
- Dec(sk, c) : μ
- Eval(evk, f, $[\mu_1], \cdots, [\mu_k]$)= c

Fully Homomorphic Encryption

- KeyGen(1^λ): evk, sk

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

- Dec(sk, c) : μ

- Eval(evk, f, $[\mu_1], \cdots, [\mu_k]$)= c

Correctness:

1) if $c = \mathsf{Enc}(\mathsf{sk}, \mu)$, $\mathsf{Dec}(\mathsf{sk}, c) = \mu$;
2) if $c = \mathsf{Eval}(\mathsf{evk}, f, [\mu_1], .. [\mu_k]))$, $\mathsf{Dec}(\mathsf{sk}, c) = f(\mu_1, \cdots, \mu_k))$

FHE ciphertexts (noisy ciphertexts)

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

$$\cdots \longrightarrow ?$$

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

Noise management

Level Mode

- circuit depth d known in advance
- set the parameters relatively to d
- bounded number of operations

Bootstrapped Mode

- depth circuit can set dynamically
- unlimited depth
- flexibility: bootstrap (set of) gate(s) by gate.

Bootstrapping, [Gentry09]

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

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

Bootstrapping, [Gentry09]

Let [m] = Encrypt(sk,m) , Decrypt(sk, [m]) = m

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

Let [sk] = Encrypt(sk,sk)

Bootstrapping, [Gentry09]

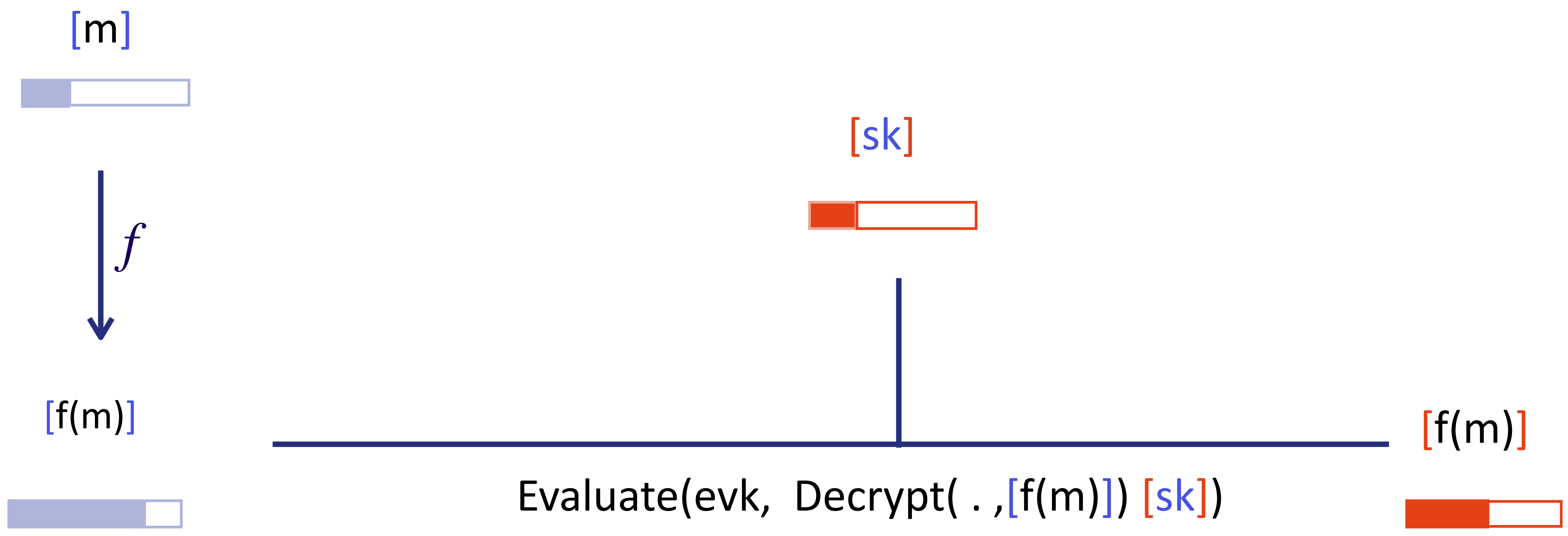
By definition $[m] = \mathrm{Encrypt}(sk, m)$, $\mathrm{Decrypt}(sk, [m]) = m$

Define $\mathrm{Decrypt}_{[m]}(\,.\,) = \mathrm{Decrypt}(\,.,\,[m])$

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

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

Bootstrapping noise growth, [Gentry09]



GSW encryption

[GSW13] Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. Crypto2013.

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

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

Decomposition

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

Addition

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

Product

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

GSW encryption

$$C_1 = (\mathbf{A}_1 \mid \mathbf{s}\mathbf{A}_1 + e_1) + \mu_1 \mathbf{G} \qquad\qquad C_2 = (\mathbf{A}_2 \mid \mathbf{s}\mathbf{A}_2 + e_2) + \mu_2 \mathbf{G}$$

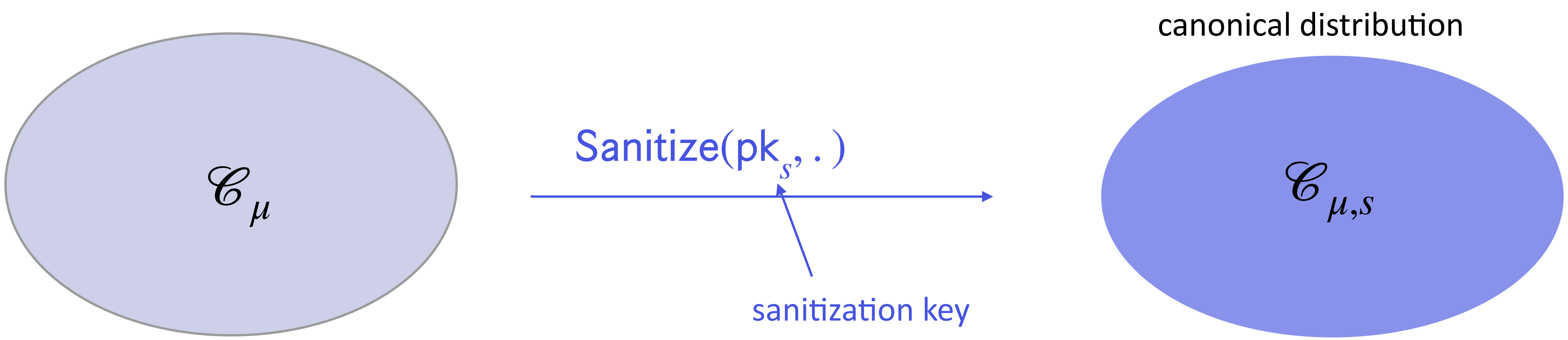
Given **s** and a ciphertext C, one can distinguish $C_1 + C_2$ from $\mathbf{G}^{-1}(C_1) \cdot C_1$;

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

Sanitization

$\mathscr{C}_{\mu}$: all the ciphertexts that decrypt to μ

$\mathscr{C}_{\mu,s}$: $C,\ C' \in \mathscr{C}_{\mu,s}\quad \Delta((\mathsf{pk}_s, C, (\mathsf{pk}), \mathsf{sk})), (\mathsf{pk}_s, C', (\mathsf{pk}, \mathsf{sk}))) < \mathsf{negl}(n)$



Sanitization

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

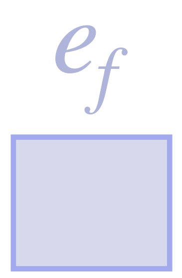
A Sanitize algorithm is a PPT algorithm s.t.:

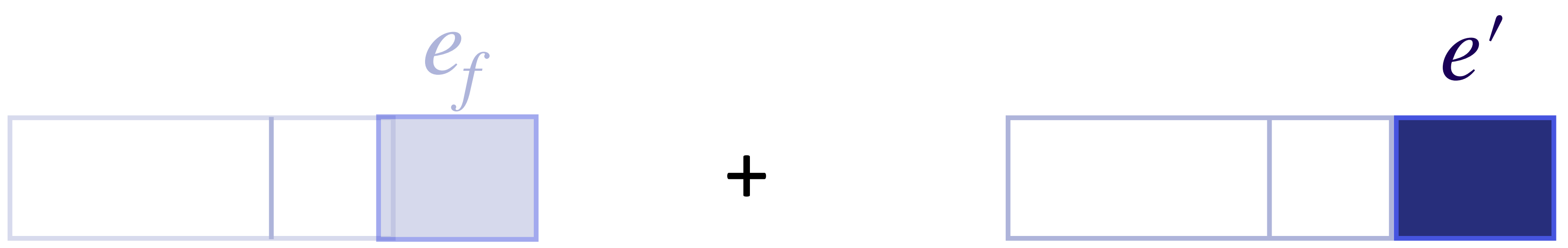
1) For all ciphertext $C \in \mathscr{C}_\mu$, $\mathsf{Sanitize}(\mathsf{pk}_s, C) \in \mathscr{C}_\mu$

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

Noise flooding, [Gentry09]

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



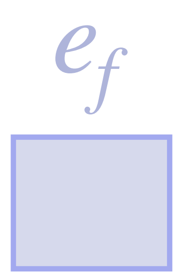


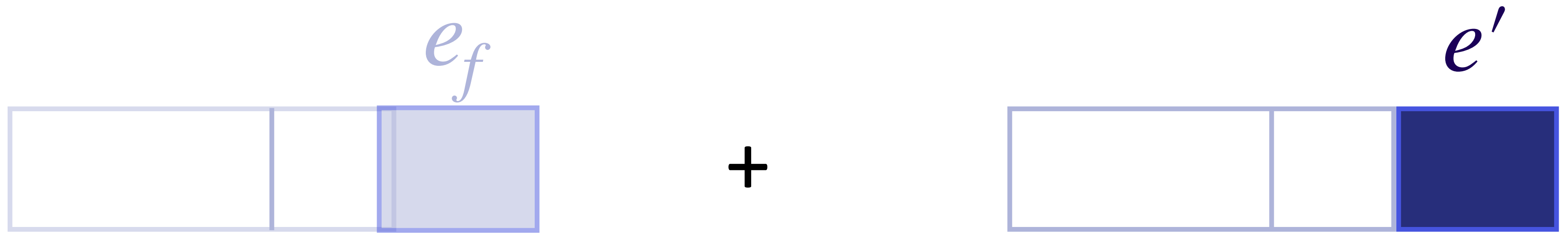
$$e' \gg e_f$$

- Add super-polynomial noise —> large modulus
- FHE must support large noise
- strong LWE assumption

Noise flooding, [Gentry09]

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



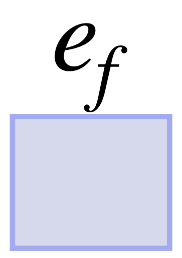


$$e' \gg e_f$$

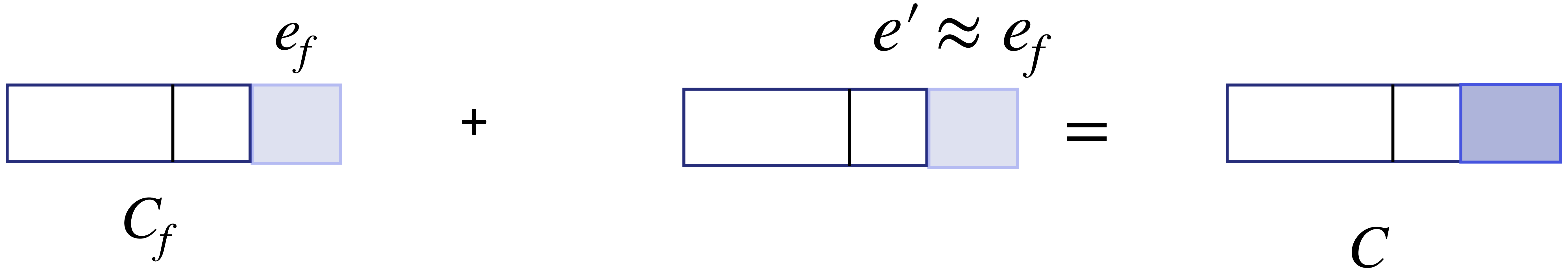
+ **Randomization step**

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

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



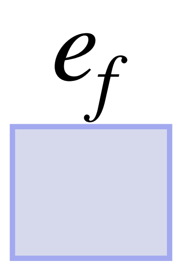
① **Randomization step :**



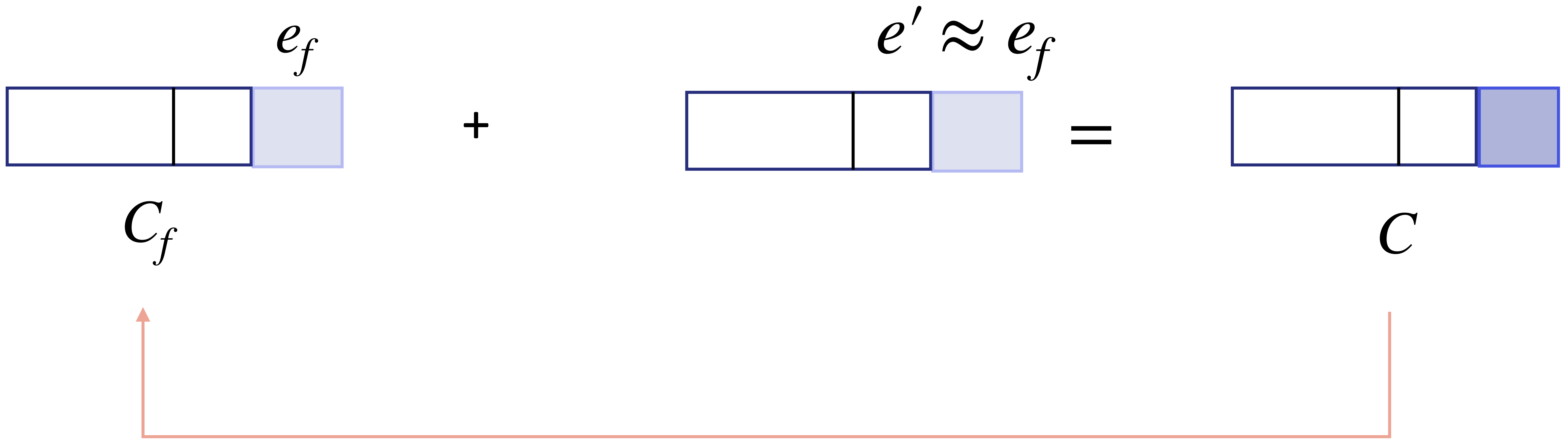
$$\Pr[C \in \mathscr{C}_{\mu,S}] = \mathsf{cste}$$

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

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



1 **Randomization step :**



$$\Pr[C \in \mathscr{C}_{\mu,S}] = \text{cste}$$

2 **Refreshing step (bootstrapping)**

3 **repeat**

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

Circuit-private Branching Program, [BdMW16]

GSW encryption scheme

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

Branching Program evaluation

Circuit-private Branching Program, [BdMW16]

GSW encryption scheme

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

Branching Program evaluation

C_f e_f $+$ e' $+$ $\mathbf{G}^{-1}(\,.\,)$

Our approach

1. One invocation of the bootstrapping
2. Branching Program evaluation inside the bootstrapping ([BdMW16] over rings)
3. Analyzing the noise
4. Randomization amplification

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

Noisy encoding

- Message space = $\mathbb{Z}_t, \quad t = 2$
- Ciphertext space = $\mathbb{Z}_q, \quad q = 2^3$
- message $m \in \mathbb{Z}_t \quad \Delta = \lfloor \frac{q}{t} \rceil$

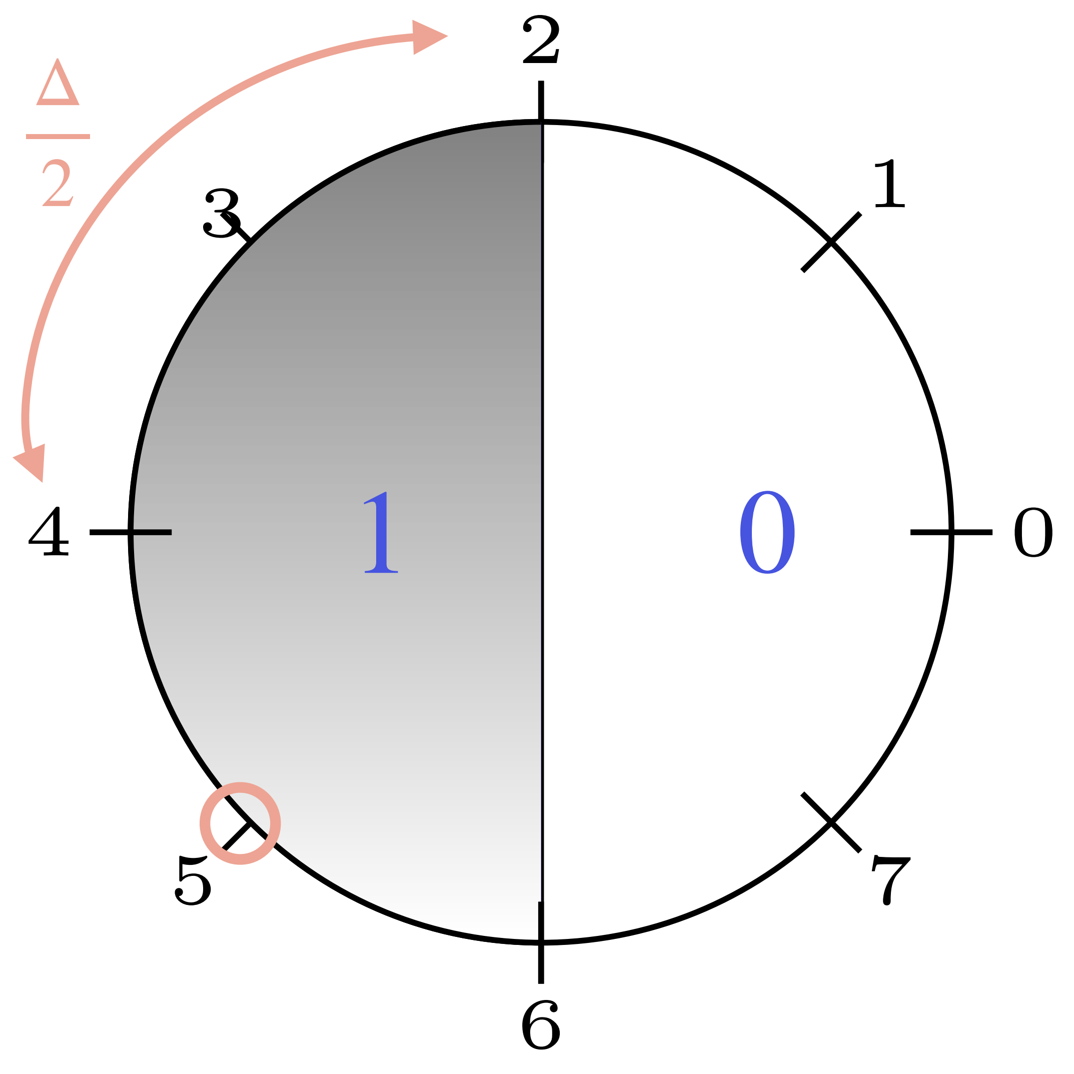
encode*

encoding of m : $\mu = \Delta \cdot 1 = 4$

noisy encoding of m : $\mu^\star = \mu + e = 4 + 1 = 5$

decode

decode $\bar{\mu}^\star$ by rescaling and rounding: $\mu = \lfloor \frac{\mu^\star}{\Delta} \rceil$



Noisy encoding

Other possible message space :
- discretised torus elements
- {0,1} message space, ...

- Message space = $\mathbb{Z}_t, \quad t = 2$
- Ciphertext space = $\mathbb{Z}_q, \quad q = 2^3$
- message $m \in \mathbb{Z}_t \quad \Delta = \lfloor \frac{q}{t} \rceil$

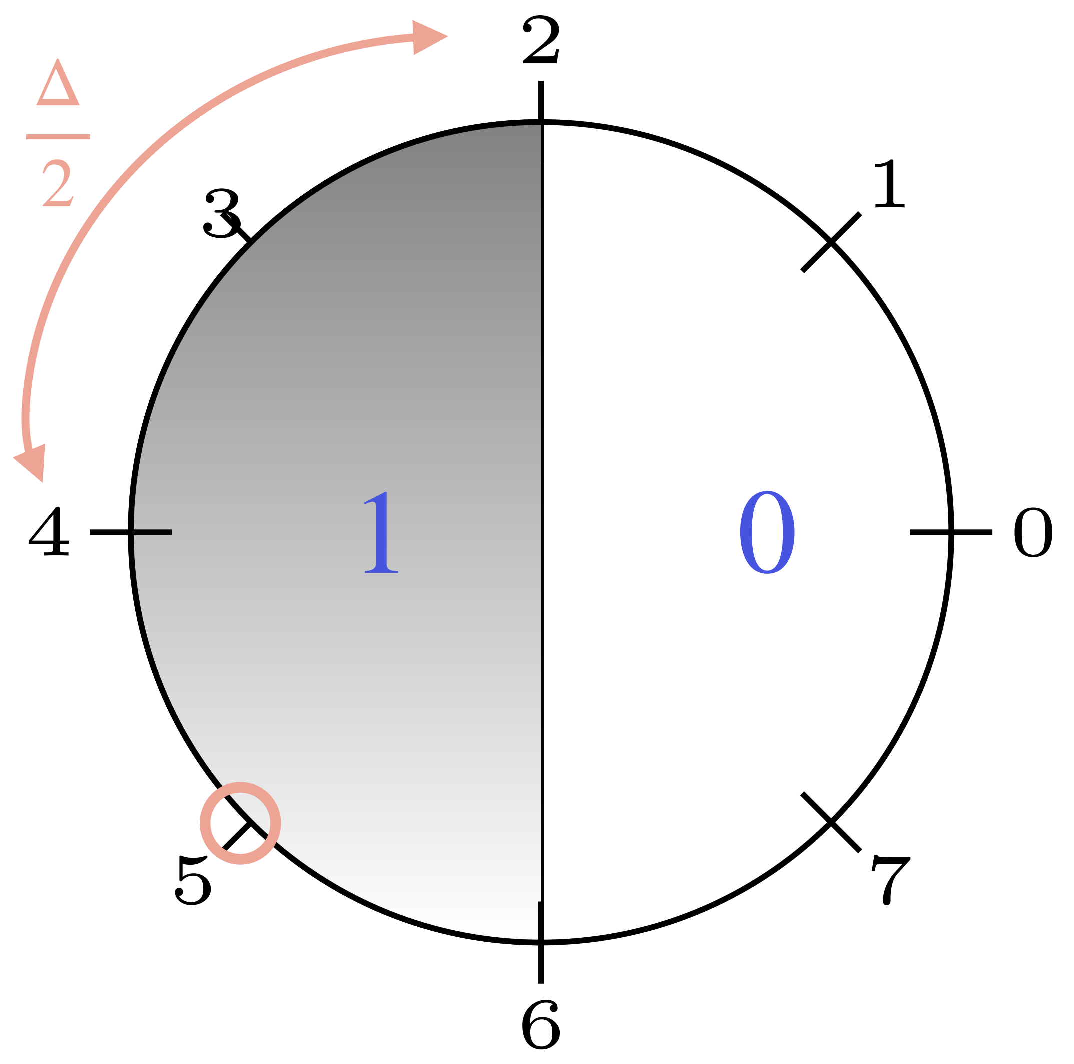
encode*

encoding of m : $\mu = \Delta \cdot 1 = 4$

noisy encoding of m : $\mu^\star = \mu + e = 4 + 1 = 5$

decode

decode $\bar{\mu}^\star$ by rescaling and rounding: $\mu = \lfloor \frac{\mu^\star}{\Delta} \rceil$



LWE encryption

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

Encryption

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

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

where $\mathbf{a}$ is random in $\mathbb{Z}_q^n$, 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

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

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

$a(X)$

a_{N-1}	$\cdots$	a_0

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

b_{N-1}	$\cdots$	b_0

Decryption

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

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

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

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

RingGSW encryption

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

Addition

multiplication by a small constant

internal multiplication

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

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

RingGSW encryption

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

Addition

multiplication by a small constant

internal multiplication

external multiplication

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

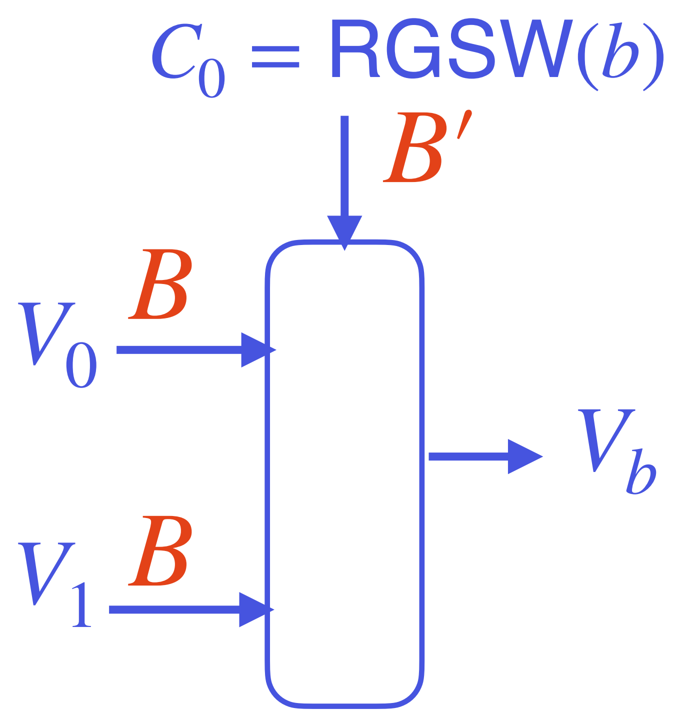
$$\overbrace{\text{RGSW}(s(X), m_1(X))}^{C_1} \odot \overbrace{\text{RLWE}(s(X), m_2(X))}^{C_2} = \mathbf{G}^{-1}(C_2)\cdot C_1$$

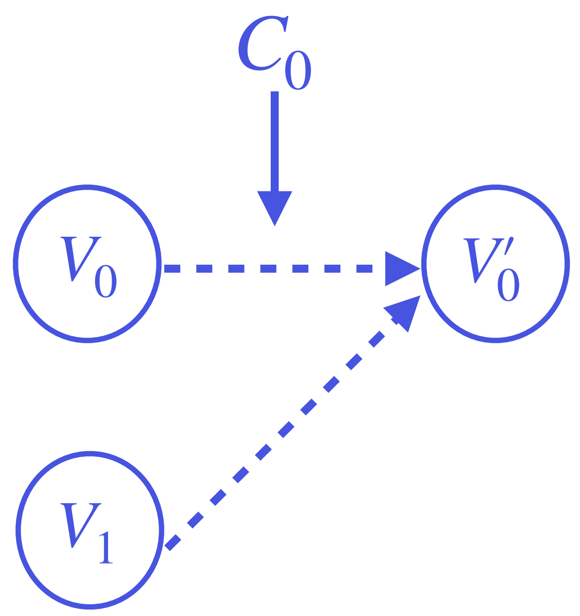
$$\ldots. = \text{RLWE}(s(X), m_1(X)\cdot m_2(X))$$

Controlled MUX gate

$$\mathsf{MUX}(b_0, v_0, v_0') = v_0' + b_0(v_0 - v_0'), \quad b_0 \in \{0, 1\}$$

Controlled MUX gate



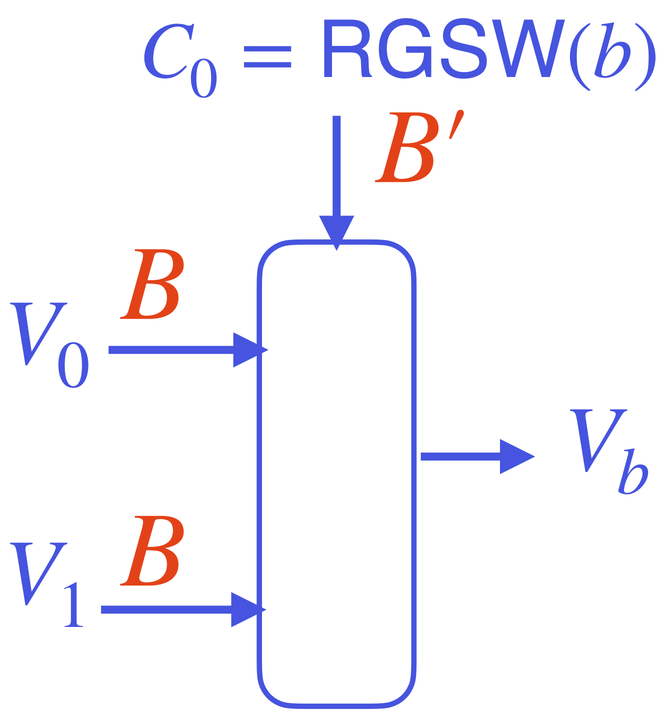


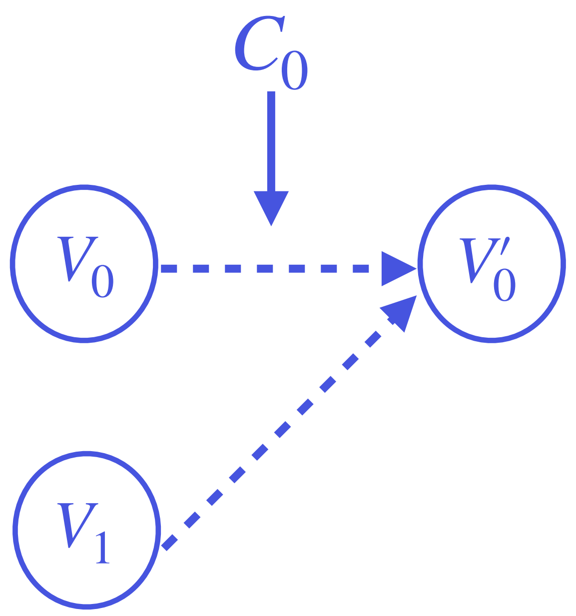
$$\mathsf{MUX}(b_0, v_0, v_0') = v_0' + b_0(v_0 - v_0'), \quad b_0 \in \{0,1\}$$

$$V_t := \mathsf{RLWE}(s(X), v_t)$$

$$V_1 = V_0' + \mathbf{G}^{-1}(V_0 - V_0') \cdot C_0$$

Controlled MUX gate

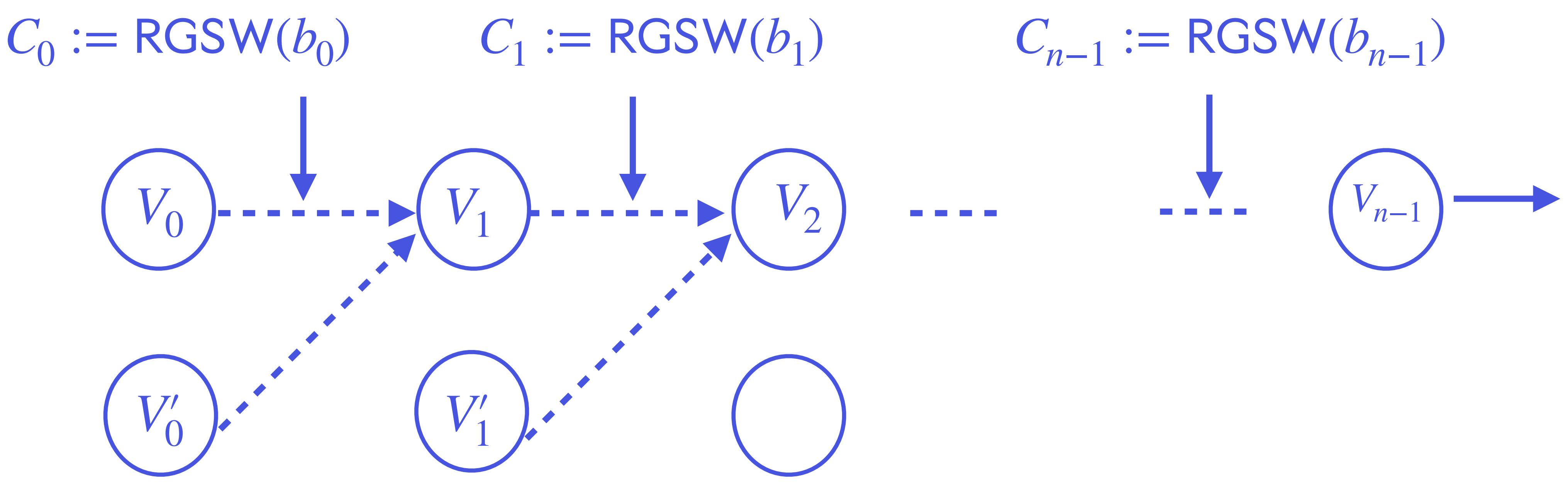




$$\mathsf{MUX}(b_0, v_0, v'_0) = v'_0 + b_0(v_0 - v'_0), \quad b_0 \in \{0,1\}$$

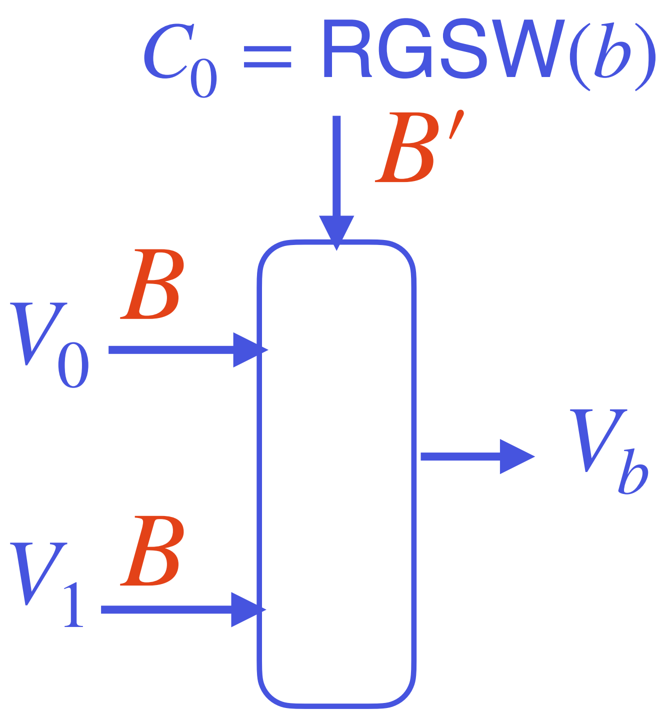
$$V_t := \mathsf{RLWE}(s(X), v_t)$$

$$V_1 = V'_0 + \mathbf{G}^{-1}(V_0 - V'_0) \cdot C_0$$

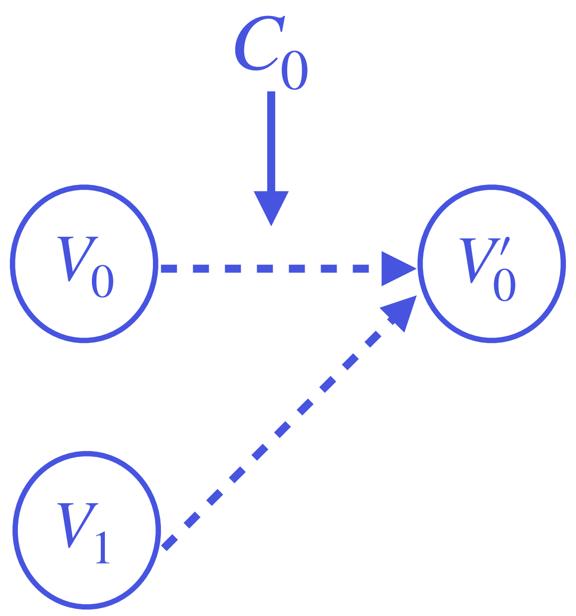


Controlled MUX gate

$$\mathsf{MUX}(b_0, v_0, v_0') = v_0' + b_0(v_0 - v_0'), \quad b_0 \in \{0,1\}$$



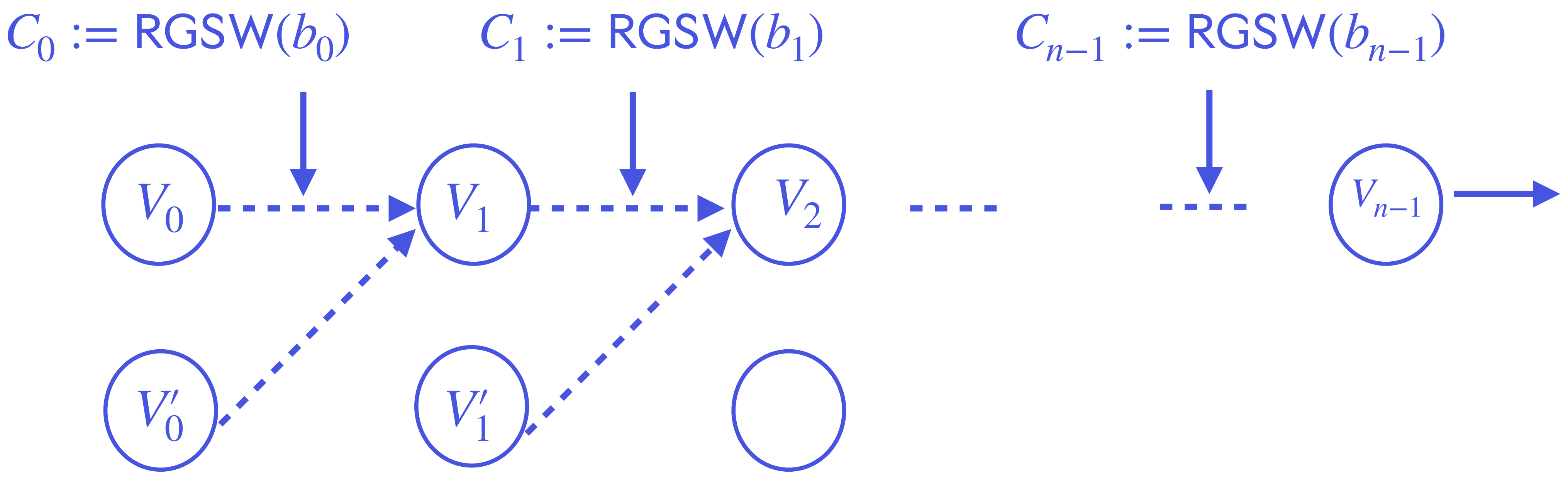
$$V_t := \mathsf{RLWE}(s(X), v_t)$$



$$V_1 = V_0' + \mathbf{G}^{-1}(V_0 - V_0') \cdot C_0$$

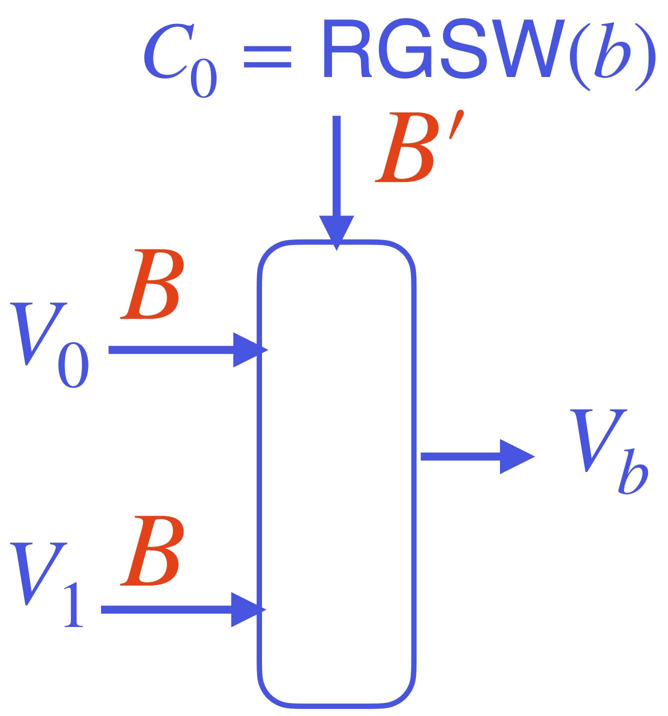
$$V_t = V_{t-1} + C_{t-1} \odot (V_{t-1} - V_{t-1}')$$

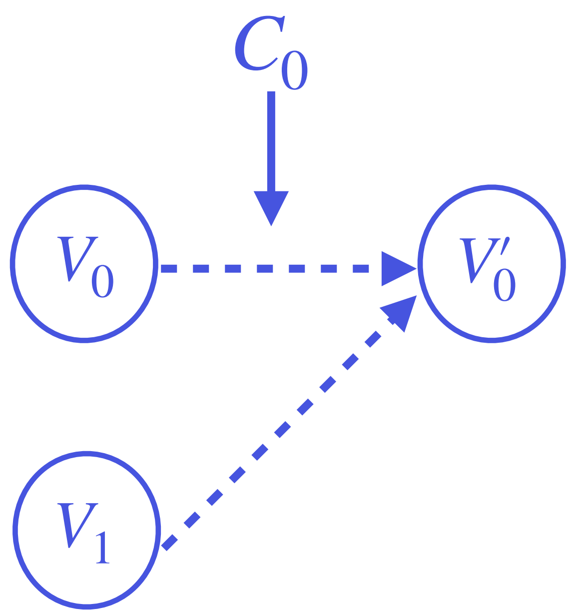
$$= V_{t-1} + \mathbf{G}^{-1}(V_{t-1} - V_{t-1}') \cdot C_{t-1}$$



Controlled MUX gate

$$\mathsf{MUX}(b_0, v_0, v_0') = v_0' + b_0(v_0 - v_0'), \quad b_0 \in \{0,1\}$$

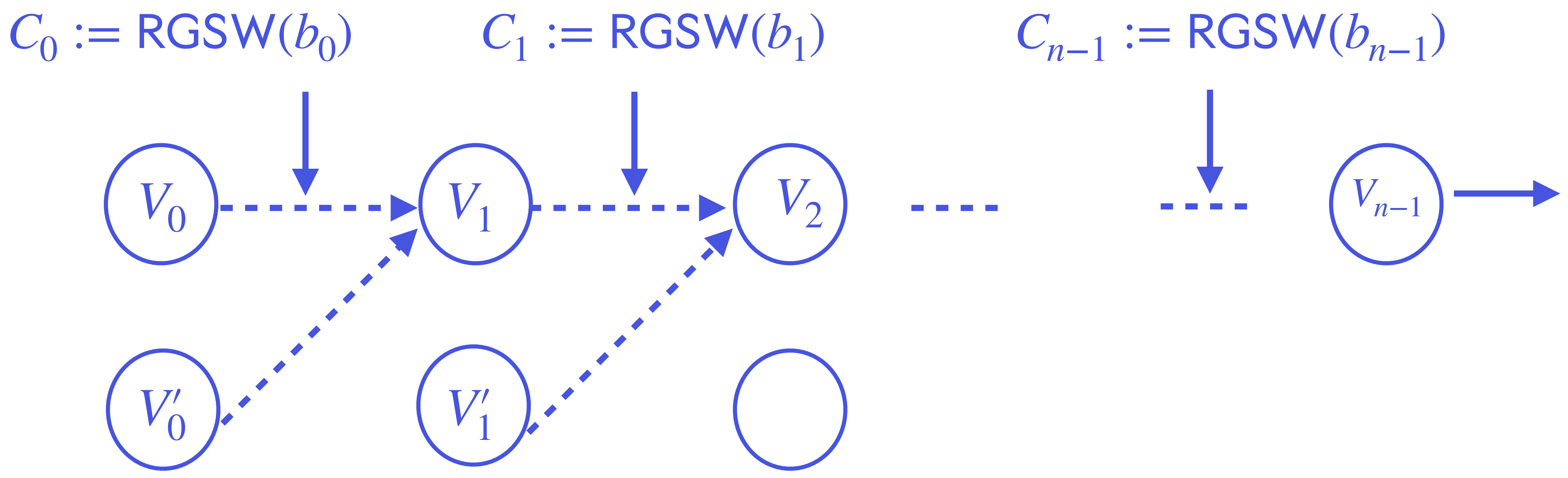




$$V_t := \mathsf{RLWE}(s(X), v_t)$$

$$V_1 = V_0' + \mathbf{G}^{-1}(V_0 - V_0') \cdot C_0$$

$$\begin{aligned} V_t &= V_{t-1} + C_{t-1} \odot (V_{t-1} - V'_{t-1}) \\ &= V_{t-1} + \mathbf{G}^{-1}(V_{t-1} - V'_{t-1}) \cdot C_{t-1} \end{aligned}$$



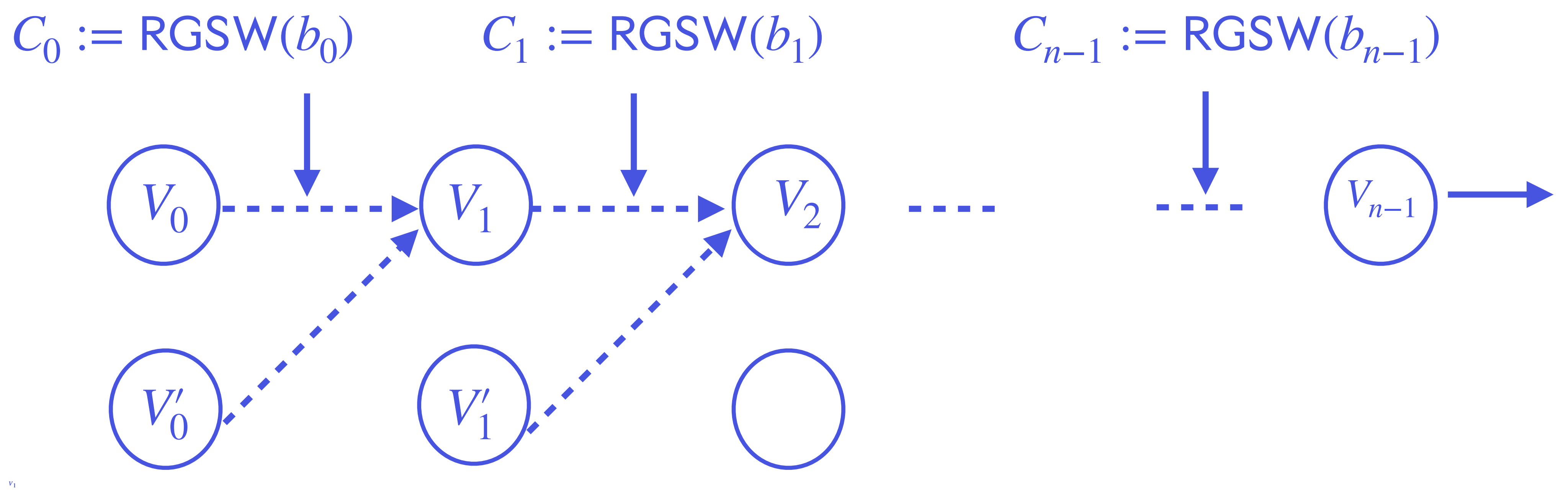
Circuit-private BP-like evaluation, [BdMW16]

$$\mathsf{MUX}(b, v_1, v_0) = v_0 + b_0(v_1 - v_0), \quad b \in \{0,1\}$$

$$V_1 = V_0' + \mathbf{G}^{-1}(V_0 - V_0') \cdot C_0$$

Using [BdMW16], V_1 is now:

$$V_1 = V_0' + \mathbf{G}^{-1}(V_0 - V_0') \cdot C_0 + \left(\ \mathbf{0} \mid y \ \right)$$



Circuit-private BP-like evaluation, [BdMW16]

$$\mathsf{MUX}(b, v_1, v_0) = v_0 + b_0(v_1 - v_0), \quad b \in \{0,1\}$$

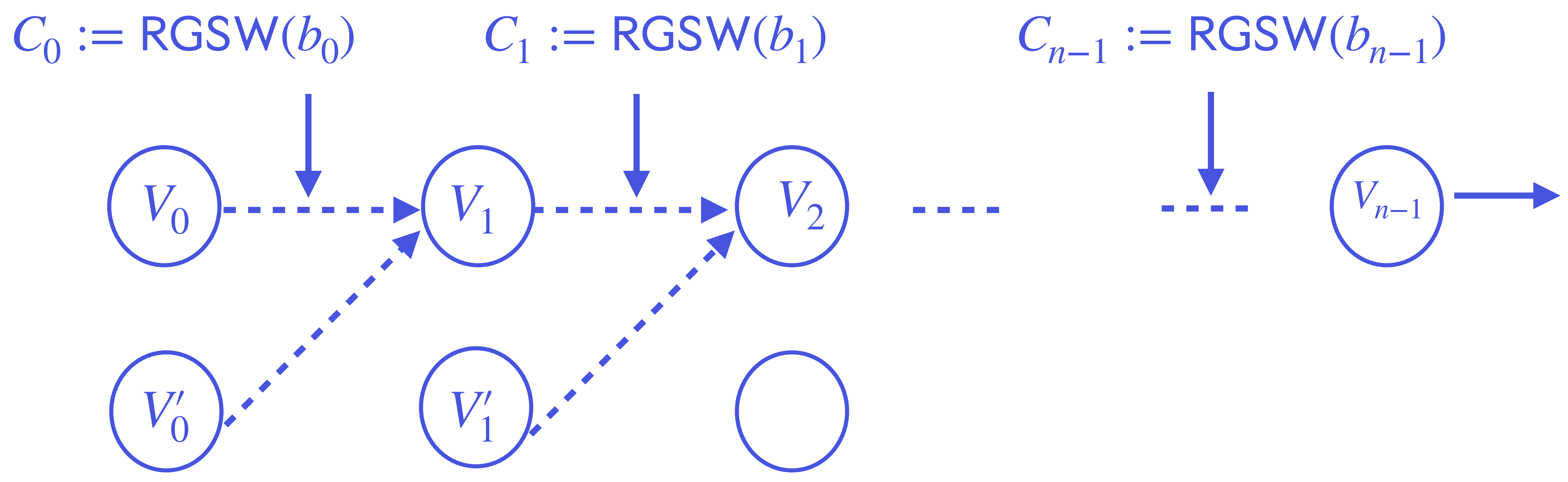
$$V_1 = V_0' + \mathbf{G}^{-1}(V_0 - V_0') \cdot C_0$$

Using [BdMW16], V_1 is now:

$$V_1 = V_0' + \mathbf{G}^{-1}(V_0 - V_0') \cdot C_0 + (\ \mathbf{0} \mid y\ )$$

$$= V_{b_0} + \mathbf{G}^{-1}(V_0 - V_1) \cdot \mathsf{RGSW}(0) + (\ \mathbf{0} \mid y\ )$$

fresh encryption of 0



Circuit-private BP-like evaluation, [BdMW16]

$$\mathsf{MUX}(b, v_1, v_0) = v_0 + b_0(v_1 - v_0), \quad b \in \{0, 1\}$$

$$V_1 = V_0' + \mathbf{G}^{-1}(V_0 - V_0') \cdot C_0$$

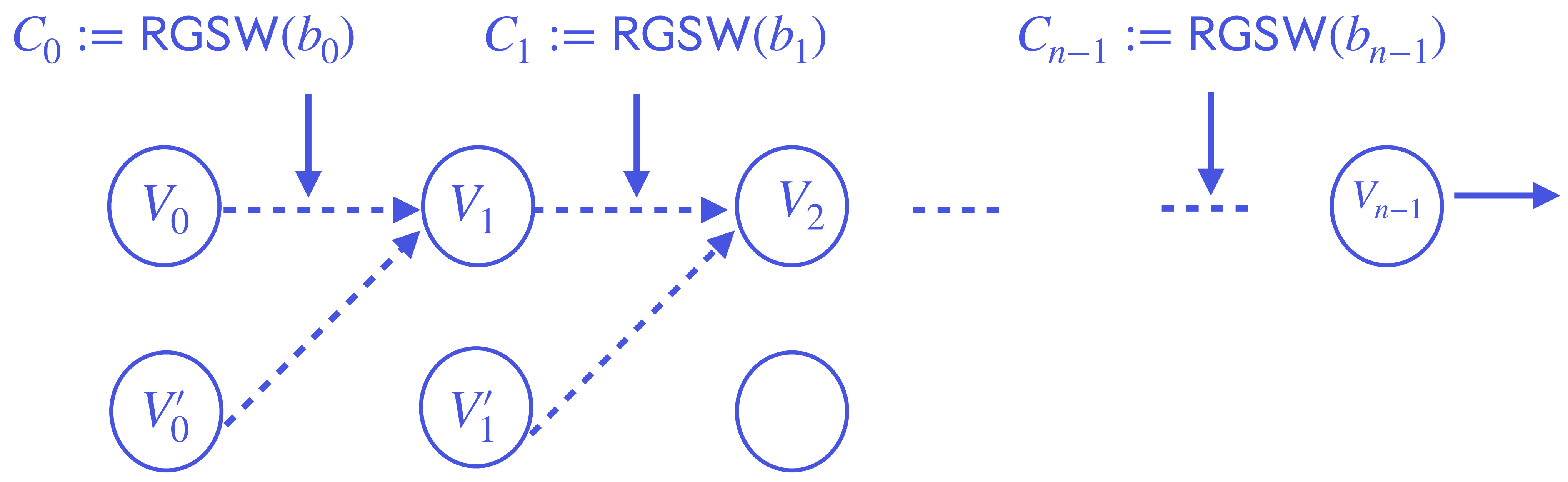
Using [BdMW16], V_1 is now:

$$V_1 = V_0' + \mathbf{G}^{-1}(V_0 - V_0') \cdot C_0 + (\ \mathbf{0} \mid y\ )$$

$$= V_{b_0} + \mathbf{G}^{-1}(V_0 - V_1) \cdot \mathsf{RGSW}(0) + (\ \mathbf{0} \mid y\ )$$

$$\approx_s V_{b_0} + C$$

fresh encryption of 0



TFHE bootstrapping building blocks

TFHE building blocks

Extraction

RLWE $\to$ LWE

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

TFHE building blocks

Extraction

RLWE $\rightarrow$ LWE

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

Blind rotation

RGSW $\times$ RLWE $\rightarrow$ RLWE

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

m_0	m_1	m_2	...	m_{N-2}	m_{N-2}	m_{N-1}

$\times X^i$

m_i	m_{i+1}	...	m_{N-1}	$-m_0$	...	$-m_{i-1}$

TFHE building blocks

Extraction

RLWE $\rightarrow$ LWE

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

Blind rotation

RGSW $\times$ RLWE $\rightarrow$ RLWE

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

m_0	m_1	m_2	...	m_{N-2}	m_{N-2}	m_{N-1}

$$\xrightarrow{\times X^i}$$

m_i	m_{i+1}	...	m_{N-1}	$-m_0$	...	$-m_{i-1}$

Keyswitching

LWE $\times$ LWE $\rightarrow$ LWE

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

Also, $\mathsf{LWE}_{\mathbf{s}}(m) \longrightarrow \mathsf{LWE}_{\mathbf{s}'}\big(f(m)\big)$

bound on the noise induced by f:

$\exists k$ s.t. $\forall x, y: \quad |f(x) - f(y)| < k|x - y|$

TFHE bootstrapping

input: an LWE ciphertext $\mathsf{LWE}_{\mathbf{s}}(m) = (\mathbf{a} = (a_0, \cdots, a_{n-1}), b)$, $\mathsf{BK} = \mathsf{RGSW}(s'(X), s_i)$, $\mathsf{KSK}_{\mathbf{s}' \to \mathbf{s}}$

output: an LWE cipherext with controlled noise (independent of the input noise)

Goal : evaluate $\lfloor b - \mathbf{as} \rceil$ homomorphically

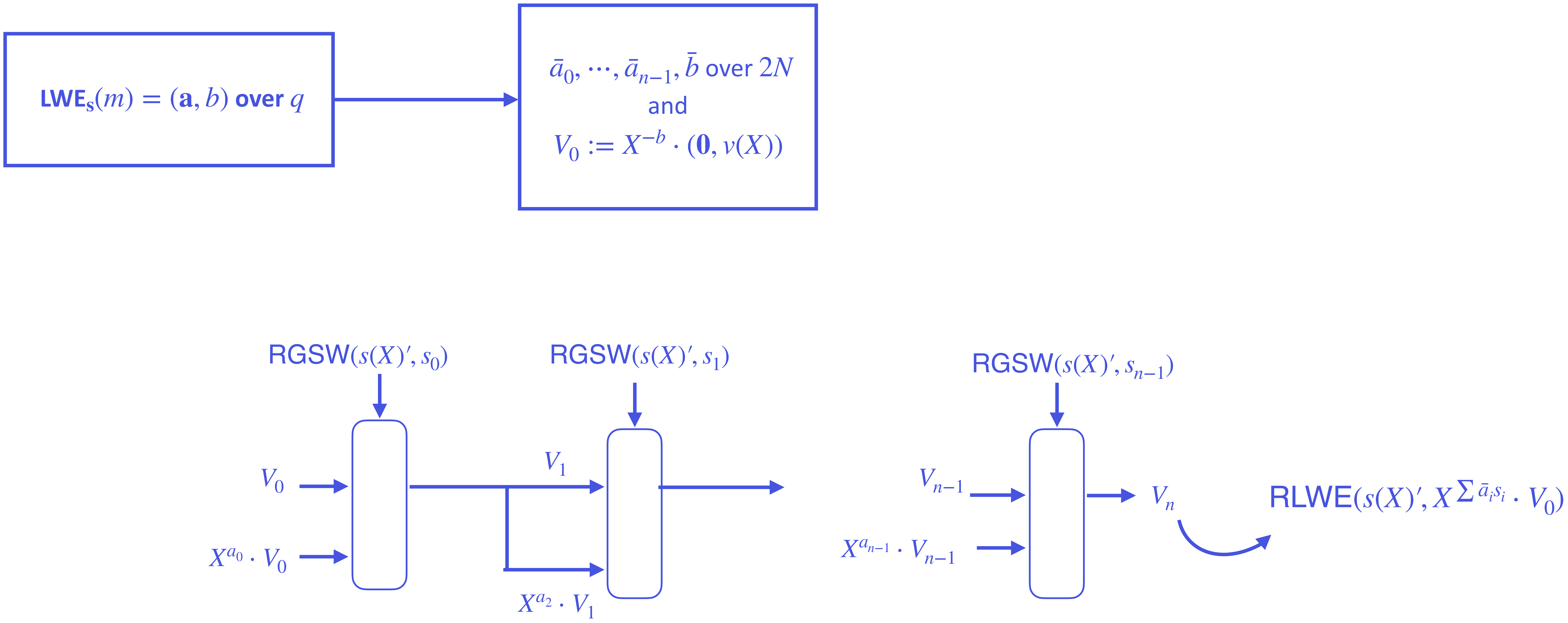
TFHE bootstrapping

LWE$_{\mathbf{s}}(m) = (\mathbf{a}, b)$ over q $\longrightarrow$ $\bar{a}_0, \cdots, \bar{a}_{n-1}, \bar{b}$ over $2N$

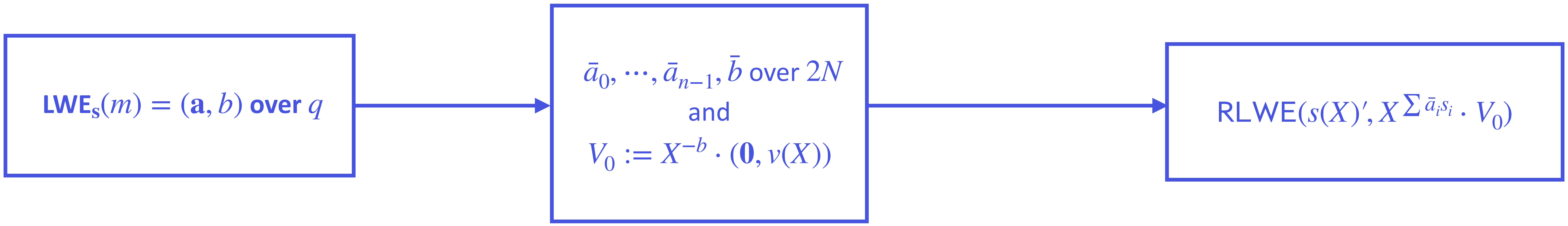
TFHE bootstrapping

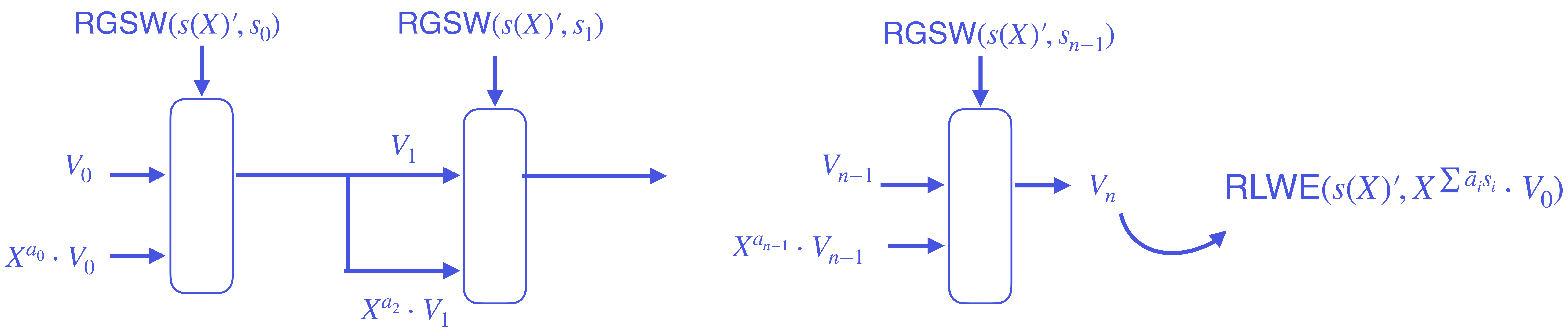
LWE$_{\mathbf{s}}(m) = (\mathbf{a}, b)$ over q $\longrightarrow$ $\bar{a}_0, \cdots, \bar{a}_{n-1}, \bar{b}$ over $2N$
and
$V_0 := X^{-b} \cdot (\mathbf{0}, v(X))$

TFHE bootstrapping



TFHE bootstrapping





TFHE bootstrapping

LWE$_{\mathbf{s}}(m) = (\mathbf{a}, b)$ over q

$\bar{a}_0, \cdots, \bar{a}_{n-1}, \bar{b}$ over $2N$
and
$V_0 := X^{-b} \cdot (\mathbf{0}, v(X))$

RLWE$(s(X)', X^{\sum \bar{a}_i s_i} \cdot V_0)$

TFHE bootstrapping

LWE$_{\mathbf{s}}(m) = (\mathbf{a}, b)$ over q $\longrightarrow$ $\bar{a}_0, \cdots, \bar{a}_{n-1}, \bar{b}$ over $2N$ and $V_0 := X^{-b} \cdot (\mathbf{0}, v(X))$ $\longrightarrow$ RLWE$(s(X)', X^{\sum \bar{a}_i s_i} \cdot V_0)$

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

TFHE bootstrapping

$$\text{LWE}_{\mathbf{s}}(m) = (\mathbf{a}, b) \text{ over } q \longrightarrow \begin{array}{c} \bar{a}_0, \cdots, \bar{a}_{n-1}, \bar{b} \text{ over } 2N \\ \text{and} \\ V_0 := X^{-b} \cdot (\mathbf{0}, v(X)) \end{array} \longrightarrow \text{RLWE}(s(X)', X^{\sum \bar{a}_i s_i} \cdot V_0)$$

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

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

LWE$_{\mathrm{s}}(m) = (\mathbf{a}, b)$ over q $\rightarrow$ $\bar{a}_0, \cdots, \bar{a}_{n-1}, \bar{b}$ over $2N$ and $V_0 := X^{-b} \cdot (\mathbf{0}, v(X))$ $\rightarrow$ RLWE$(s(X)', X^{\sum \bar{a}_i s_i} \cdot V_0)$

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

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

$\mathsf{LWE}_{\mathbf{s}}(m) = (\mathbf{a}, b)$ over q $\rightarrow$ $\bar{a}_0, \cdots, \bar{a}_{n-1}, \bar{b}$ over $2N$ and $V_0 := X^{-b} \cdot (\mathbf{0}, v(X))$ $\rightarrow$ $\mathsf{RLWE}(s(X)', X^{\sum \bar{a}_i s_i} \cdot V_0)$ $\rightarrow$ $\mathsf{LWE}(\tilde{\mathbf{s}}, m)$

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

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

LWE$_{\mathbf{s}}(m) = (\mathbf{a}, b)$ over q

$\downarrow$

$\bar{a}_0, \cdots, \bar{a}_{n-1}, \bar{b}$ over $2N$
and
$V_0 := X^{-b} \cdot (\mathbf{0}, v(X))$

$\downarrow$

RLWE$(s(X)', X^{\sum \bar{a}_i s_i} \cdot V_0)$

$\downarrow$

LWE$(\tilde{\mathbf{s}}, m)$

$\downarrow$

LWE$_{\mathbf{s}}(m) = (\mathbf{a}, b)$ over q

TFHE bootstrapping

LWE$_{\mathbf{s}}(m) = (\mathbf{a}, b)$ over q

$\bar{a}_0, \cdots, \bar{a}_{n-1}, \bar{b}$ over $2N$

and

$V_0 := X^{-b} \cdot (\mathbf{0}, v(X))$

RLWE$(s(X)', X^{\sum \bar{a}_i s_i} \cdot V_0)$

LWE$(\tilde{\mathbf{s}}, m)$

LWE$_{\mathbf{s}}(m) = (\mathbf{a}, b)$ over q

TFHE bootstrapping

Setup

BlindRotate

Extraction

KeySwitch

$\bar{a}_i, \bar{b}$ ← rescaling of a_i, b

TFHE bootstrapping

Goal : from $X^{-(\bar{b}-\bar{\mathbf{a}}\mathbf{s})}$, enable to retrieve decode$(\bar{b} - \bar{a}s)$

Setup

BlindRotate

Extraction

KeySwitch

$\bar{a}_i, \bar{b}$ ← rescaling of a_i, b

Assume we have $X^{-\bar{\mu}^\star} = X^{-(\bar{b}-\mathbf{s}\bar{a})}$, $\mathrm{coeff}_0\left(X^{-\bar{\mu}^\star} \cdot (v_{N-1}X^{N-1} + \cdots + v_1 X + v_0)\right) = v_{\bar{\mu}^\star}$

TFHE bootstrapping

Goal : from $X^{-(\bar{b}-\bar{\mathbf{a}}\mathbf{s})}$, enable to retrieve decode$(\bar{b} - \bar{a}s)$

Setup

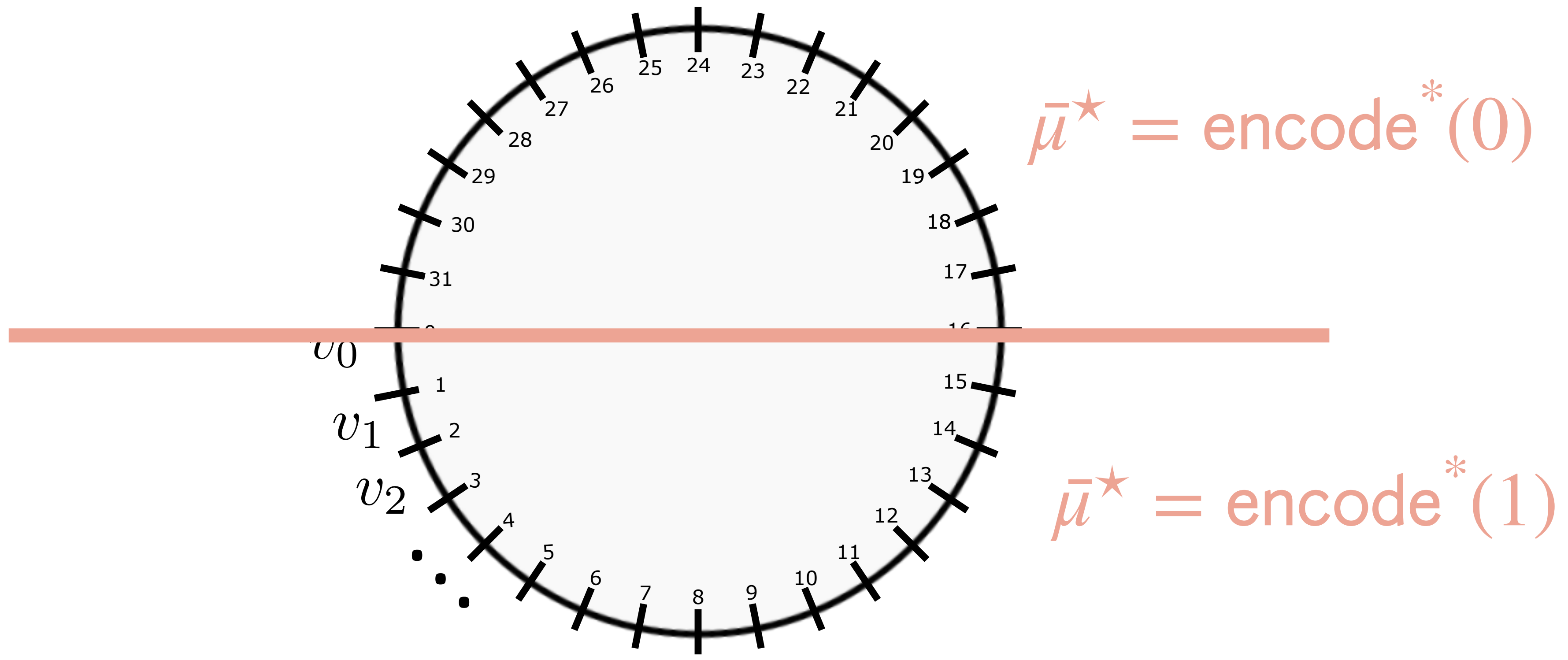
BlindRotate

Extraction

KeySwitch

$\bar{a}_i, \bar{b}$ ← rescaling of a_i, b

Assume we have $X^{-\bar{\mu}^\star} = X^{-(\bar{b}-\mathbf{s}\bar{a})}$, $\quad \mathrm{coeff}_0\left(X^{-\bar{\mu}^\star} \cdot (v_{N-1}X^{N-1} + \cdots + v_1 X + v_0)\right) = v_{\bar{\mu}^\star}$



TFHE bootstrapping

Goal : from $X^{-(\bar{b}-\bar{\mathbf{a}}\mathbf{s})}$, enable to retrieve decode$(\bar{b} - \bar{a}s)$

Setup

BlindRotate

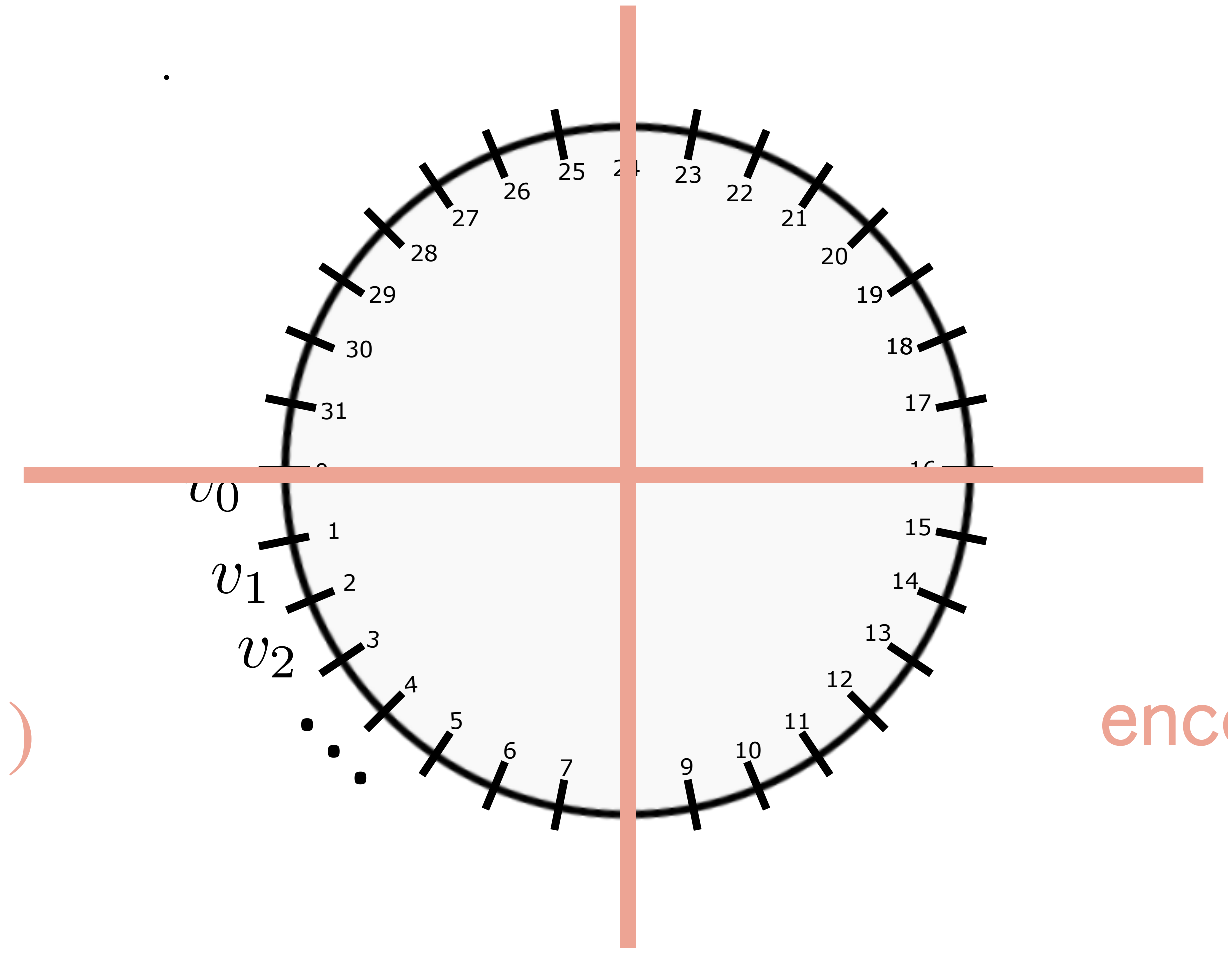
Extraction

KeySwitch

$\bar{a}_i, \bar{b}$ ← rescaling of a_i, b

Assume we have $X^{-\bar{\mu}^\star} = X^{-(\bar{b}-\mathbf{s}\bar{a})}$, $\quad \mathrm{coeff}_0\left(X^{-\bar{\mu}^\star} \cdot (v_{N-1}X^{N-1} + \cdots + v_1X + v_0)\right) = v_{\bar{\mu}^\star}$

encode$^*(1)$



encode$^*(0)$

TFHE bootstrapping

Goal : evaluate $X^{-(\bar{b}-\bar{\mathbf{a}}\mathbf{s})}$ homomorphically

Setup | **BlindRotate** | **Extraction** | **KeySwitch**

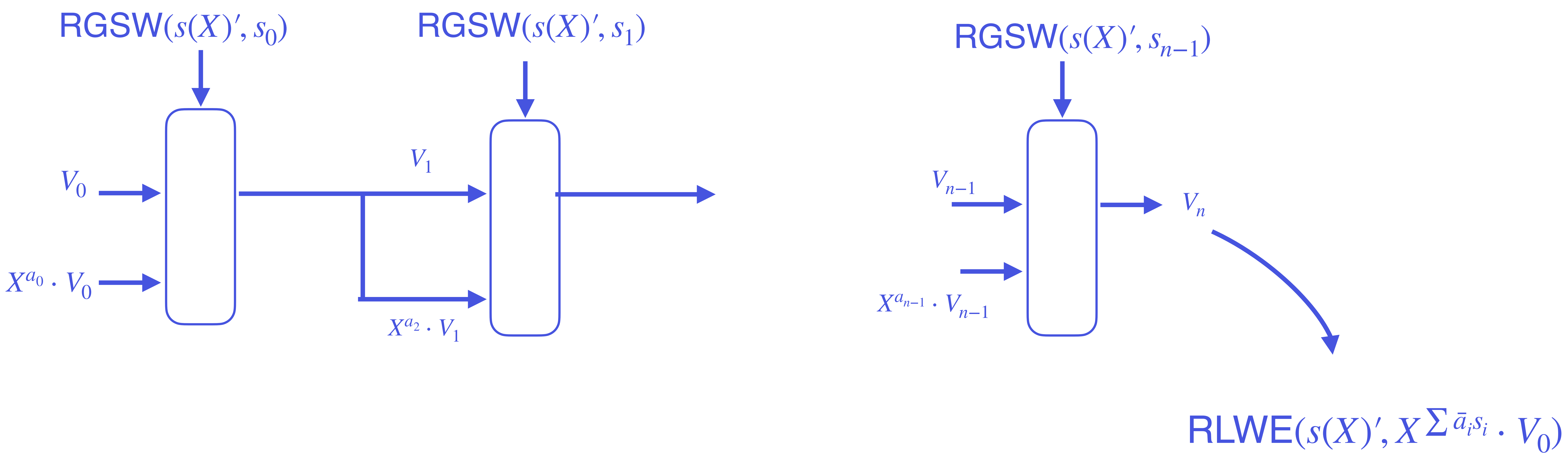
$$V_0 = X^{-b} \cdot (\mathbf{0}, v(X))$$

TFHE bootstrapping

Goal : evaluate $X^{-(\bar{b}-\bar{\mathbf{a}}\mathbf{s})}$ homomorphically

Setup **BlindRotate** **Extraction** **KeySwitch**

$$V_0 = X^{-\bar{b}} \cdot (\mathbf{0}, v(X))$$



TFHE bootstrapping

Setup

BlindRotate

Extraction

KeySwitch

$$\mathsf{RLWE}(s(X)', X^{-\sum \bar{a}_i s_i} \cdot V_0) \longrightarrow \mathsf{LWE}_{\mathbf{s}'}(m)$$

TFHE bootstrapping

Setup

BlindRotate

Extraction

KeySwitch

$\mathsf{LWE}_{\mathbf{s}'}(m) \longrightarrow \mathsf{LWE}_{\mathbf{s}}(m)$

Circuit private BlindRotate

Setup

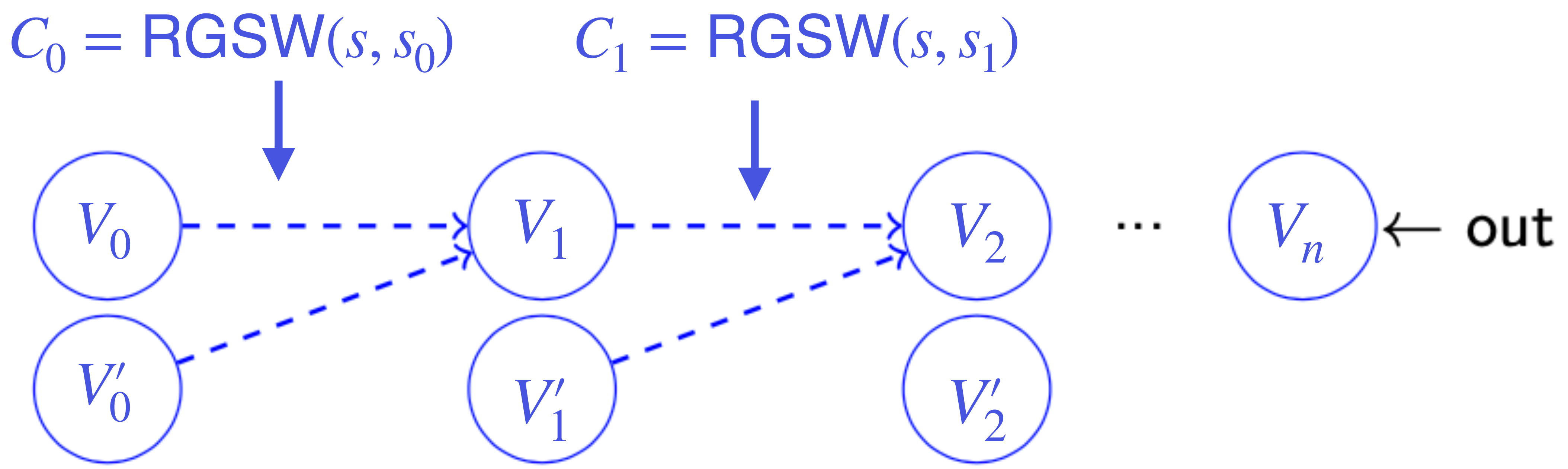
circuit private BlindRotate

Extraction

KeySwitch

Circuit private BlindRotate

Setup | circuit private BlindRotate | Extraction | KeySwitch



$$\underbrace{\mathbf{G}^{-1}(V_t - V_t')}_{=x} \cdot \tilde{C}_t + (\ \mathbf{0} \mid y\ ) \approx_S C$$

fresh GSW encryption of 0

GSW encryption of 0
independent from V_t, V_t'

Uniformity over R_q, q power of 2

$$x \cdot \left(B \mid sB + e \right) + \left(\mathbf{0} \mid y \right) \approx_S C$$

RLWE encryption of 0

Needs to show :

- uniformity of the left part $x \cdot B$
- noise is Gaussian of parameter independent of that of V_0, V_0' $x \cdot (sB + e + y)$

Uniformity over R_q, q power of 2

$$x \cdot \left(B \mid sB + e \right) + \left(\mathbf{0} \mid y \right) \approx_S C$$

RLWE encryption of 0

Needs to show :

- uniformity of the left part $x \cdot B$
- noise is Gaussian of parameter independent of that of V_0, V_0' $x \cdot (sB + e + y)$

Uniformity over R_q, q power of 2

$$x \cdot \left(B \mid sB + e \right) + \left(\mathbf{0} \mid y \right) \approx_S C$$

RLWE encryption of 0

Needs to show :

- uniformity of the left part $x \cdot B$

- noise is Gaussian of parameter independent of that of V_0, V_0' $\quad x \cdot (sB + e + y)$

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 $\mathbb{Z}_2$? $(q = 2,\ N = 2^0)$

x

x invertible ?

0 if $x = a \cdot b$

1 if x uniform

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

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

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

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

Uniformity over R_q, q power of 2

$$x \cdot \underbrace{\left(B \mid sB + e \right)}_{\in R_q^{(d+1)\ell \times d \; + \; (d+1)\ell \times d}} + \left(\mathbf{0} \mid y \right) \approx_S C$$

RLWE encryption of 0

BUT, LHL needs large ℓ, in practice $\ell = 3$ or 4

Needs to show :

- uniformity of the left part $x \cdot B$
- noise is Gaussian of parameter independent of that of V_0, V_0' $x \cdot (sB + e + y)$

Add fresh encryptions of 0

$$\mathsf{Enc}(\mathsf{pk}_s) + \underbrace{\mathbf{G}^{-1}(V_0 - V_0')}_{R^{1\times(d+1)\ell}} \cdot \underbrace{C_0}_{\in R_q^{(d+1)\ell\times(d+1)}} + \left(\ \mathbf{0} \mid y\ \right) \approx_S C$$

randomizer

- pk_s sanitization key = fresh encryptions of 0

- $\mathsf{PK_i} = \mathsf{Enc}(\mathsf{pk}_s)$ = a large subset sum of encryption of 0 with $\{0,1\}$ coefficients

(at least $N \log q$)

Add fresh encryptions of 0

$$\mathsf{Enc}(\mathsf{pk}_s) + \underbrace{\mathbf{G}^{-1}(V_0 - V_0')}_{R^{1\times(d+1)\ell}} \cdot \underbrace{C_0}_{\in R_q^{(d+1)\ell\times(d+1)}} + \left(\ \mathbf{0} \mid y\ \right) \approx_S C$$

randomizer

- pk_s sanitization key = fresh encryptions of 0
- $\mathsf{PK_i} = \mathsf{Enc}(\mathsf{pk}_s)$ = a large subset sum of encryption of 0 with $\{0,1\}$ coefficients

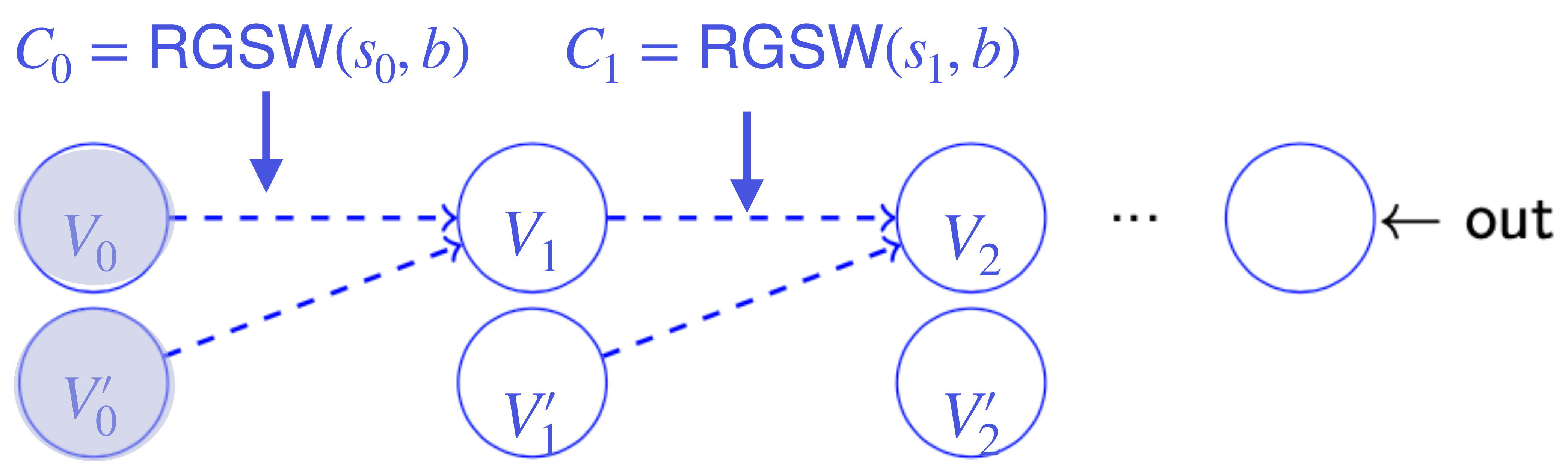
(at least $N \log q$)

- Adaptation over rings (knowing the sk)
- $\Delta(X^u \cdot \mathsf{PK}_i, \mathsf{PK}_i) = 0$

Sanitized TFHE bootstrapping

Circuit-private BlindRotate

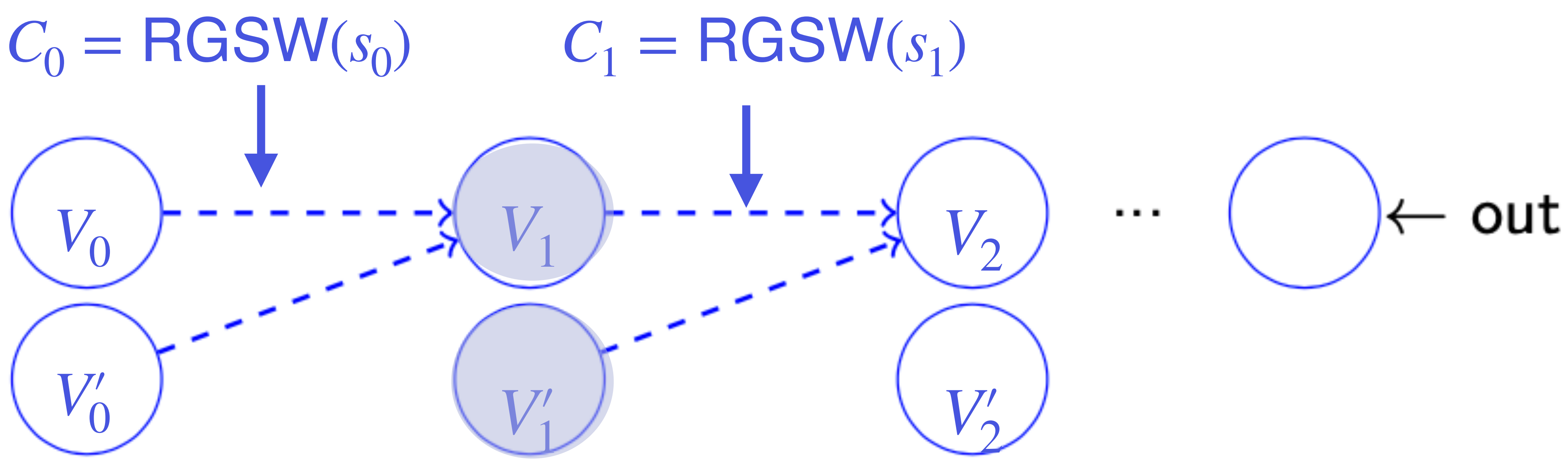
$$\mathsf{Enc}(\mathsf{pk}_s) + \mathbf{G}^{-1}(V_0 - V_0') \cdot C_0 + \left(\ \mathbf{0} \mid y\ \right) \approx_S C$$



By induction :

Circuit private BlindRotate

$$\mathsf{Enc}(\mathsf{pk}_s) + \mathbf{G}^{-1}(V_0 - V_0') \cdot C_0 + \left(\mathbf{0} \mid y \right) \approx_S C$$

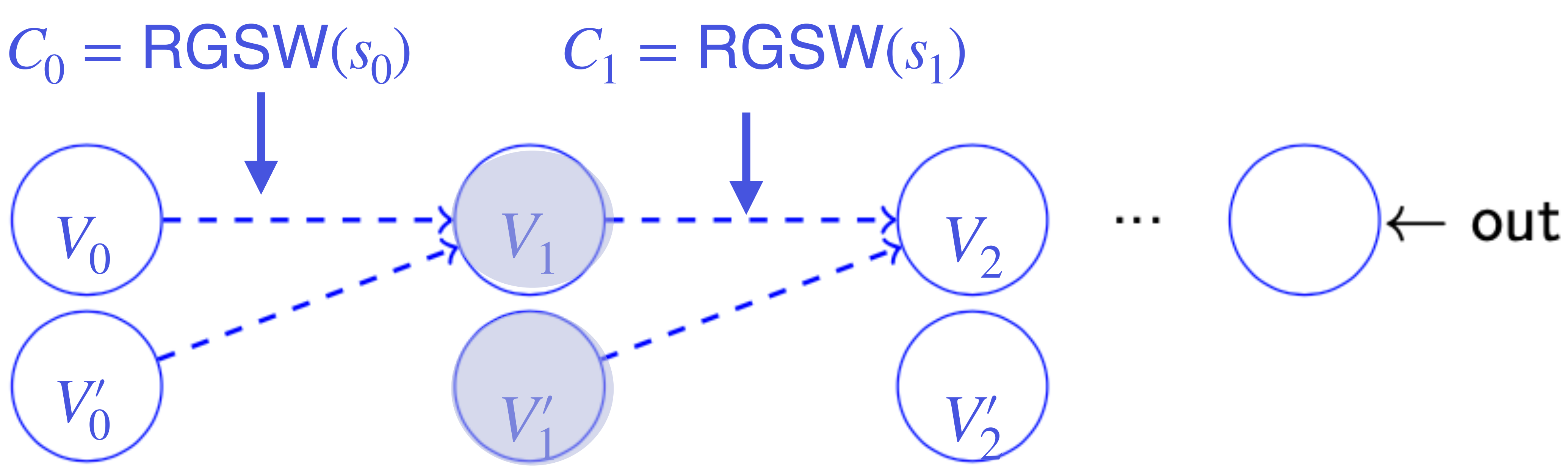


By induction :

$$\mathrm{Noise}(V_1) = \mathrm{Noise}(V_0) + \mathrm{Noise}(\mathsf{PK}_1) + \mathrm{Noise}(y) + \underbrace{\mathrm{Noise}(C_0 \odot (V_0 - V_0'))}_{=\mathsf{param}_G \times \mathrm{Noise}(C_0)}$$

Circuit private BlindRotate

$$\mathsf{Enc}(\mathsf{pk}_s) + \mathbf{G}^{-1}(V_0 - V_0') \cdot C_0 + \left(\mathbf{0} \mid y \right) \approx_S C$$



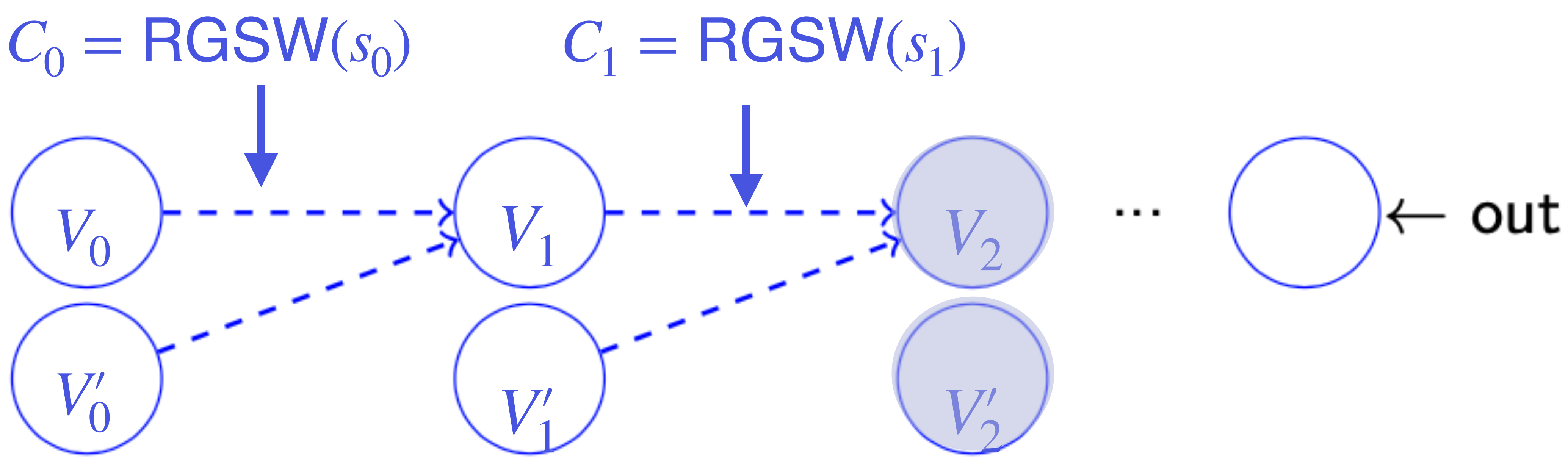
By induction :

$$\mathrm{Noise}(V_1) = \mathrm{Noise}(V_0) + \mathrm{Noise}(\mathsf{PK}_1) + \mathrm{Noise}(y) + \underbrace{\mathrm{Noise}(C_0 \odot (V_0 - V_0'))}_{=\mathsf{param}_G \times \mathrm{Noise}(C_0)}$$

$$= \mathrm{Noise}(V_0) + \underbrace{\mathrm{Noise}(\mathsf{PK}_1) + \mathrm{Noise}(y) + \mathrm{Noise}(C_0 \odot (V_0 - V_0'))}_{y_1}$$

Circuit private BlindRotate

$$\mathsf{Enc}(\mathsf{pk}_s) + \mathbf{G}^{-1}(V_0 - V_0') \cdot C_0 + \left(\mathbf{0} \mid y \right) \approx_S C$$

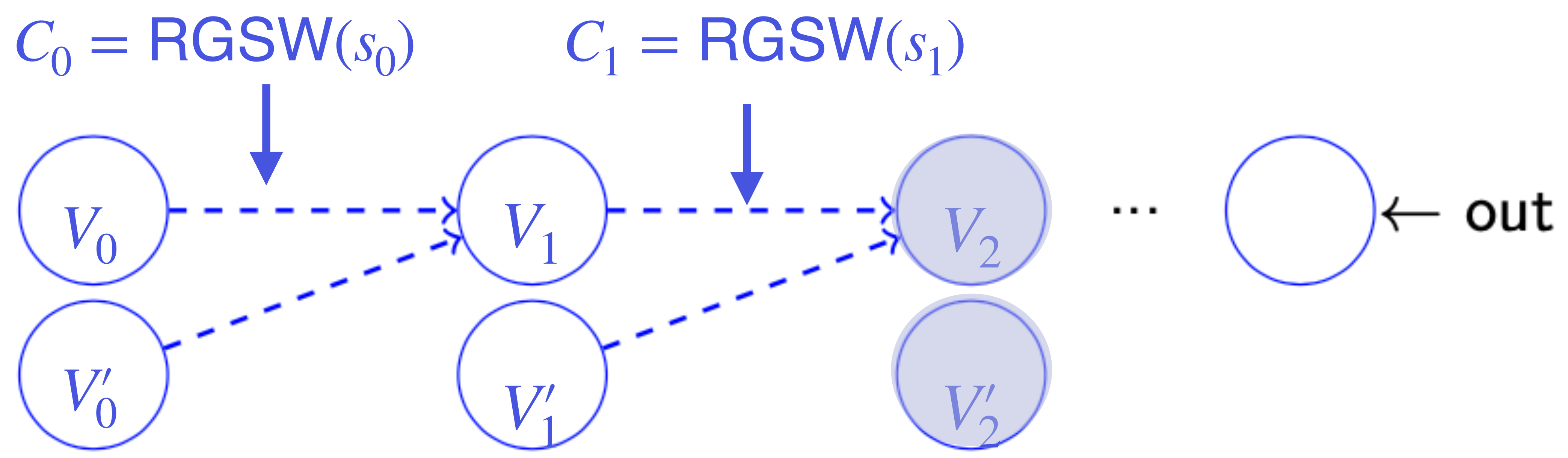


By induction :

$$\mathrm{Noise}(V_2) = \mathrm{Noise}(V_1) + \underbrace{\mathrm{Noise}(\mathsf{PK}_2) + \mathrm{Noise}(y) + \mathsf{param}_G \cdot \mathrm{Noise}(C_1)}_{y_2}$$

Circuit private BlindRotate

$$\mathsf{Enc}(\mathsf{pk}_s) + \mathbf{G}^{-1}(V_0 - V_0') \cdot C_0 + \left(\mathbf{0} \mid y \right) \approx_S C$$



By induction :

$$\mathrm{Noise}(V_2) = \mathrm{Noise}(V_1) + \underbrace{\mathrm{Noise}(\mathsf{PK}_2) + \mathrm{Noise}(y) + \mathsf{param}_G \cdot \mathrm{Noise}(C_1)}_{y_2}$$

$$= \mathrm{Noise}(V_0) + y_1 + y_2$$

Simulator for the sanitized bootstrapping

Apply TFHE bootstrapping on input (0,0)

with circuit-private BlindRotation

and add the message

Conclusion

- Capture the conditions to reach sanitization in practice
- Not in this talk, but also a proof of concept implementation
- Follow up works :
    - Reduce the size of the randomiser
    - Trade-off bootstrapping key-sizes and BlindRotate computation

Thank you

Questions : malika.izabachene@cosmian.com