HasteBoots: Proving FHE Bootstrapping in Seconds

Fengrun Liu, Haofei Liang, Tianyu Zhang,
Yuncong Hu, Xiang Xie, Haisheng Tan, Yu Yu

2025/01/10

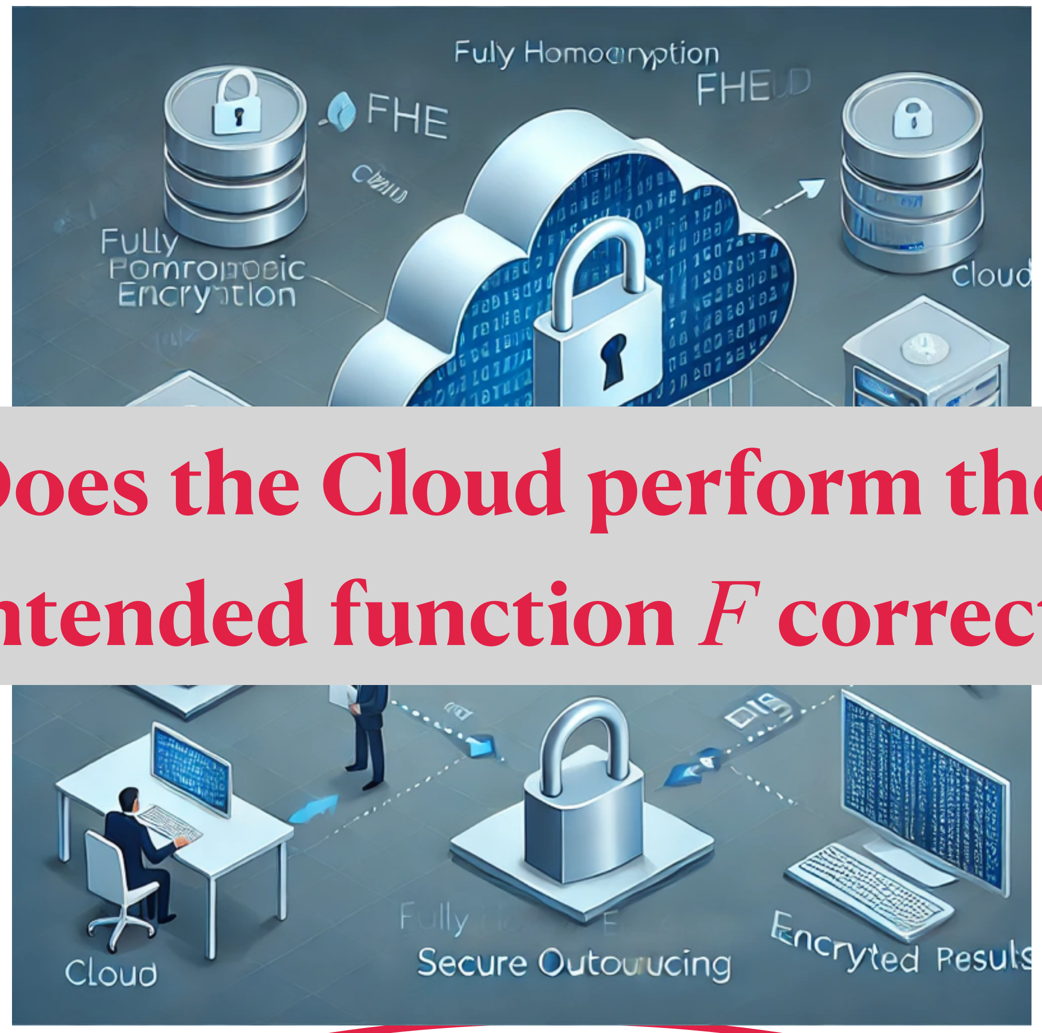
Integrity Issues in FHE

FHE enables computation to be performed directly on encrypted data.

Application: Privacy-Preserving Cloud Computing

Client

Cloud Service



Does the Cloud perform the intended function F correctly?

2. $[x]$ f →

4. $[f(x)]$ **Proofs** ←

1. $[x] = \mathrm{Enc}_k(x)$

5. $f(x) = \mathrm{Dec}_k([f(x)])$

3. $[f(x)] = F([x])$

F is the FHE circuit w.r.t f

SNARKs check

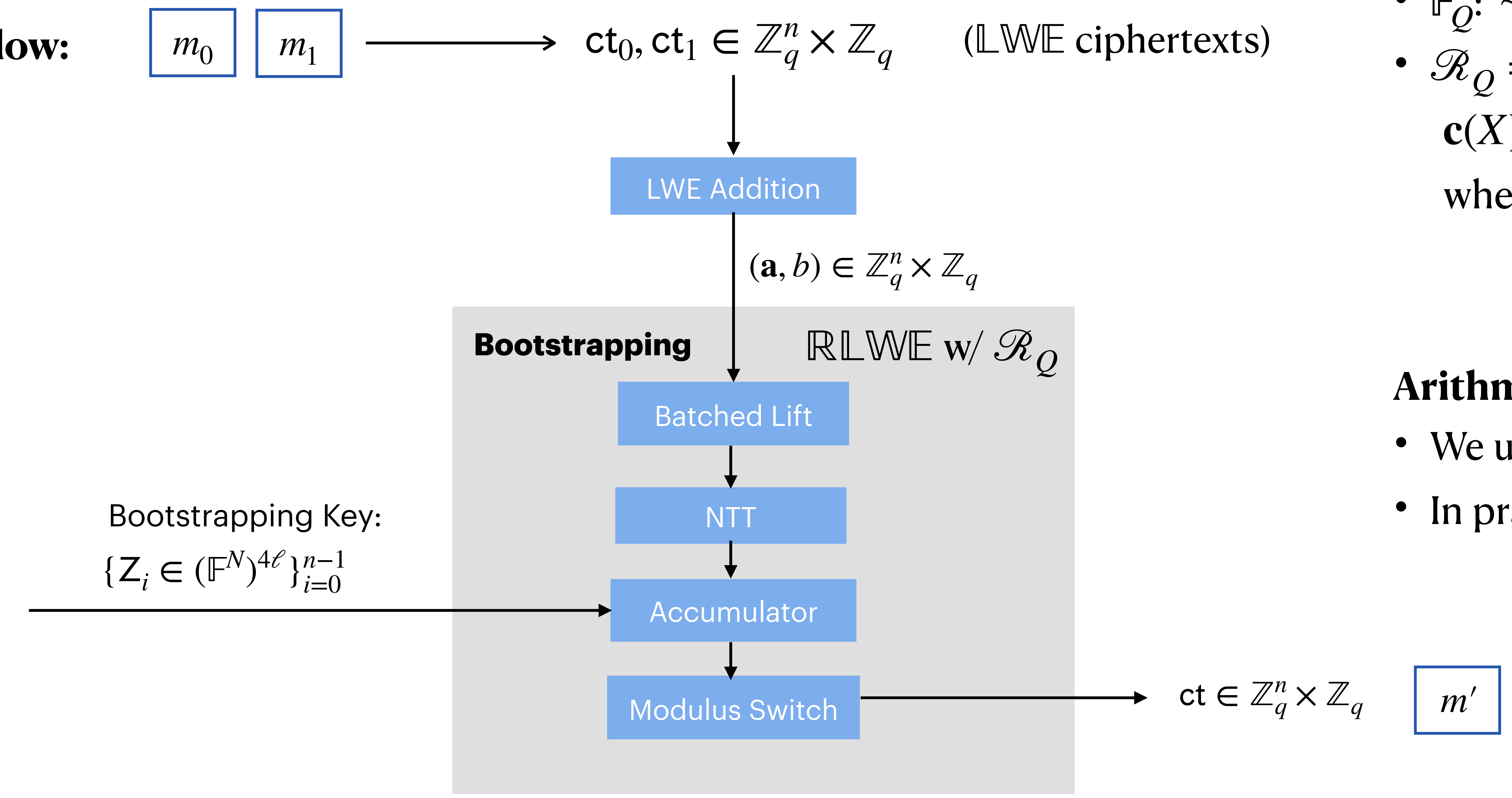
FHE NAND Operation

Full homomorphism requires the complex bootstrapping procedure

Given $m_0, m_1 \in \{0,1\}$, **NAND** computes $m' = \overline{m_0 m_1}$

In FHE: Given two FHE ciphertexts m_0 m_1, **FHE NAND** computes m'

FHE NAND Workflow:



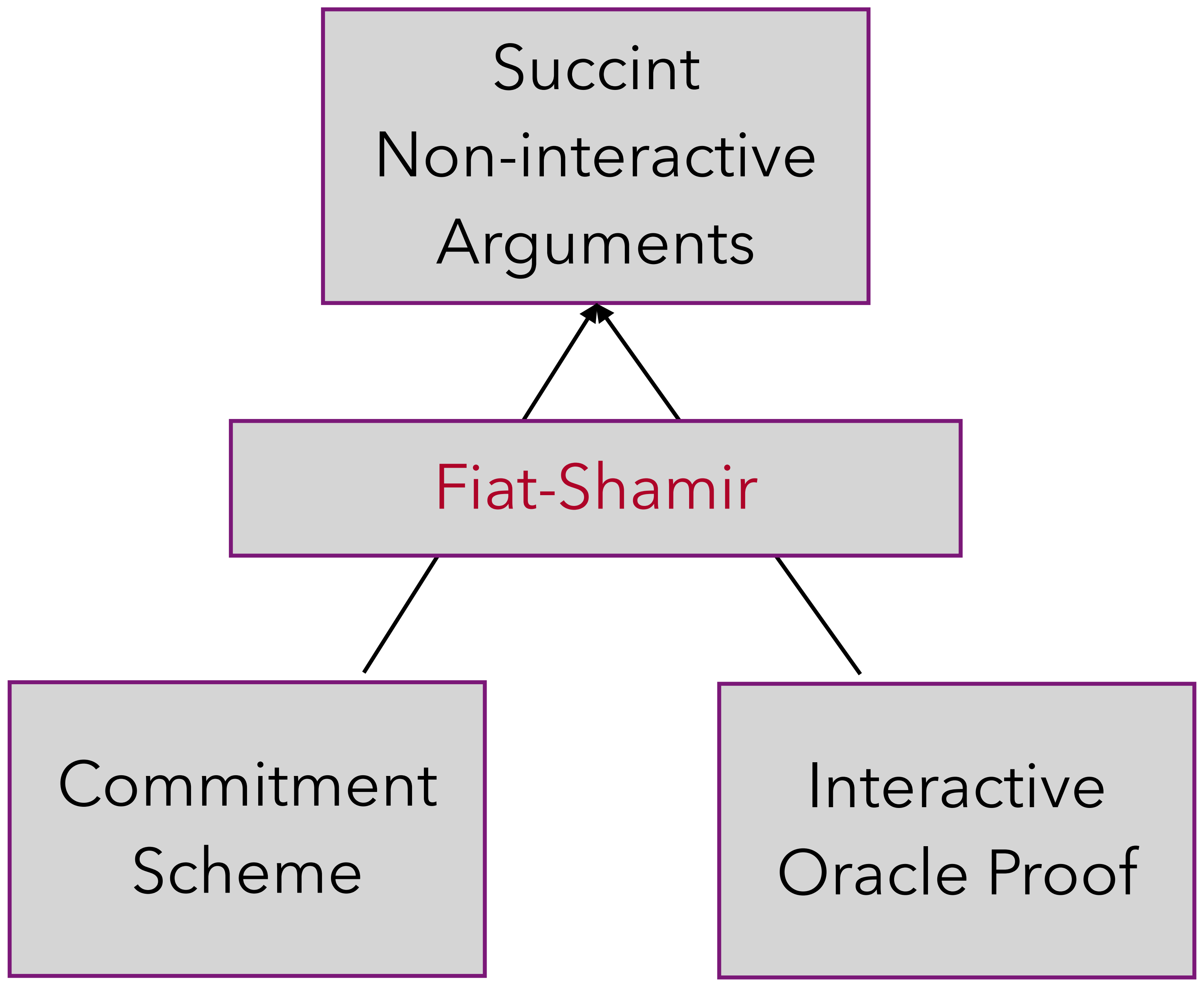
Notations:

- $\mathbb{Z}_q$: power-of-2 ring, e.g. $q = 1024$
- $\mathbb{F}_Q$: ~32-bit prime field
- $\mathcal{R}_Q = \mathbb{F}_Q[X]/(X^N + 1)$: polynomial ring
  $\mathbf{c}(X) = c_0 + \ldots + c_{N-1}X^{N-1} \in \mathcal{R}_Q$
  where $c_i \in \mathbb{F}_Q$ (rep. with $\mathbf{c} \in \mathbb{F}^N$)

Arithmetic in the proof system:

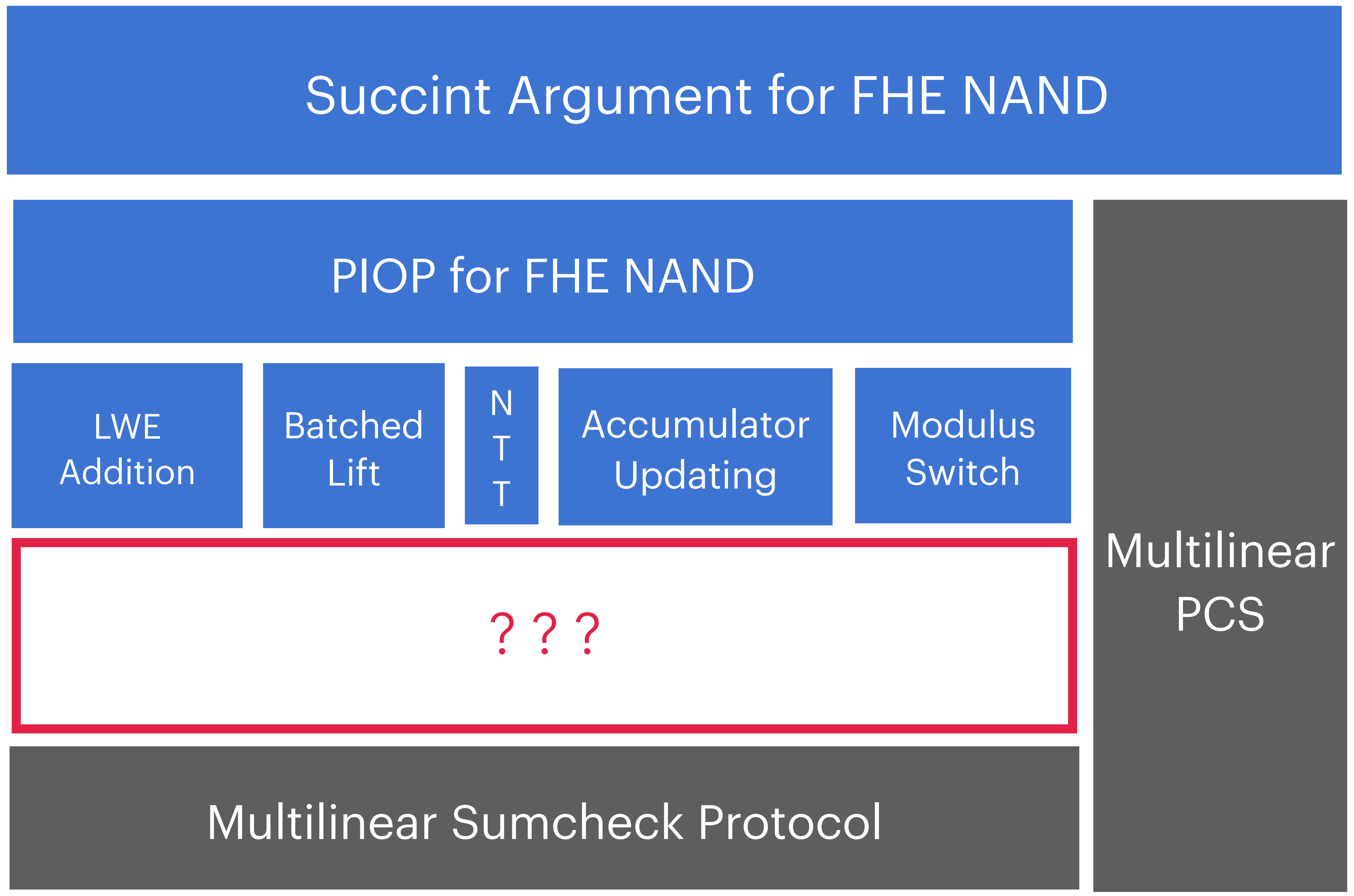
- We use $\mathbb{F}_Q$ in line with FHE
- In practice, we use $(\mathbb{F}_Q)^D$ for soundness

Paradigm for Building Succinct Arguments



(Cryptographic object) (Info. theoretic object)

Our Protocol Design



Our choice	Multilinear PCS **Brakedown**	Multilinear Polynomial IOP **Sumcheck Protocol**

Step 1: LWE Addition

FHE NAND Workflow: m_0 m_1 → $\mathsf{ct}_0, \mathsf{ct}_1 \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ (LWE ciphertexts)

↓ LWE Addition ↓ $(\mathbf{a}, b) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$

Notations:

- $\mathbb{Z}_q$: power-of-2 ring, e.g. $q = 1024$
- $\mathbb{F}_Q$: ~32-bit prime field

LWE Addition:
Given $(\mathbf{a}_1, b_1), (\mathbf{a}_2, b_2) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$,
computes $(\mathbf{a}_1 + \mathbf{a}_2, b_1 + b_2) \mod q$.

Core Relation: Given $a, b, c \in \mathbb{Z}_q$, check that $a + b = c \mod q$

Range Check ↕

$\exists w \in \{0,1\}$ such that $a + b = w \cdot q + c \mod Q$

↕

$w \cdot (1 - w) = 0$

Hadamard

Range Check:
- check $\mathbf{c} \in \mathbb{Z}_q^N$ (q is small)
- check $\mathbf{c} \in \mathbb{F}_K^N$ (K is as large as Q)

Hadamard (Sumcheck):
- check $\mathbf{a} \circ \mathbf{b} = \mathbf{c}$ $(c_i = a_i \cdot b_i)$
- check $\sum_{i=1}^{M} \mathbf{a}_i \circ \mathbf{b}_i = \mathbf{c}$

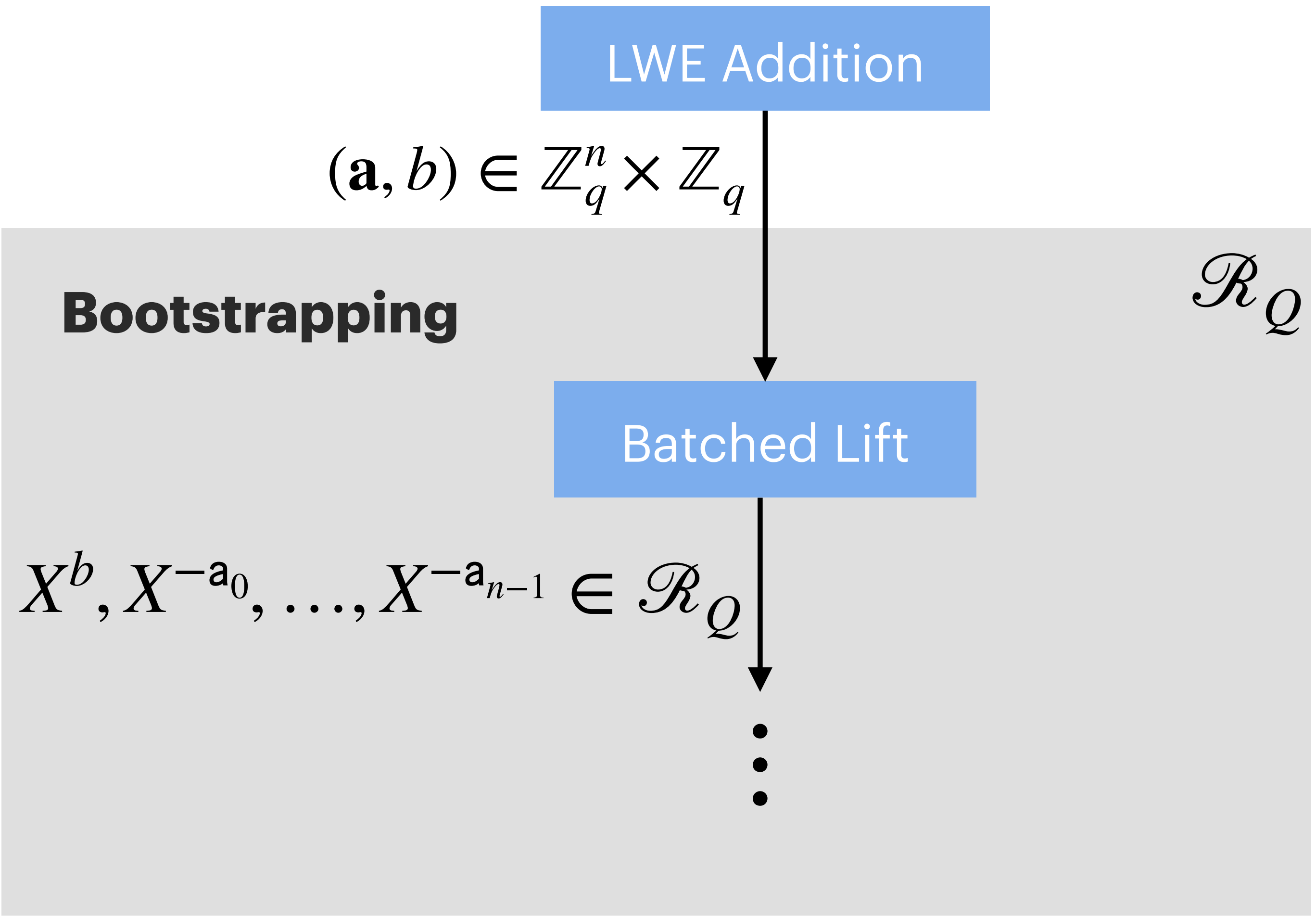
Step 2: Batched Lift

FHE NAND Workflow:

Q: Why Perform Lift?

It enables bootstrapping for **homomorphic decryption.**

For an $\mathbb{LWE}$ ciphertext $(\mathbf{a}, b)$ under the secret key $\mathbf{s}$, the decryption circuit is $\left\lfloor \frac{b - \langle \mathbf{a}, \mathbf{s} \rangle}{q/4} \right\rceil$



$\mathcal{R}_Q = \mathbb{F}_Q/(X^N + 1)$ contains $\{X, X^2, \ldots, X^{2N}\}$

assm. $q = 2N$

Batched Lift: Given $(\mathbf{a}, b) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$, computes $n + 1$ polynomials denoted by $X^b, X^{-\mathsf{a}_0}, \ldots, X^{-\mathsf{a}_{n-1}} \in \mathbb{F}^N$.

$\underbrace{\phantom{X^b, X^{-\mathsf{a}_0}, \ldots, X^{-\mathsf{a}_{n-1}}}}_{X^{b - \sum_{i=0}^{n-1} \mathbf{a}_i \cdot \mathbf{s}_i}}$

Core Relation: Given $s \in \mathbb{Z}_q$ and $\mathbf{c} \in \mathbb{F}^N$, check that $X^s = \mathbf{c}(X)$ mod $X^N + 1$

assm. $q = 2N$ ↕

$\exists k \in \{0,1\}$ s.t $\underset{[2N]}{s} = k \cdot N + \underset{[N]}{r}$ and $\mathbf{c}(X) = \begin{cases} X^r & \text{if } k = 0 \\ -X^r & \text{if } k = 1 \end{cases}$

Q: What computation is performed on $\mathbf{c}(X)$?

Sparse!

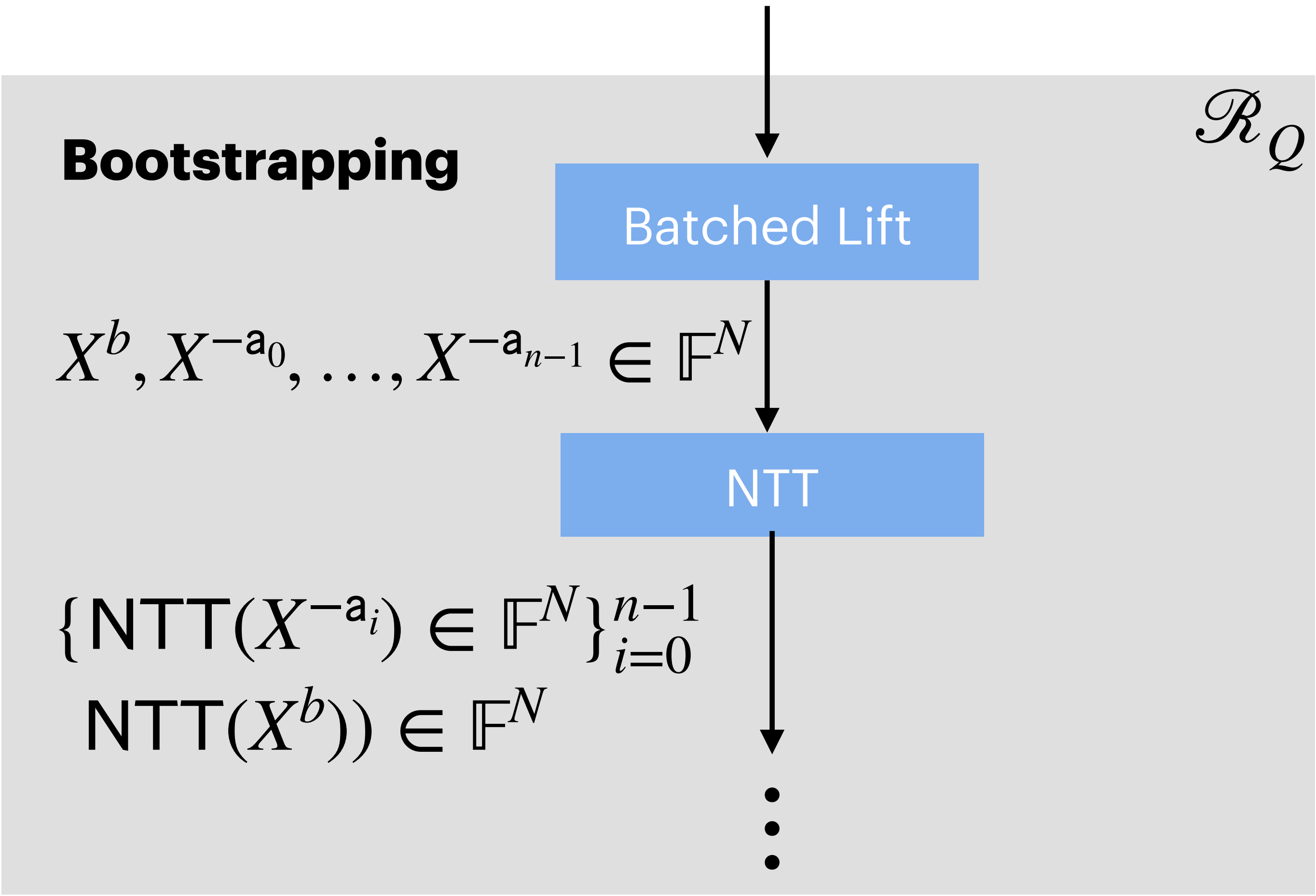
$\mathbf{c} \in \mathbb{F}^N$ contains only one non-zero entry of value $1 - 2k$, located at r

Step 2: Batched Lift +NTT

FHE NAND Workflow:

Q: Why Perform NTT?

It enables quasi-linear polynomial multiplication.



assm. $q = 2N$

Batched Lift: Given $(\mathbf{a}, b) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$, computes $n+1$ polynomials denoted by $X^b, X^{-a_0}, \ldots, X^{-a_{n-1}} \in \mathbb{F}^N$.

+

NTT: Given a polynomial $\mathbf{c}(X) \in \mathcal{R}_Q$ with coefficient vector $\mathbf{c} \in \mathbb{F}^N$, computes the evaluation vector $\mathbf{e} \in \mathbb{F}^N$, where e_i corresponds to the evaluation at point ω^{2i-1}.

$\mathbf{e} \stackrel{?}{=} \mathrm{NTT}(\mathbf{c})$ → NTT → (v, d) → V checks $d \stackrel{?}{=} \tilde{\mathbf{c}}(v)$

Batched Relation: Given $\mathbf{s} \in \mathbb{Z}_q^M$ and $\mathbf{e}_0, \ldots, \mathbf{e}_M \in \mathbb{F}^N$, check that $\mathbf{e}_i = \mathrm{NTT}(X^{s_i} \mod X^N + 1)$ for $i \in [M]$

Hadamard

Range Check

$\exists \mathbf{k} \in \{0,1\}^N$ s.t $\mathbf{s} = \mathbf{k} \cdot N + \mathbf{r}_{[N]}$ and ~~$\mathbf{c}_i(X) = \begin{cases} X^{r_i} & \text{if } k_i = 0 \\ -X^{r_i} & \text{if } k_i = 1 \end{cases}$~~ and $\mathbf{e}_i = \mathrm{NTT}(\mathbf{c}_i)$ → NTT →

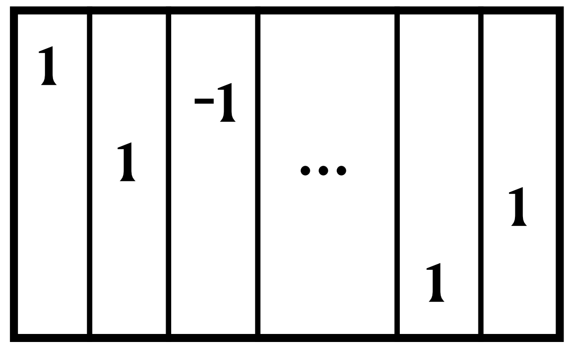
$d_0 \stackrel{?}{=} \tilde{\mathbf{c}}_0(v)$

$\vdots$

$d_{M-1} \stackrel{?}{=} \tilde{\mathbf{c}}_{M-1}(v)$

Sparse Matrix $C^{N \times M}$

sparse rep. $\begin{cases} \text{row} = \mathbf{r} \\ \text{~~col} = (0, \ldots, M-1)~~ \\ \text{val} = 1 - 2\mathbf{k} \end{cases}$



$C(u, v)$ ← random linear combination

Sparse Mat Eval

Step 3: Accumulator Updating

$$\mathbf{x} = \sum_{i=0}^{\ell-1} B^i \cdot \mathbf{a}_i$$

FHE NAND Workflow:

Bootstrapping $\mathscr{R}_Q$

$\{\mathbf{d}_i = \mathrm{NTT}(X^{-\mathrm{a}_i})\}_{i=0}^{n-1}$

$\mathrm{acc}[0] = (0, \mathrm{NTT}(X^b))$

Bootstrapping Key:

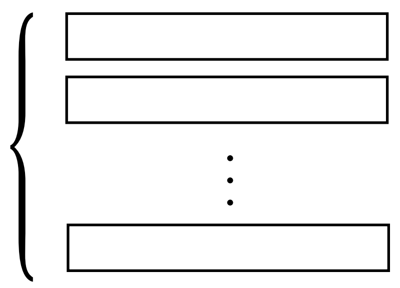
$\{\mathrm{Z}_i \in (\mathbb{F}^N)^{4\ell}\}_{i=0}^{n-1}$

Accumulator: $X^{b-\sum_{i=0}^{n-1} \mathrm{a}_i \cdot \mathrm{s}_i}$

$\mathrm{acc}[i+1] = \mathrm{acc}[i] + (d_i - 1) \otimes \mathrm{acc}[i] \diamond \mathrm{Z}_i$

$\mathrm{acc}[n] = (\mathbf{a}', \mathbf{b}')$

Each Z_i ($\mathbb{RGSW}$)



4ℓ vectors

$\mathrm{Z}_i = (\mathrm{Z}_i.\mathbf{a}_0, \ldots, \mathrm{Z}_i.\mathbf{a}_{2\ell-1}, \mathrm{Z}_i.\mathbf{b}_0, \ldots, \mathrm{Z}_i.\mathbf{b}_{2\ell-1})$

INTT: inverse of NTT

Op $\otimes$: $\mathbb{F}^N \otimes (\mathbb{F}^N, \mathbb{F}^N) \to (\mathbb{F}^N, \mathbb{F}^N)$

$$\mathbf{d} \otimes (\mathbf{a}, \mathbf{b}) = (\mathrm{INTT}(\mathbf{d} \circ \mathbf{a}), \mathrm{INTT}(\mathbf{d} \circ \mathbf{b}))$$

Hadamard NTT

Op $\diamond$: $(\mathbb{F}^N, \mathbb{F}^N) \diamond (\mathbb{F}^N)^{4\ell} \to (\mathbb{F}^N, \mathbb{F}^N)$

$$(\mathbf{x}, \mathbf{y}) \diamond \mathrm{Z}_i = (\mathbf{a}', \mathbf{b}')$$

1. Decompose **x** and **y** into 2ℓ "bits" — Gadget Dec

   $\mathrm{bits}[2\ell] = (\mathbf{a}_0, \ldots, \mathbf{a}_{\ell-1}, \mathbf{b}_0, \ldots, \mathbf{b}_{\ell-1})$ — NTT

2. Perform **NTT** on each "bit" to obtain "Nbits"

   $\mathrm{Nbits}[i] = \mathrm{NTT}(\mathrm{bits}[i])$ for $i = 0..2\ell - 1$

3. Compute — Hadamard

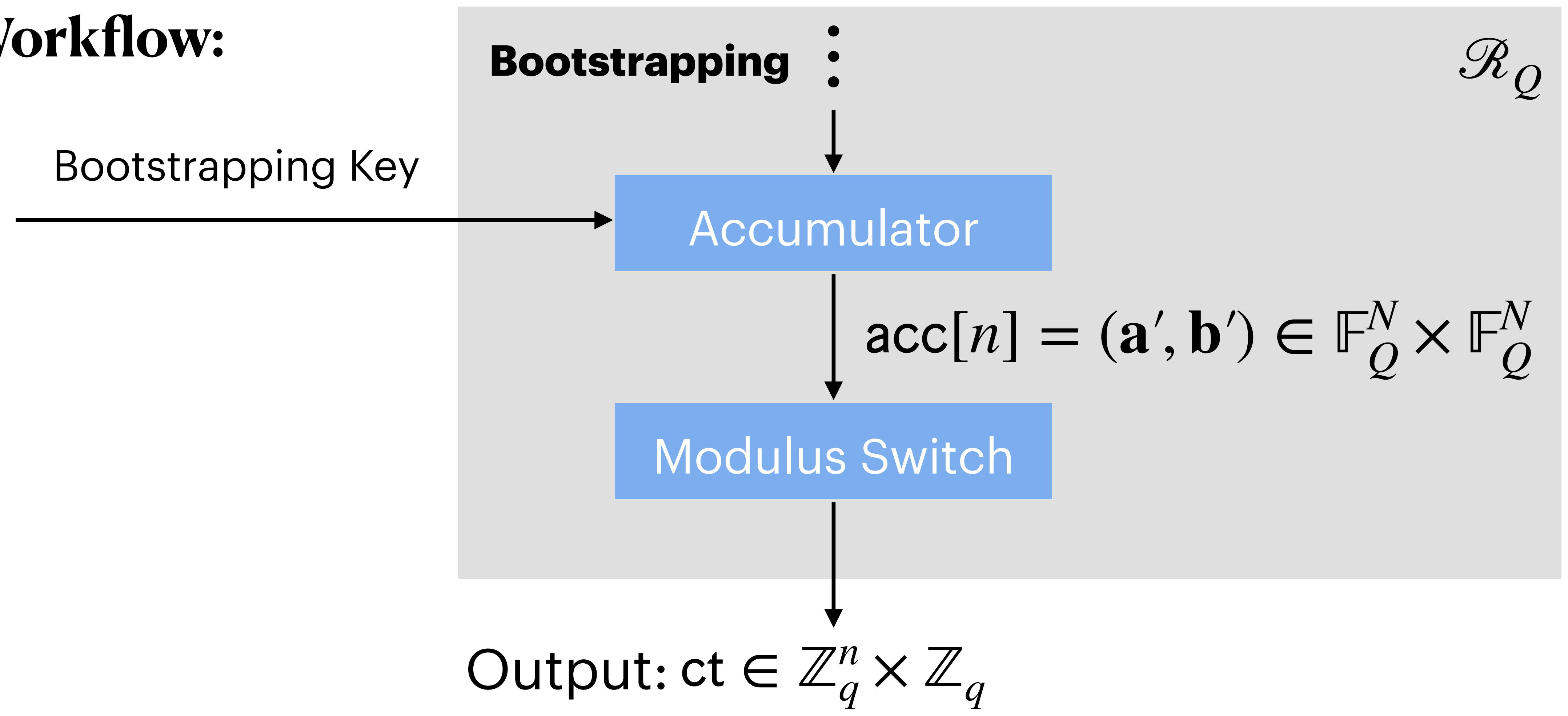
$$\left(\sum_{i=0}^{2\ell-1} \mathrm{Nbits}[i] \circ \mathrm{Z}_i.\mathbf{a}_i, \sum_{i=0}^{2\ell-1} \mathrm{Nbits}[i] \circ \mathrm{Z}_i.\mathbf{b}_i\right)$$

Core Relation:

- 4 (sums) of Hadamard products
- 2 Gadget Decomposition
- $2\ell + 2$ NTT/INTT

Step 4: Modulus Switch

FHE NAND Workflow:



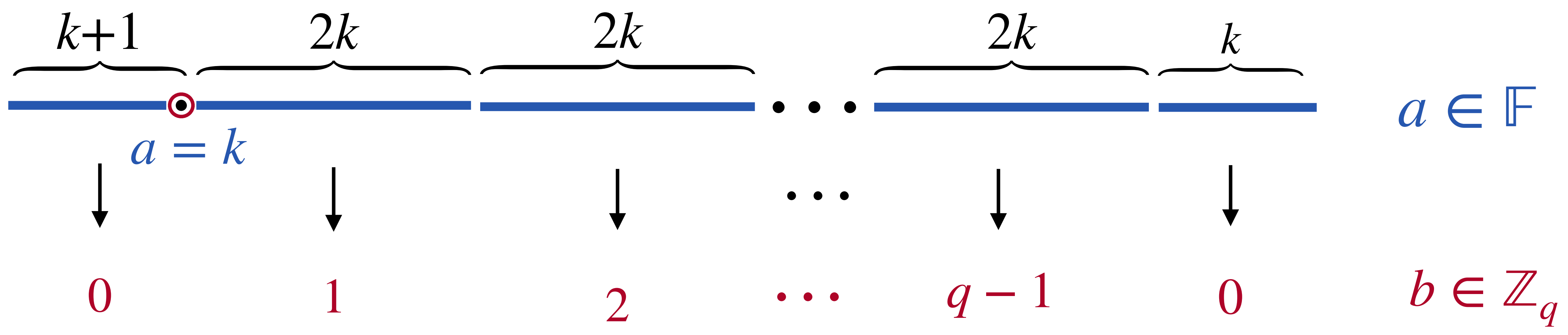
Modulus Switch from $\mathbb{F}_Q$ to $\mathbb{Z}_q$:

Given $a \in \mathbb{F}_Q$, compute

$$b = \left\lfloor \frac{a \cdot q}{Q} \right\rceil \mod q \in \mathbb{Z}_q$$

Note: $\left\lfloor \frac{a \cdot q}{Q} \right\rceil$ could be q.

Assm. $2q \mid Q - 1$, **define** $k = \dfrac{Q-1}{2q}$**:**



$$a \in [0, k] \cup [Q - k, Q) \mapsto b = 0$$

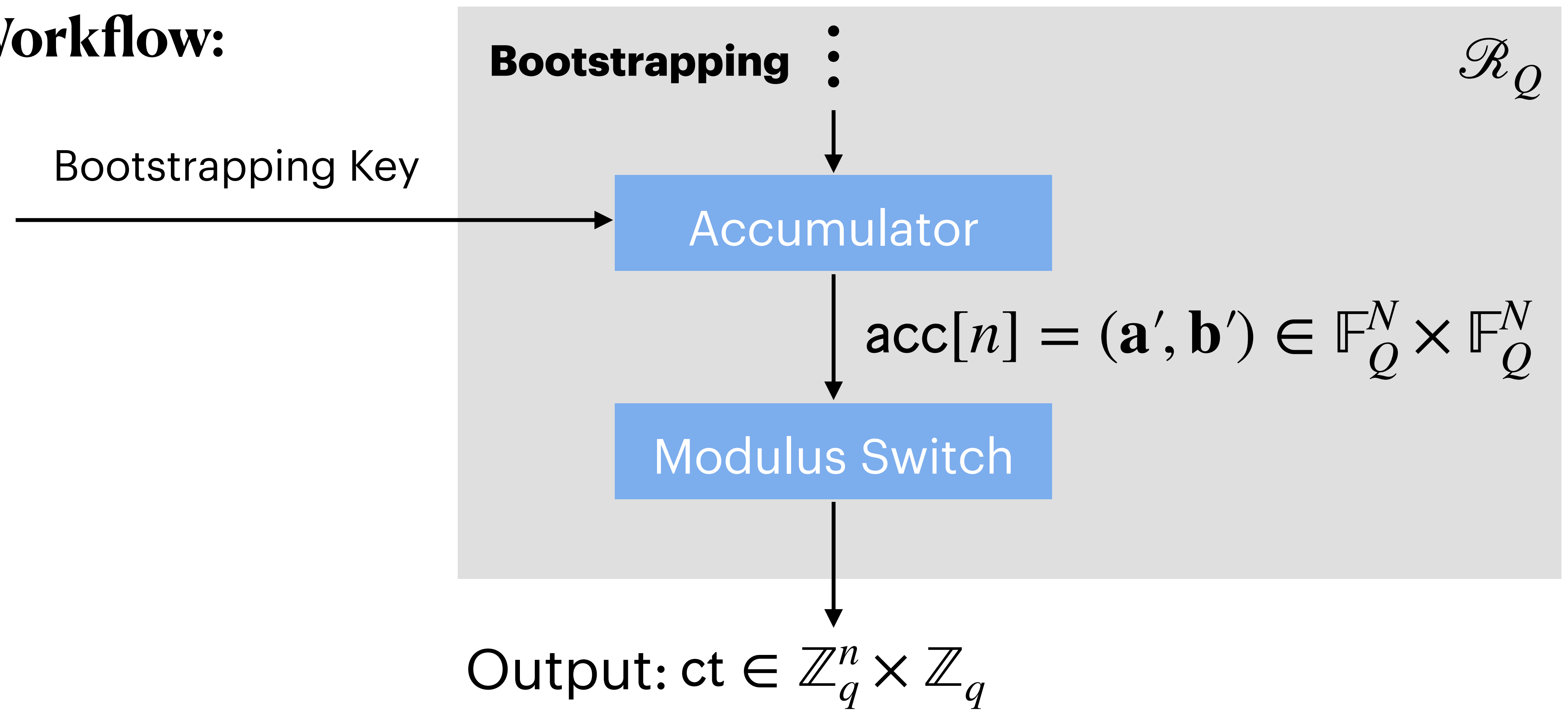
$$= k \cup [(2q - 1) \cdot k, (2q + 1) \cdot k]$$

$$a \in [(2b - 1) \cdot k + 1, (2b + 1) \cdot k] \mapsto b$$

for $b \in \{1, \ldots, q - 1\}$

Step 4: Modulus Switch

FHE NAND Workflow:



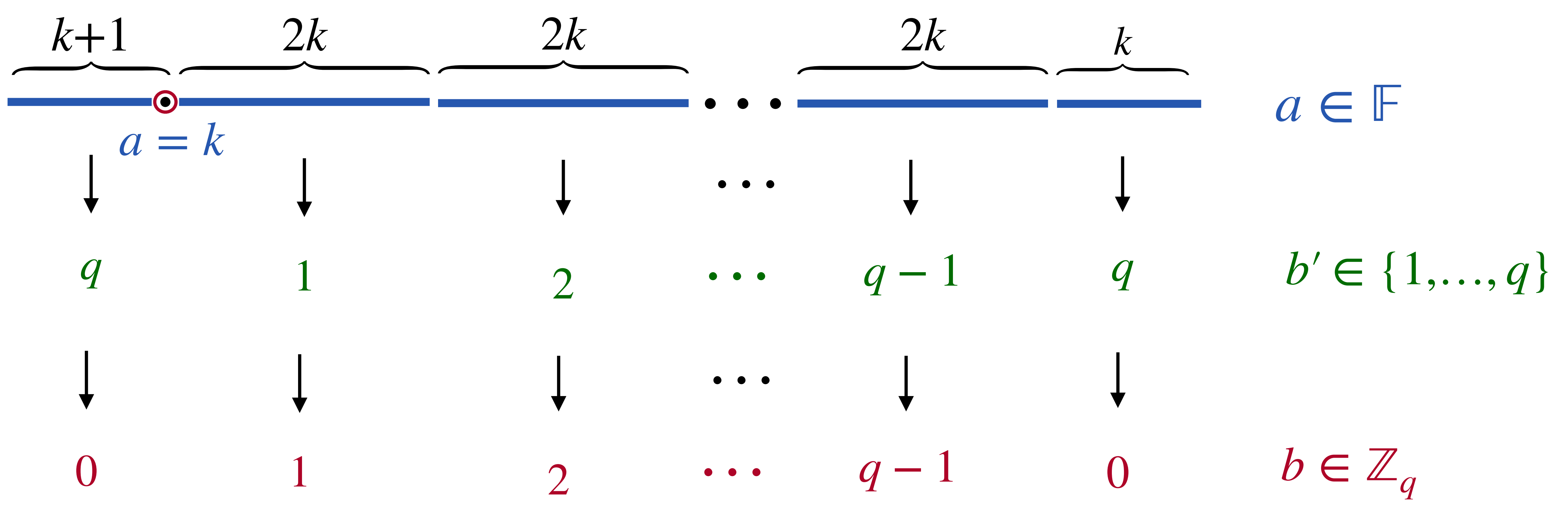
Modulus Switch from $\mathbb{F}_Q$ to $\mathbb{Z}_q$:

Given $a \in \mathbb{F}_Q$, compute

$$b = \left\lfloor \frac{a \cdot q}{Q} \right\rceil \mod q \in \mathbb{Z}_q$$

Note: $\left\lfloor \frac{a \cdot q}{Q} \right\rceil$ could be q.

Assm. $2q \mid Q - 1$**, define** $k = \dfrac{Q-1}{2q}$**:**



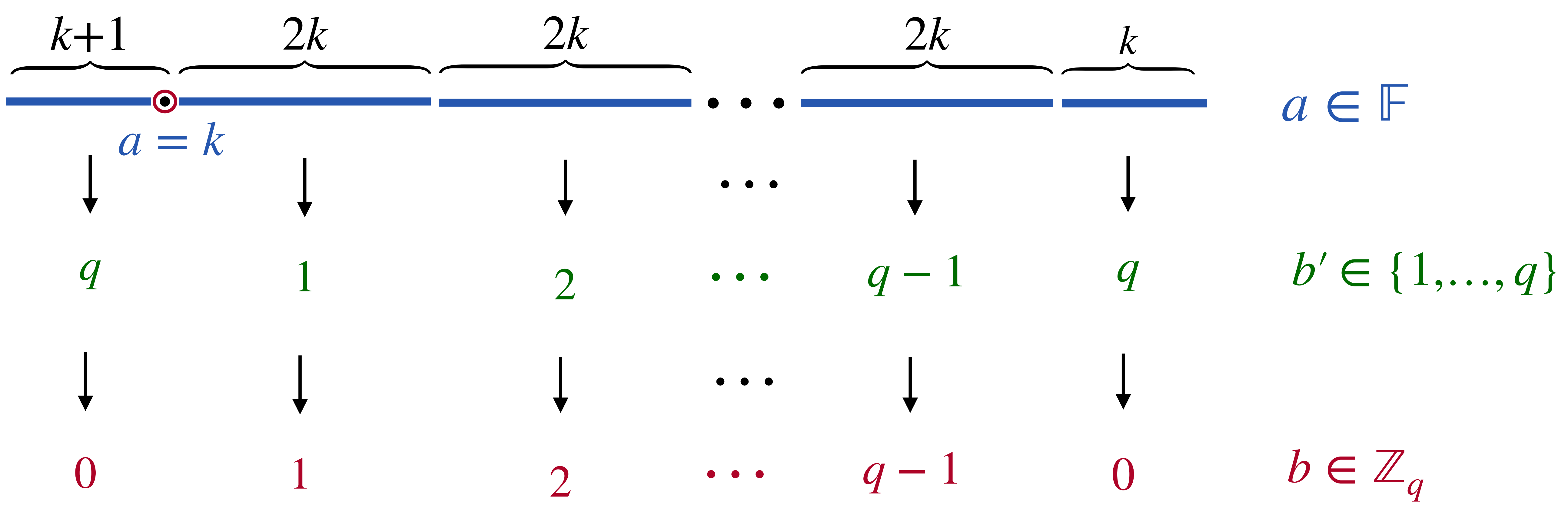
$a = k \mapsto b = 0$

$a \in [(2b'-1)\cdot k + 1, (2b'+1)\cdot k] \mapsto b'$

for $b' \in \{1, \ldots, q\}$

$b \equiv b' \mod q$

Step 4: Modulus Switch



Assm. $2q \mid Q-1$, **define** $k = \dfrac{Q-1}{2q}$. **(k is large ~ Q)**

Modulus Switch from $\mathbb{F}_Q$ to $\mathbb{Z}_q$:
Given $a \in \mathbb{F}_Q$, compute

$$b = \left\lfloor \frac{a \cdot q}{Q} \right\rceil \mod q \in \mathbb{Z}_q$$

Core Relation: Given $a \in \mathbb{F}_Q$ and $b \in \mathbb{Z}_q$, check that $b = \left\lfloor \frac{a \cdot q}{Q} \right\rceil \mod q \in \mathbb{Z}_q$

$\Updownarrow$

Dichotomy: $\exists w \in \{0,1\}$ Hadamard

$$\begin{cases} w=1: & a = k \;\; \&\& \;\; b = 0 \\ w=0: & a \in [(2b'-1)\cdot k+1, (2b'+1)\cdot k] \end{cases}$$

and $b' \in \{1,\ldots,q\}$ (Range Check) and $b \equiv b' \mod q$

$\Updownarrow$

$\exists k \in \{0,1\}$ $b' = b + k \cdot q$ (Hadamard)

$\Updownarrow$

$$\begin{cases} w=1: & p = \lambda_1 \cdot (a-k) + \lambda_2 \cdot b = 0 \\ w=0: & c = a - (2b'-1)\cdot k - 1 \;\text{ and }\; c \in [0,2k) \end{cases}$$

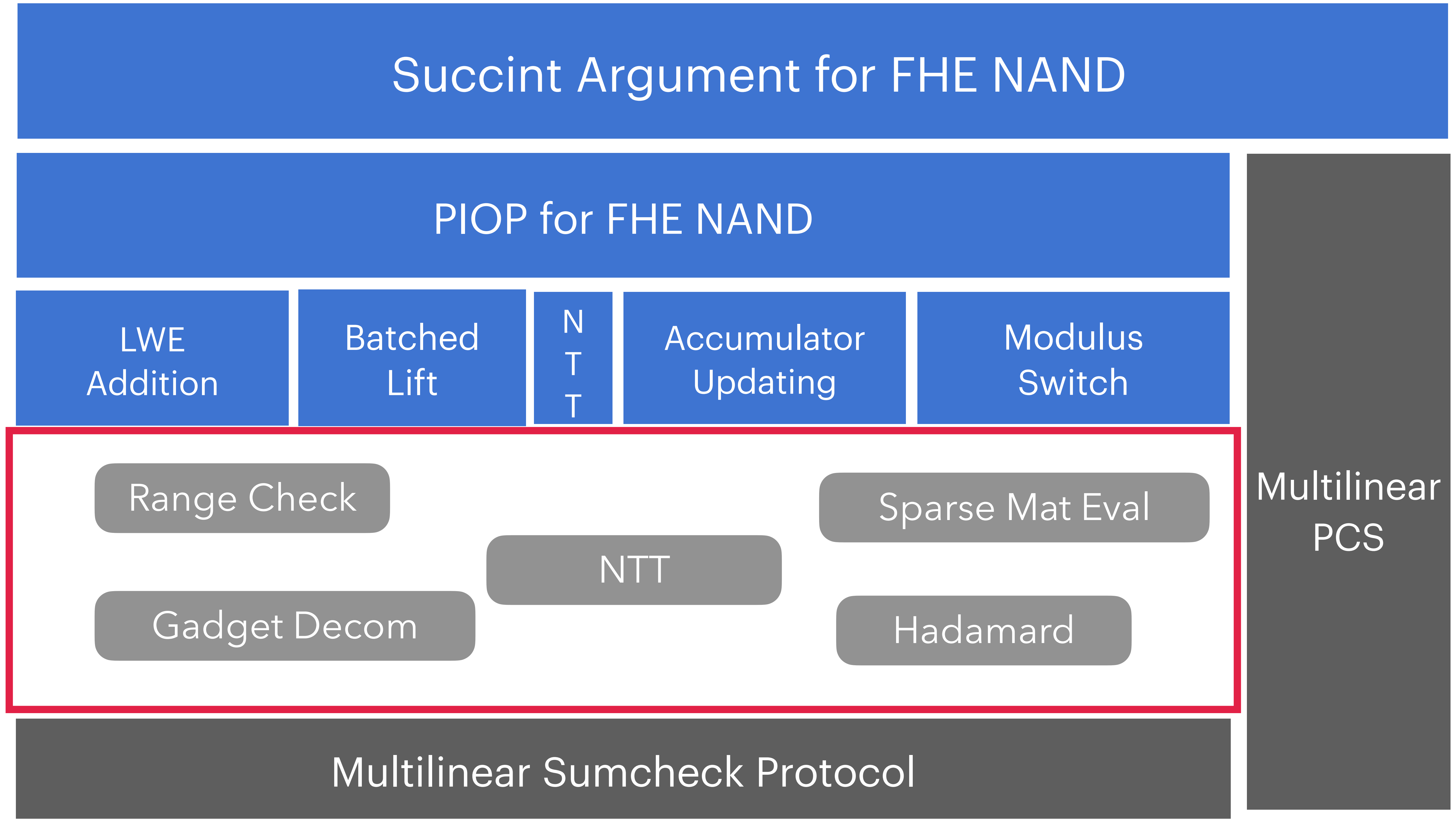
Range Check

$\Updownarrow$

Hadamard $\quad w \cdot p + (1-w)\cdot(a - (2b'-1)\cdot k - 1 - c) = 0$

Back to Our Protocol Design

5 Building Blocks



Instantiate Building Blocks

NTT

$\mathbf{c}(X) = c_0 + c_1 X + \ldots + c_{N-1} X^{N-1} \in \mathbb{F}_Q[X]/(X^N + 1)$

ω: $2N$-th roots of unity s.t. $\omega^{2N} = 1$

$O(N \log N)$

NTT

$a_Y = \mathbf{c}(\omega^{2Y+1})$ for $Y = 0, \ldots, N-1$

$\mathbf{c} = (c_0, \ldots, c_{N-1})$ ⇄ $\mathbf{a} = (a_0, \ldots, a_{N-1})$

INTT

coefficient representation

point-value representation

fast NTT: use bit-reversed order

normal order $X = \sum_{i=0}^{\ell-1} 2^i \cdot x_i$

bit-reversed order $X^R = \sum_{i=0}^{\ell-1} 2^{\log N - 1 - i} \cdot x_i$

$\mathbf{a} = (a_{000}, \quad a_{001}, \quad a_{010}, \quad a_{011}, \quad a_{100}, \quad a_{101}, \quad , a_{110}, \quad a_{111})$

$\mathbf{a}^R = (a_{000}, \quad a_{100}, \quad a_{010}, \quad a_{110}, \quad a_{001}, \quad a_{101}, \quad , a_{011}, \quad a_{111})$

Instantiate Building Blocks

NTT

$\mathbf{c}(X) = c_0 + c_1 X + \ldots + c_{N-1} X^{N-1} \in \mathbb{F}_Q[X]/(X^N + 1)$

ω: $2N$-th roots of unity s.t. $\omega^{2N} = 1$

$a'_Y = \mathbf{c}(\omega^{2Y^R+1})$ for $Y = 0, \ldots, N-1$

$\mathbf{c} = (c_0, \ldots, c_{N-1})$ — fast NTT → $\mathbf{a}^R$

$\mathbf{a}^R$ — fast INTT → $\mathbf{c} = (c_0, \ldots, c_{N-1})$

coefficient representation

point-value representation

fast NTT: use bit-reversed order

normal order $X = \sum_{i=0}^{\ell-1} 2^i \cdot x_i$

bit-reversed order $X^R = \sum_{i=0}^{\ell-1} 2^{\log N - 1 - i} \cdot x_i$

$\mathbf{a} = (a_{000}, \; a_{001}, \; a_{010}, \; a_{011}, \; a_{100}, \; a_{101}, \; , a_{110}, \; a_{111})$

$\mathbf{a}^R = (a_{000}, \; a_{100}, \; a_{010}, \; a_{110}, \; a_{001}, \; a_{101}, \; , a_{011}, \; a_{111})$

Instantiate Building Blocks

NTT

$\mathbf{c}(X) = c_0 + c_1 X + \ldots + c_{N-1} X^{N-1} \in \mathbb{F}_Q[X]/(X^N + 1)$

ω: $2N$-th roots of unity s.t. $\omega^{2N} = 1$

$a'_Y = \mathbf{c}(\omega^{2Y^R+1})$ for $Y = 0, \ldots, N-1$

$\mathbf{c} = (c_0, \ldots, c_{N-1})$ —fast NTT→ $\mathbf{a}^R$

$\mathbf{a}^R$ —fast INTT→ $\mathbf{c}$

coefficient representation

point-value representation

fast NTT: use bit-reversed order

$\mathbf{a}^R = F^R \cdot \mathbf{c}$ where F^R is $N \times N$ matrix defined as $F^R(Y, X) = \omega^{(2Y^R+1)\cdot X}$

$$\tilde{\mathbf{a}}^R(y) = \sum_{x \in \{0,1\}^{\log N}} \tilde{F}^R(y, x) \cdot \tilde{\mathbf{c}}(x) \quad \text{for } y \in \{0,1\}^{\log N}$$

↓ Schwartz-Zipple Lemma

Sumcheck

$$\tilde{\mathbf{a}}^R(u) = \sum_{x \in \{0,1\}^{\log N}} \tilde{F}^R(u, x) \cdot \tilde{\mathbf{c}}(x) \quad \text{where } u \in \mathbb{F}$$

Instantiate Building Blocks

$$\tilde{\mathbf{a}}^R(u) = \sum_{x \in \{0,1\}^{\log N}} \tilde{F}^R(u, x) \cdot \tilde{\mathbf{c}}(x) \quad \text{where } u \in \mathbb{F}$$

Sumcheck

Idea from [LXZ 21] :

- $\omega^{2^{\log N - i}} = \omega^{\frac{2N}{2^i+1}}$ is the 2^{i+1}-the roots of unity
- $2Y^R = \sum_{i=0}^{\log N - 1} 2^{\log N - i} \cdot y_i$
- $\omega^{2Y^R} = \prod_{i=0}^{\log N - 1} \left(\omega^{2^{\log N - i}}\right)^{y_i}$
- Decompose the exponents of ω
- Divide the computation into $\log N$ rounds via a dynamic algorithm

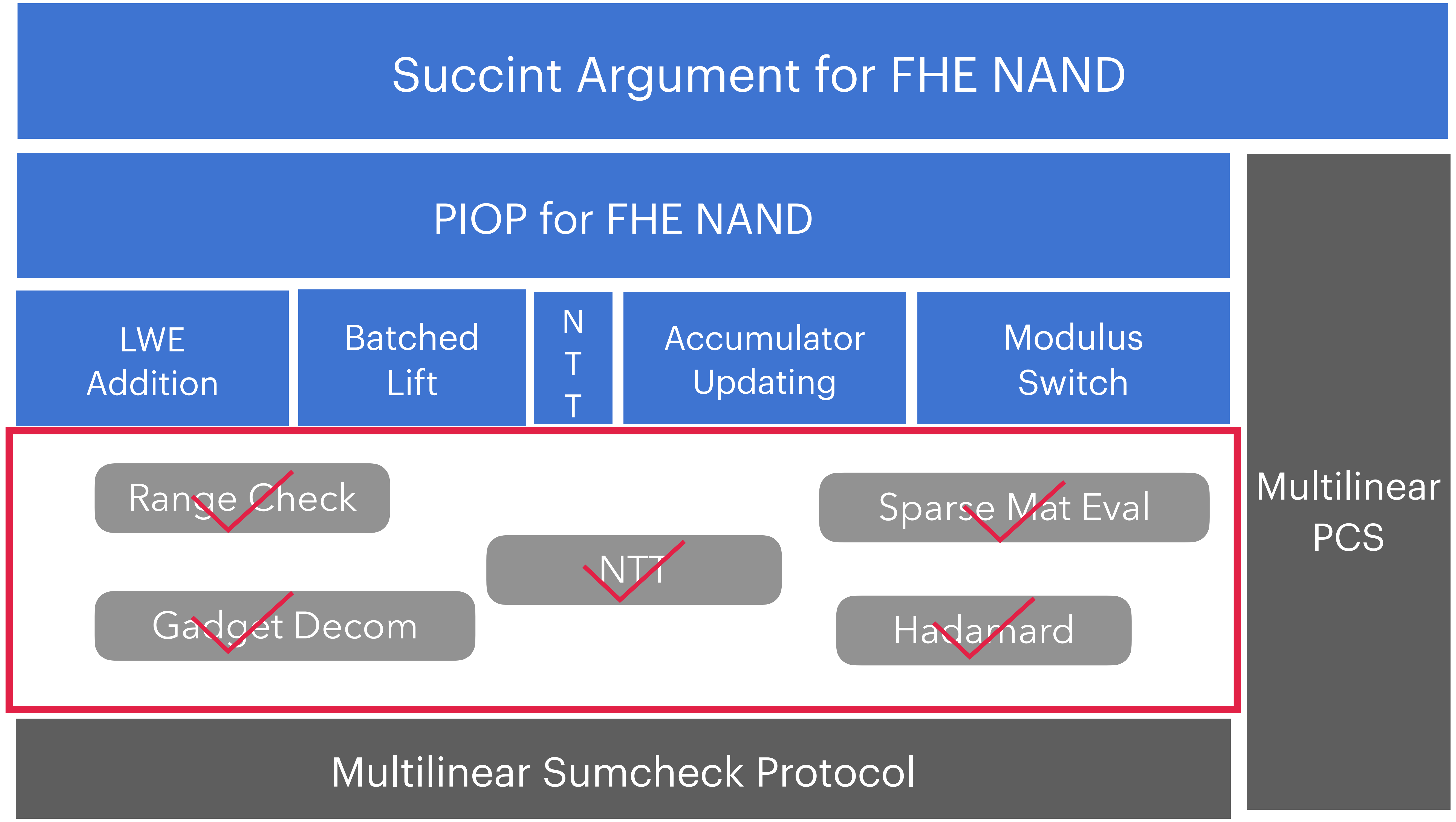
Compute $\tilde{F}^R(u, x)$ in $O(N)$!

$$\begin{aligned}
\tilde{F}^{\mathsf{R}}(u, x) &= \sum_{y \in \{0,1\}^{\log N}} \tilde{eq}(u, y)\, \tilde{F}^{\mathsf{R}}(y, x) \\
&= \sum_{y \in \{0,1\}^{\log N}} \tilde{eq}(u, y)\, \omega^{(2\mathcal{Y}^{\mathsf{R}}+1)\cdot \mathcal{X}} \\
&= \omega^{\boxed{\mathcal{X}}} \cdot \sum_{y \in \{0,1\}^{\log N}} \tilde{eq}(u, y)\, \omega^{\mathcal{X} \cdot \boxed{2\mathcal{Y}^{\mathsf{R}}}} \\
&= \prod_{i=0}^{\log N - 1} \left(1 - u_i + u_i \cdot \omega_{2^i+1}^{\mathcal{X}}\right) \cdot \omega^{2^i \cdot x_i}
\end{aligned}$$

$^*\ \omega_{2^i+1} = \omega^{2^{\log N - i}} = \omega^{\frac{2N}{2^i+1}}$

Back to Our Protocol Design

5 Building Blocks



Performance

Proving time for a single bootstrapping operation

	zkVM		Plonky2	
Proving Time	R1SC0	SP1	Zama	**Ours**
M3 Pro (8 cores)	-		40 min	**7 s**
C61.meta (128 cores)	-		21 min	**5 s**
Hpc7a.96xlarge (192 cores)	4600 min	1500 min	18 min	**4 s**
M4 Pro	-			**3 s**

Thank you for your attention!

Contact: fengrun.liu@gmail.com