Parameterized Hardware Accelerators for Lattice-Based Cryptography

and Their Application to the HW/SW Co-Design of qTESLA

Wen Wang, Shanquan Tian, Bernhard Jungk, Nina Bindel,
Patrick Longa, and Jakub Szefer

CHES 2020 – September 14, 2020

Outline

- Yet another hardware design for a lattice-based scheme?
- qTESLA
- Hardware blocks
  - Binary-search CDT sampler
  - NTT-based polynomial multiplier
- Software-hardware co-design on RISC-V
- Evaluation

Yet another hardware design for a lattice-based scheme?

Existing lattice-based hardware designs

Existing designs	Accelerated	Security parameters	Hardware architecture	Lattice-based scheme	Standard IO
Building blocks	Partly	Fixed	Fixed	Specific scheme	N/A
Full hardware design	Fully	Fixed	Fixed	Specific scheme	N/A
Software-hardware co-design	Partly	Fixed	Flexible	Specific scheme	N/A

Existing lattice-based hardware designs

Existing designs	Accelerated	Security parameters	Hardware architecture	Lattice-based scheme	Standard IO
Building blocks	Partly	Fixed	Fixed	Specific scheme	N/A
Full hardware design	Fully	Fixed	Fixed	Specific scheme	N/A
Software-hardware co-design	Partly	Fixed	Flexible	Specific scheme	N/A

Existing lattice-based hardware designs

Existing designs	Accelerated	Security parameters	Hardware architecture	Lattice-based scheme	Standard IO
Building blocks	Partly	Fixed	Fixed	Specific scheme	N/A
Full hardware design	Fully	Fixed	Fixed	Specific scheme	N/A
Software-hardware co-design	Partly	Fixed	Flexible	Specific scheme	N/A

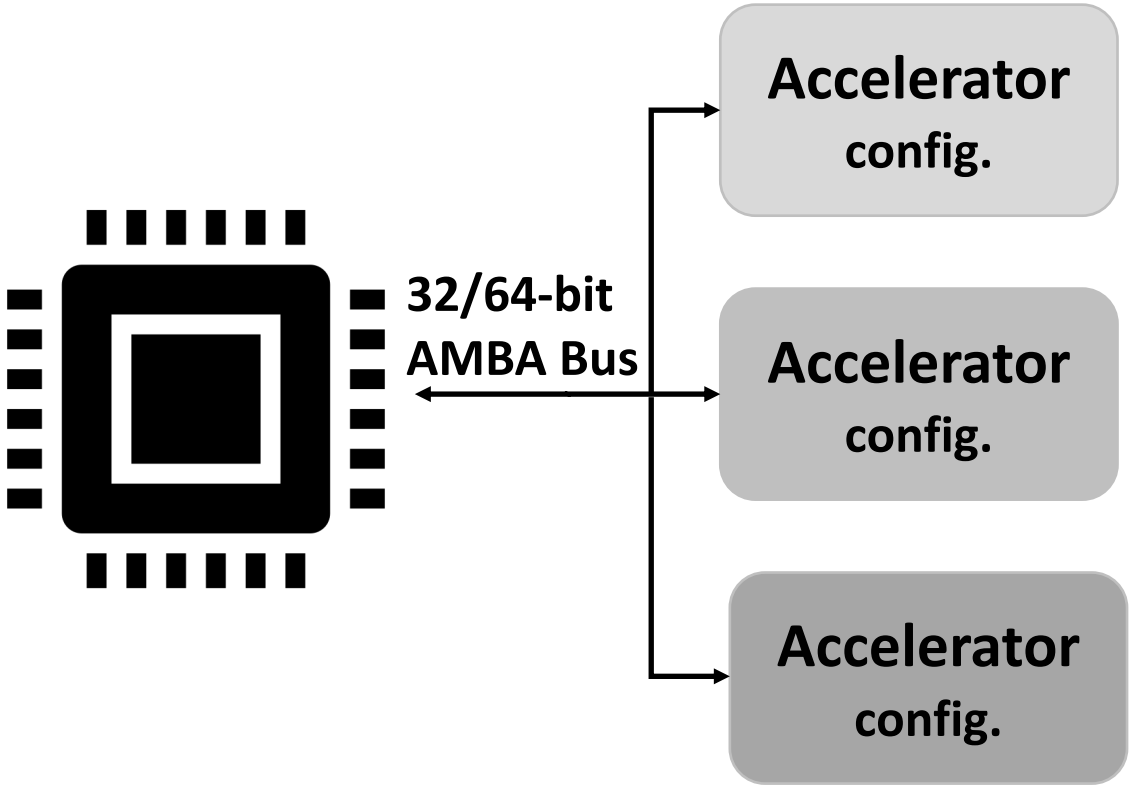
Existing lattice-based hardware designs

Existing designs	Accelerated	Security parameters	Hardware architecture	Lattice-based scheme	Standard IO
Building blocks	Partly	Fixed	Fixed	Specific scheme	N/A
Full hardware design	Fully	Fixed	Fixed	Specific scheme	N/A
Software-hardware co-design	Partly	Fixed	Flexible	Specific scheme	N/A

Our new lattice-based hardware design

Existing designs	Accelerated	Security parameters	Hardware architecture	Lattice-based scheme	Standard IO
Building blocks	Partly	Fixed	Fixed	Specific scheme	N/A
Full hardware design	Fully	Fixed	Fixed	Specific scheme	N/A
Software-hardware co-design	Partly	Fixed	Fixed	Specific scheme	N/A
Our new design	Fully	Flexible	Tunable	Universal applicability	Portable

Our new lattice-based hardware design



- ✓ Full acceleration
- ✓ Flexible security parameters
- ✓ Tunable hardware architecture
- ✓ Universal applicability to lattice-based schemes
- ✓ Portable among different platforms

qTESLA

qTESLA

Round 2 submission in NIST PQ standardization

Reference C implementation

liboqs library

BouncyCastle library

See qtesla.org

qTESLA

- ✓ Secure against classical and quantum adversaries

Round 2 submission in NIST PQ standardization

Reference C implementation

liboqs library

BouncyCastle library

See qtesla.org

qTESLA

- ✓ Secure against classical and quantum adversaries
- ✓ Implementation security

Round 2 submission in NIST PQ standardization

Reference C implementation

liboqs library

BouncyCastle library

See qtesla.org

qTESLA

- ✓ Secure against classical and quantum adversaries
- ✓ Implementation security
- ✓ Simple arithmetic operations

Round 2 submission in NIST PQ standardization	Reference C implementation
liboqs library	BouncyCastle library

See qtesla.org

qTESLA

- ✓ Secure against classical and quantum adversaries
- ✓ Implementation security
- ✓ Simple arithmetic operations
- ✓ Provable-secure parameters

Parameter set	Public key size (in B)	Signature size (in B)
qTESLA-p-I	14, 880	2, 592
qTESLA-p-III	38, 432	5, 664

Round 2 submission in NIST PQ standardization

Reference C implementation

liboqs library

BouncyCastle library

See qtesla.org

qTESLA‘s sign and verify

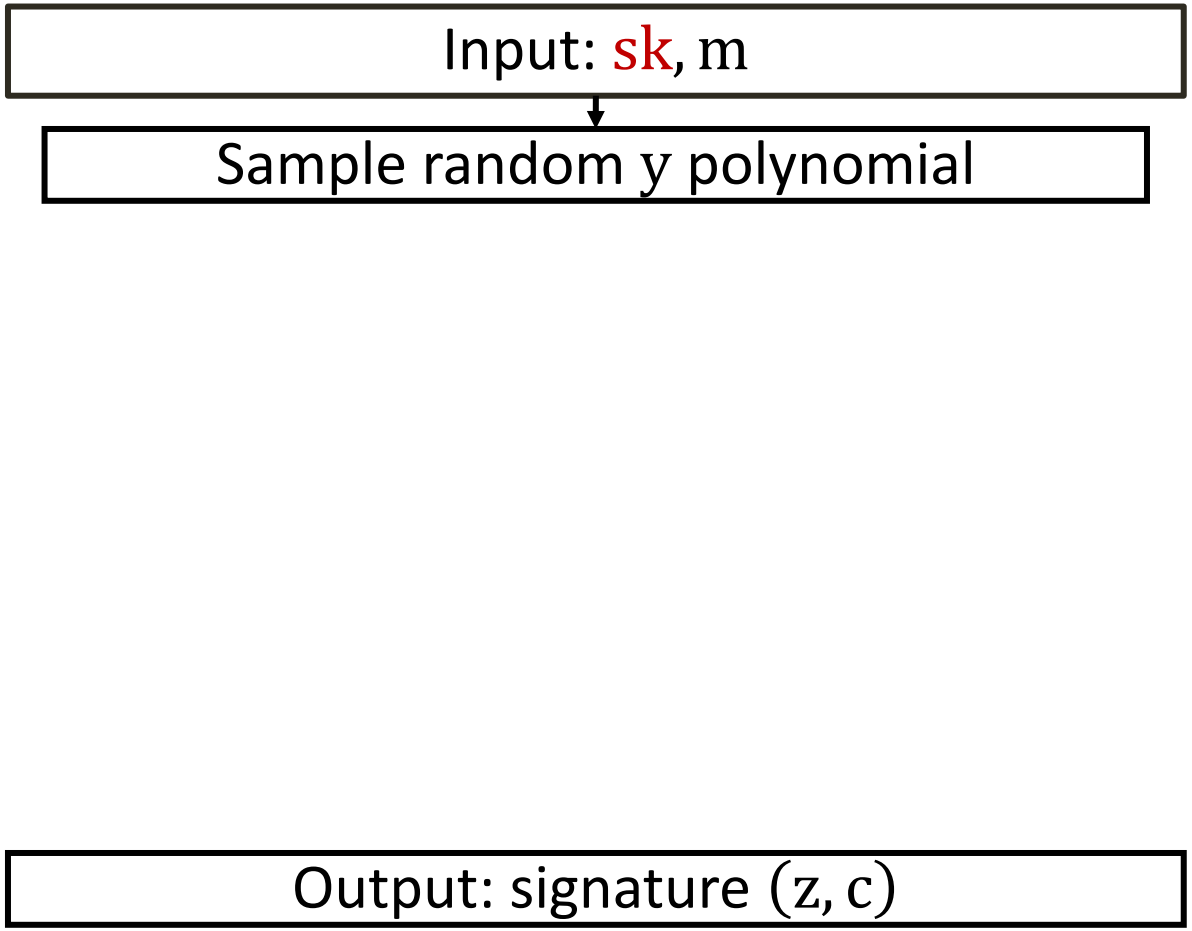
Signature generation

Input: sk, m

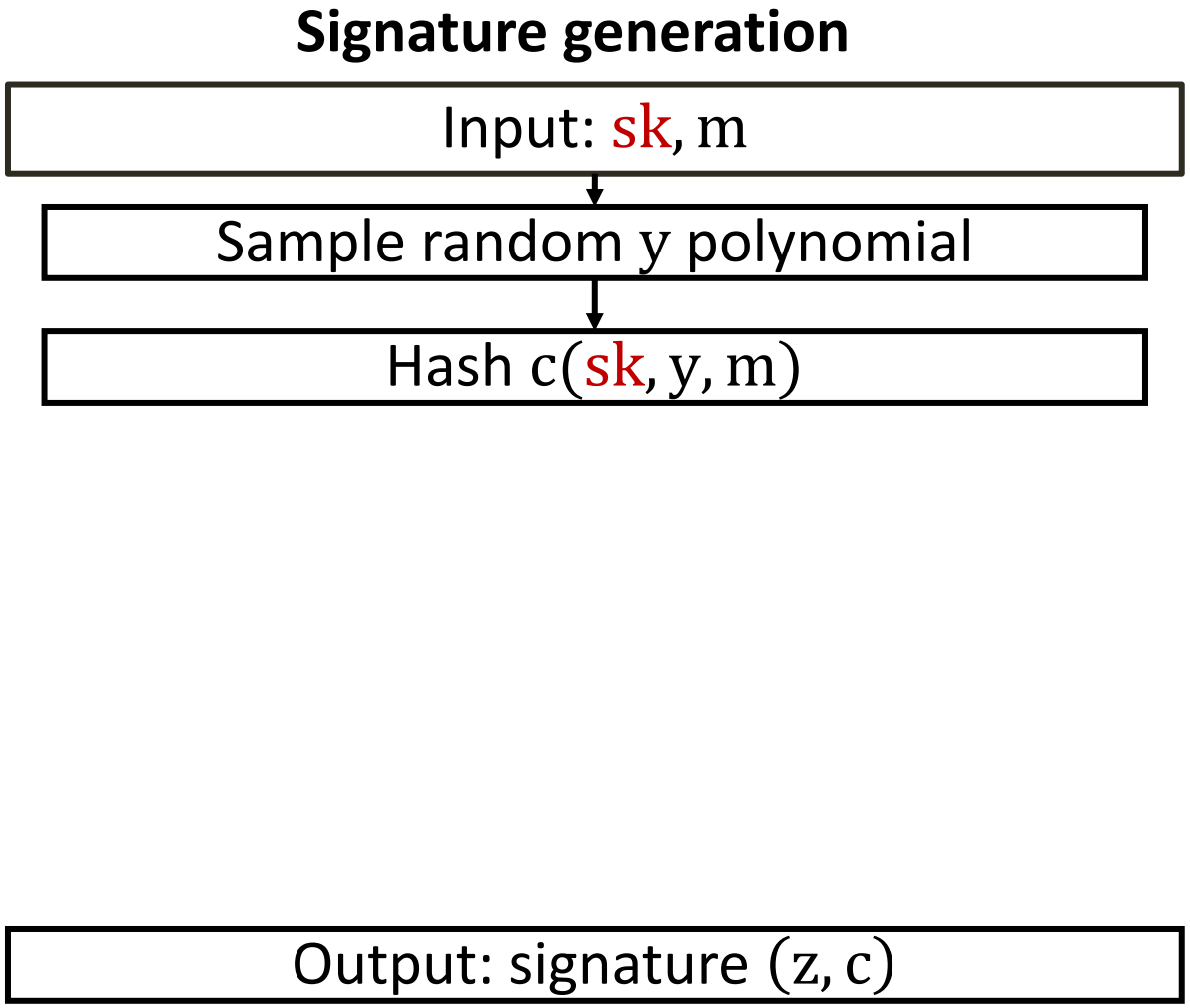
Output: signature (z, c)

qTESLA‘s sign and verify

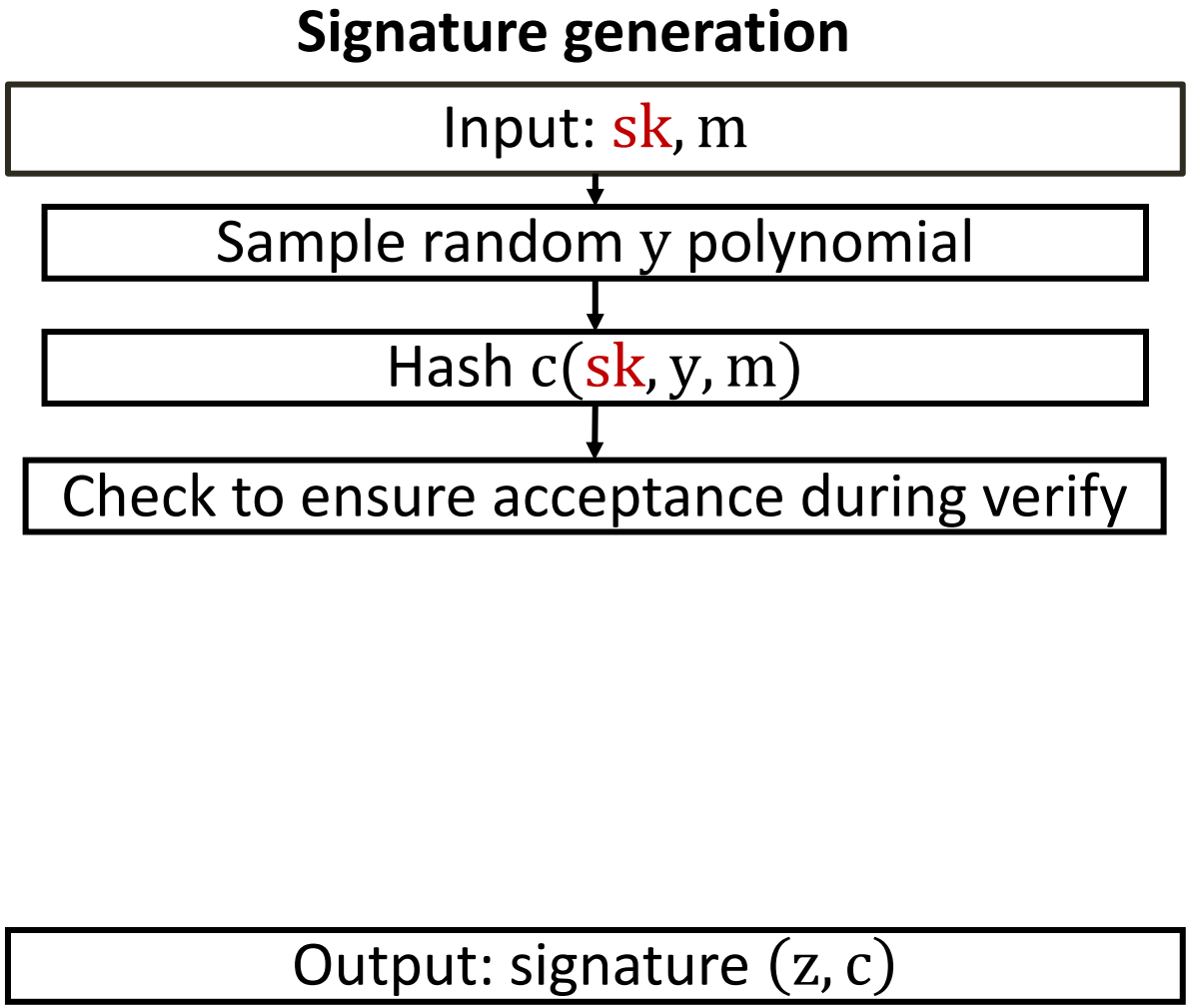
Signature generation



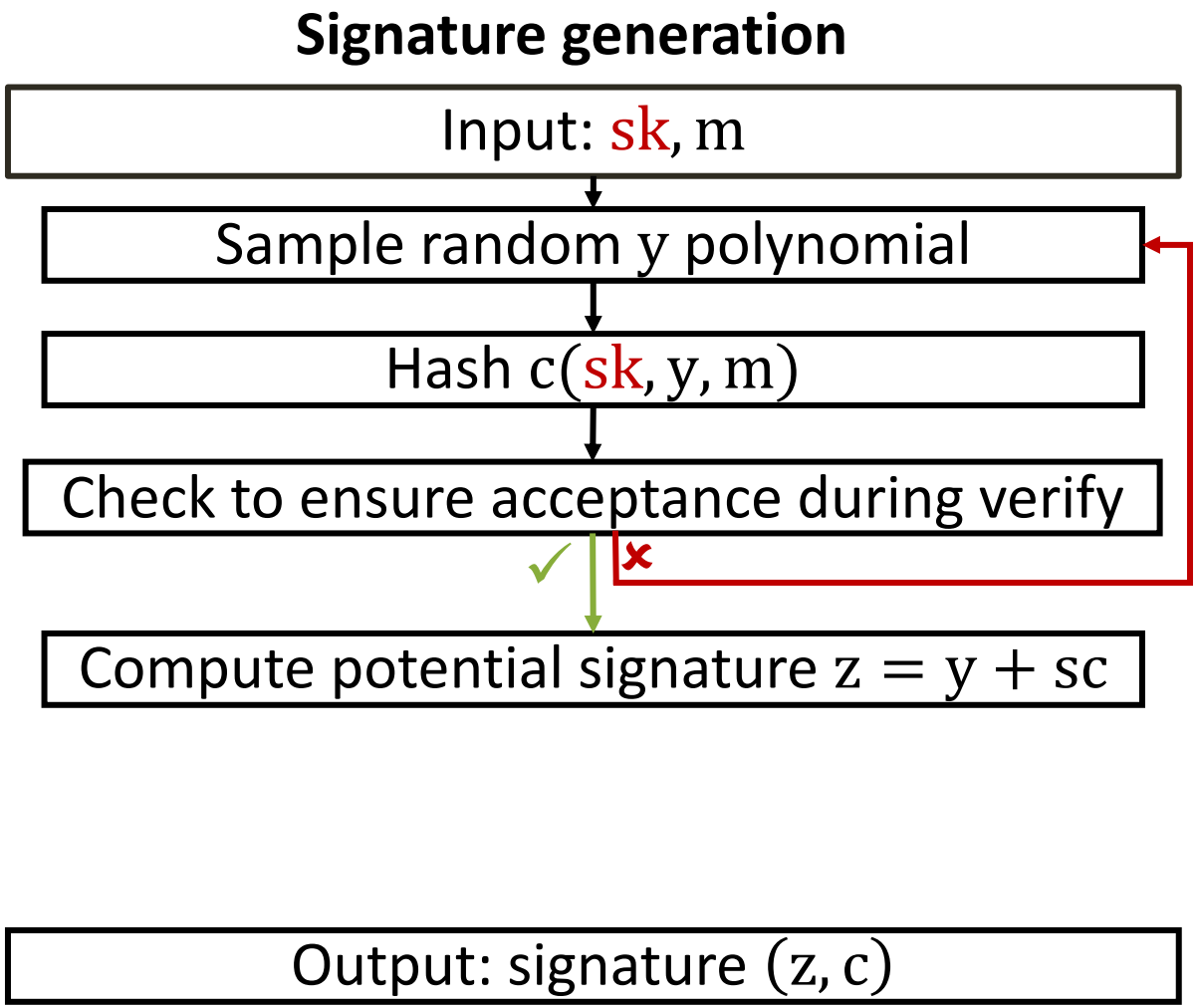
qTESLA‘s sign and verify



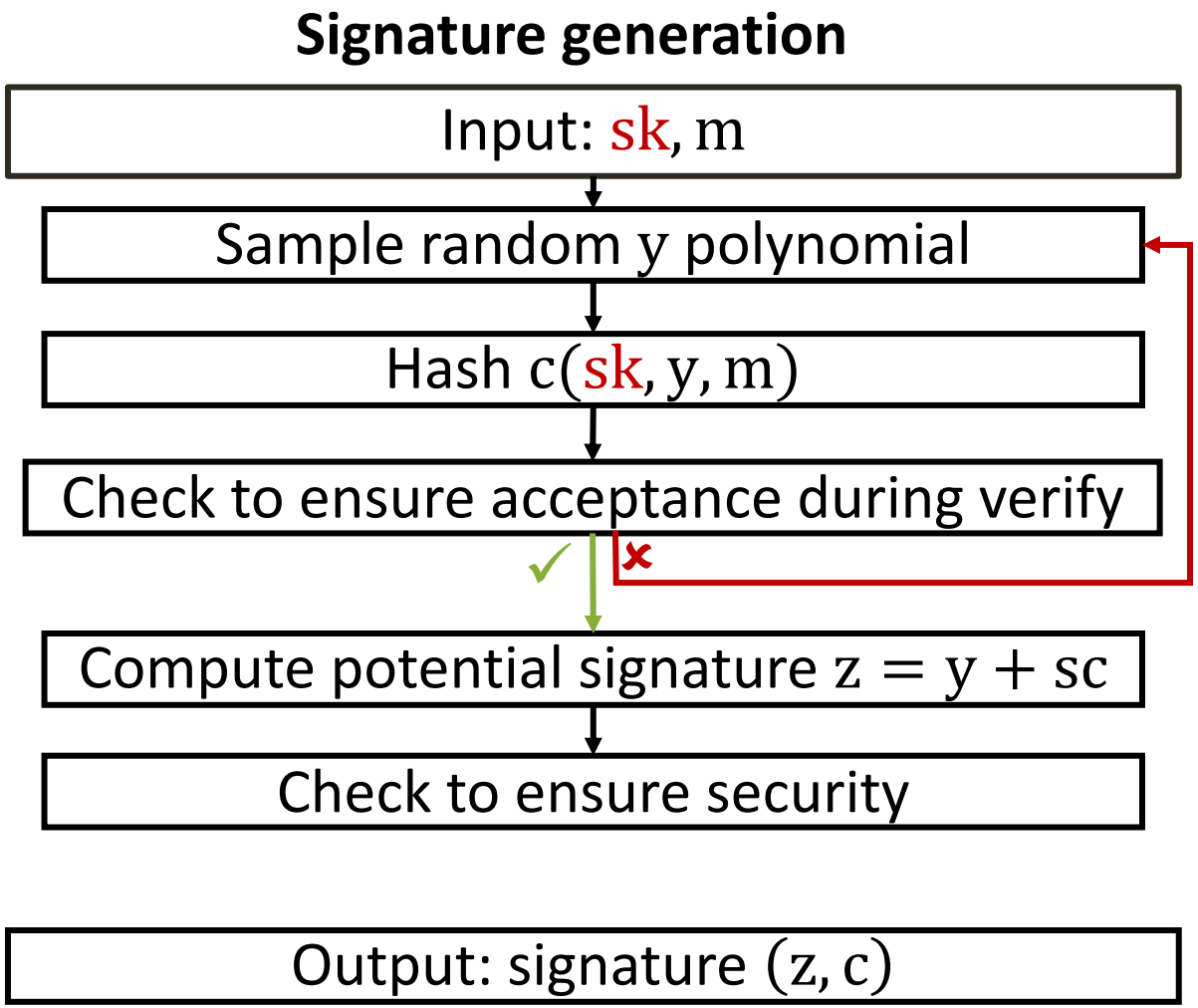
qTESLA's sign and verify



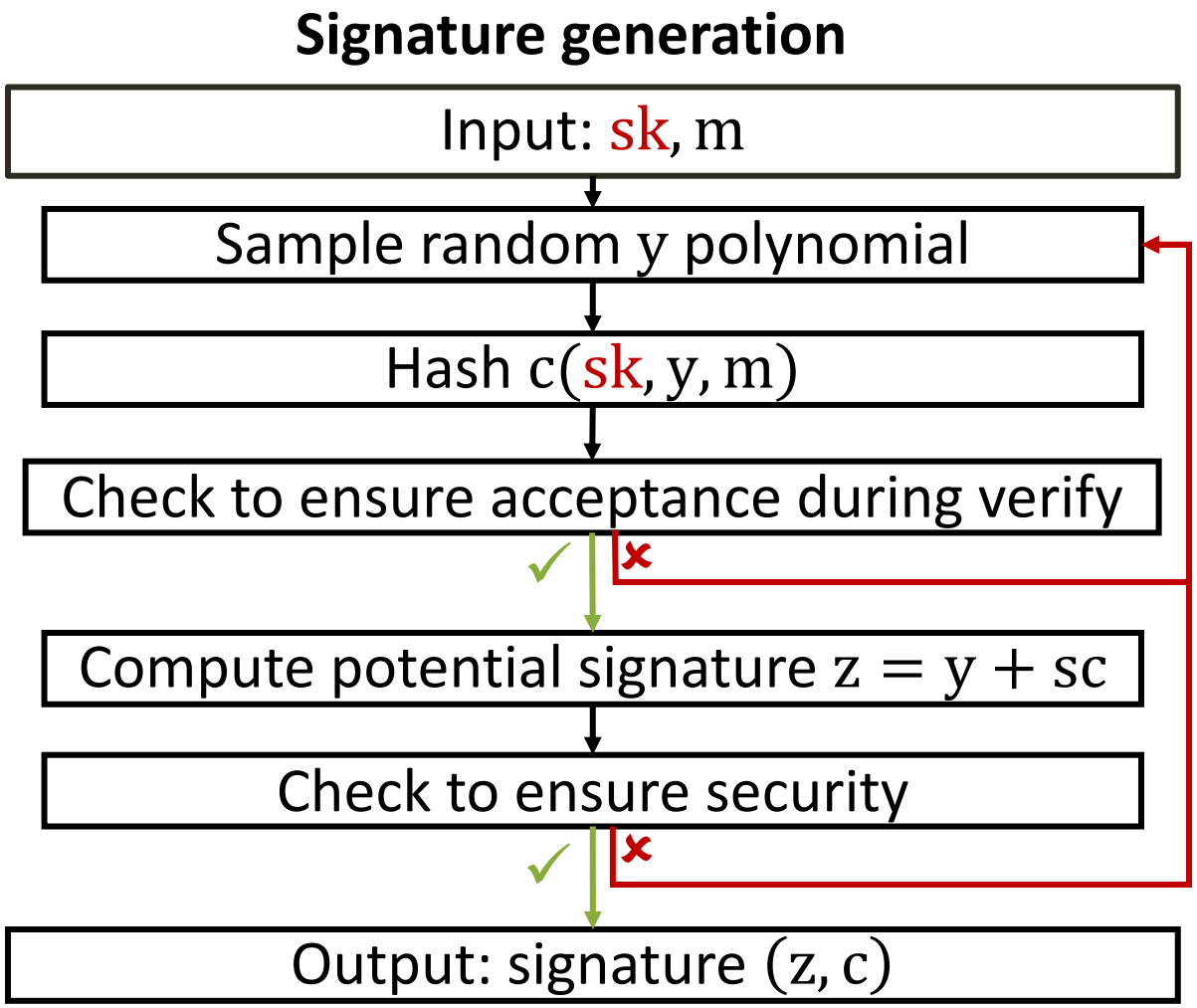
qTESLA's sign and verify



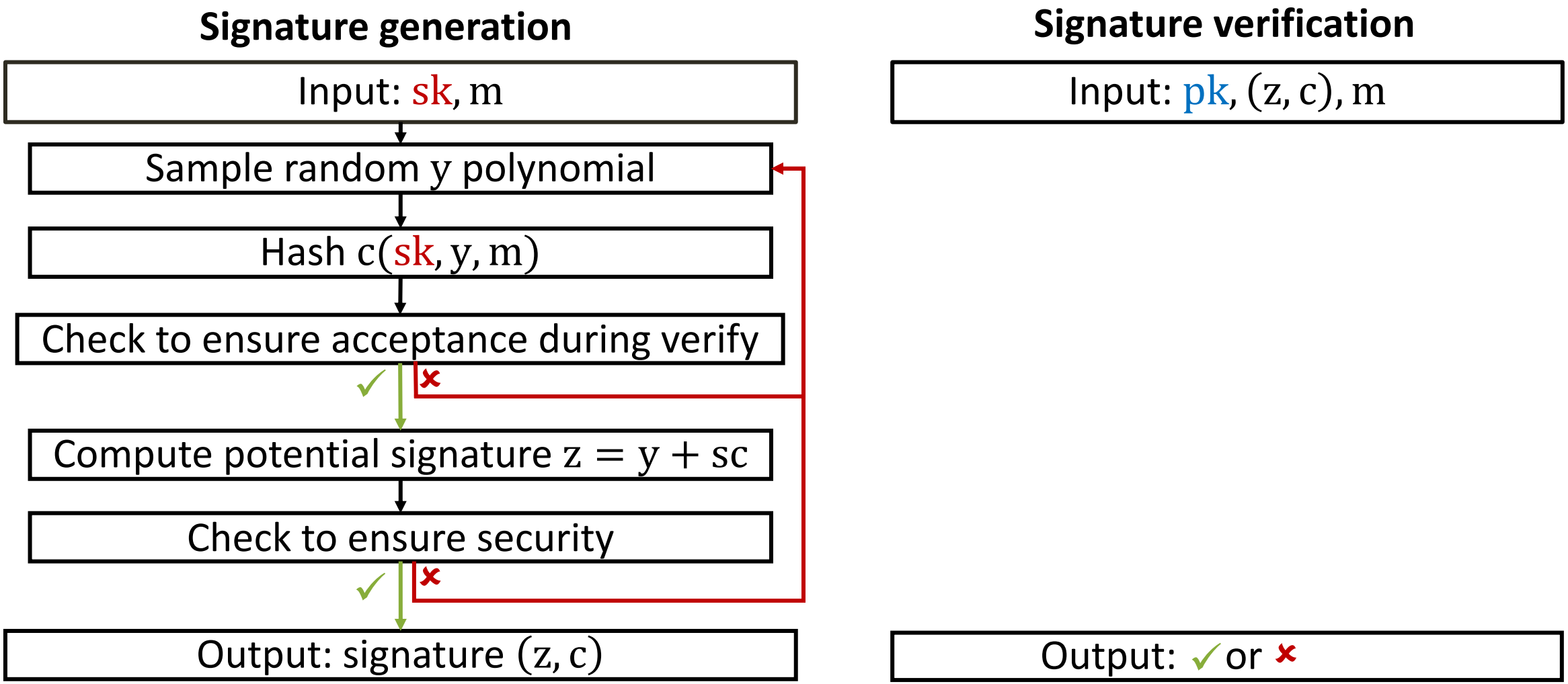
qTESLA's sign and verify



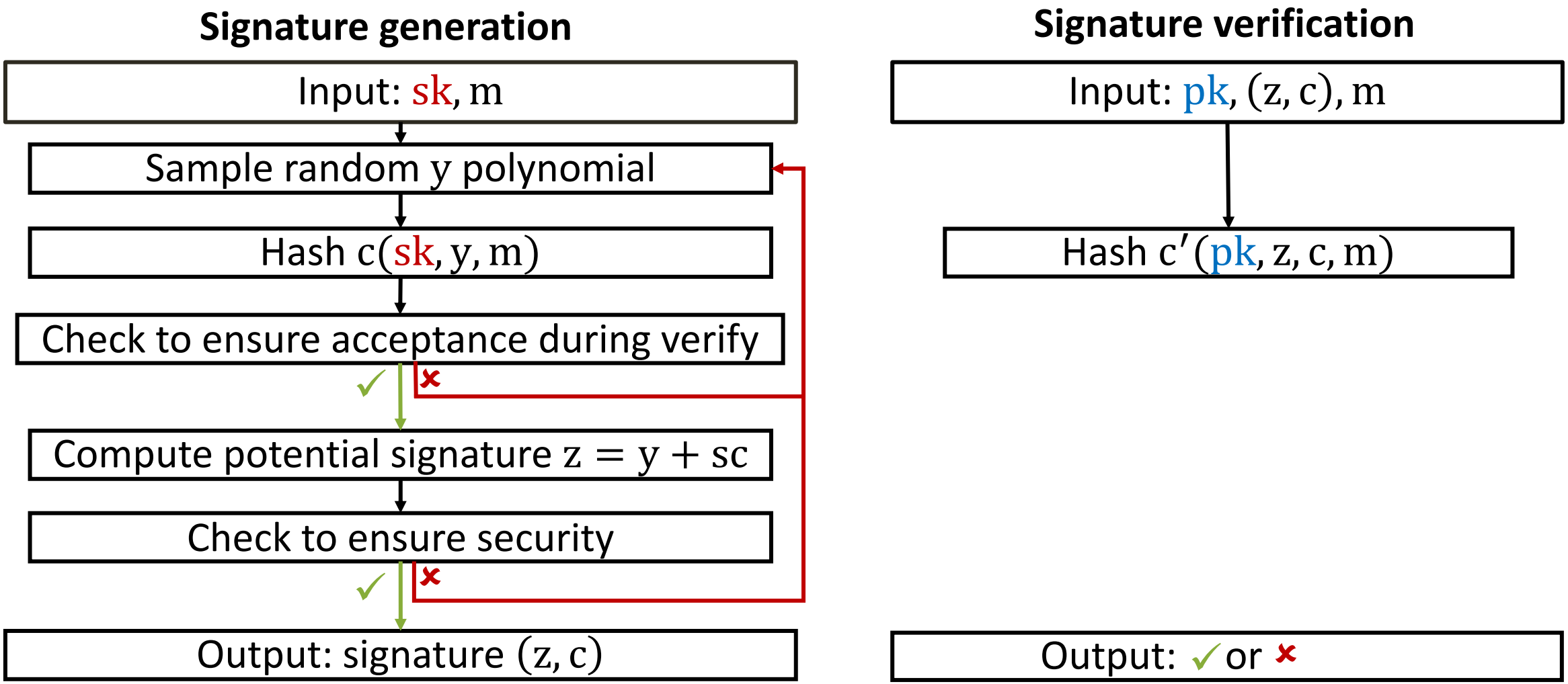
qTESLA's sign and verify



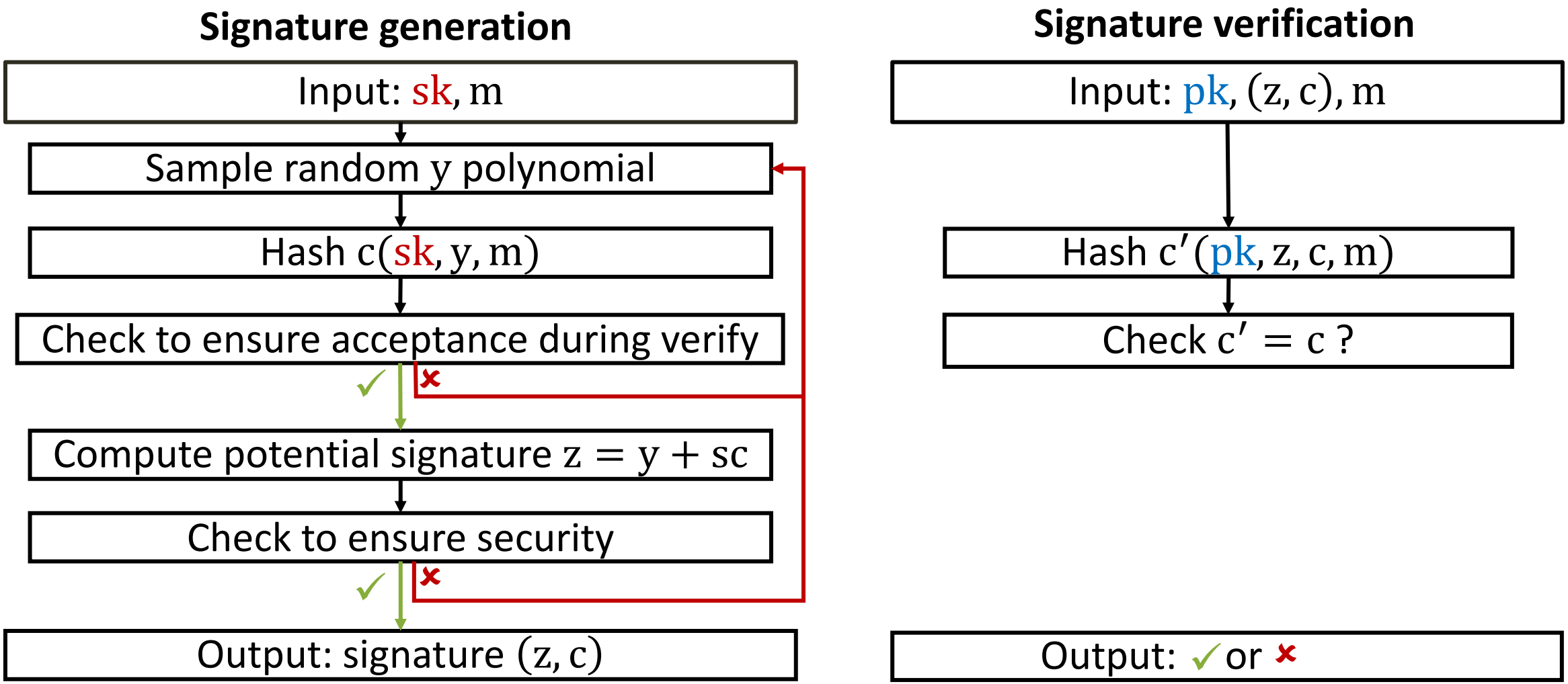
qTESLA's sign and verify



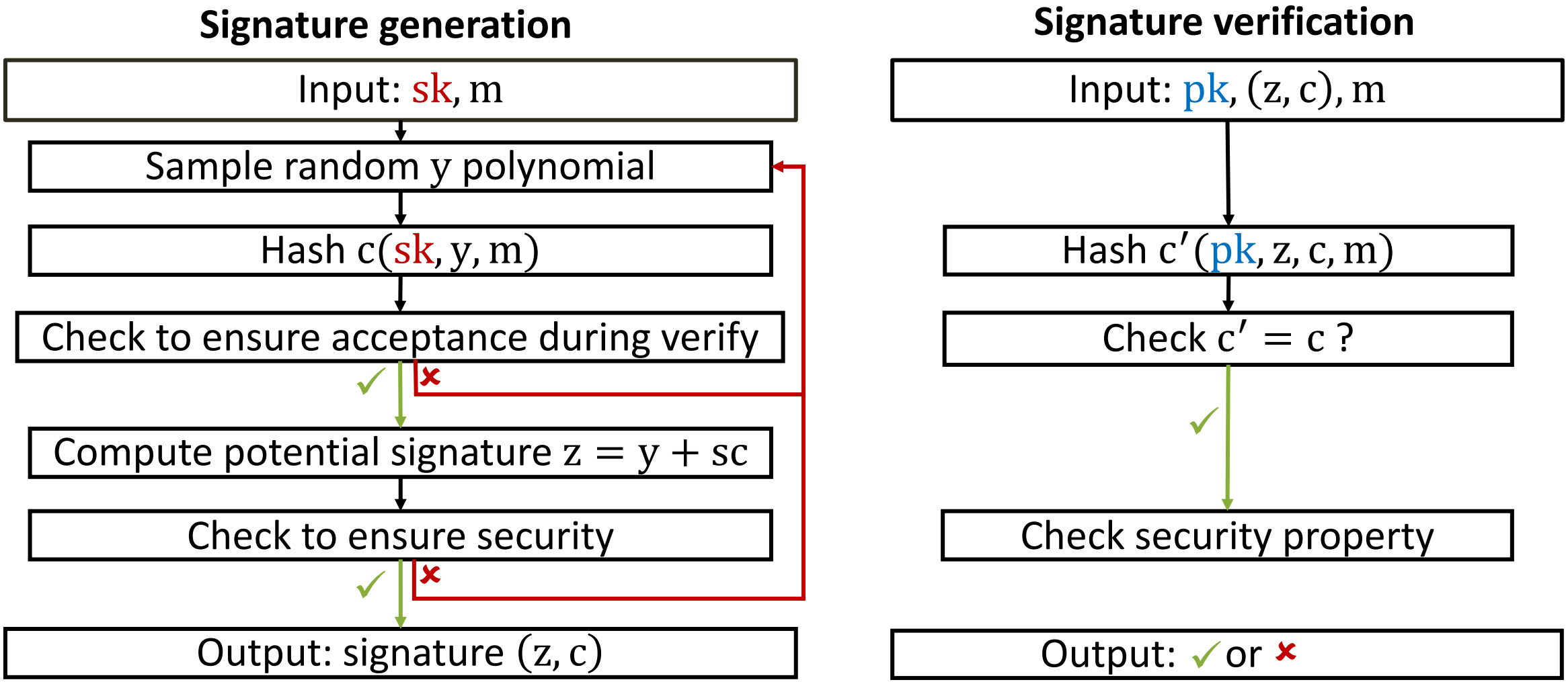
qTESLA's sign and verify



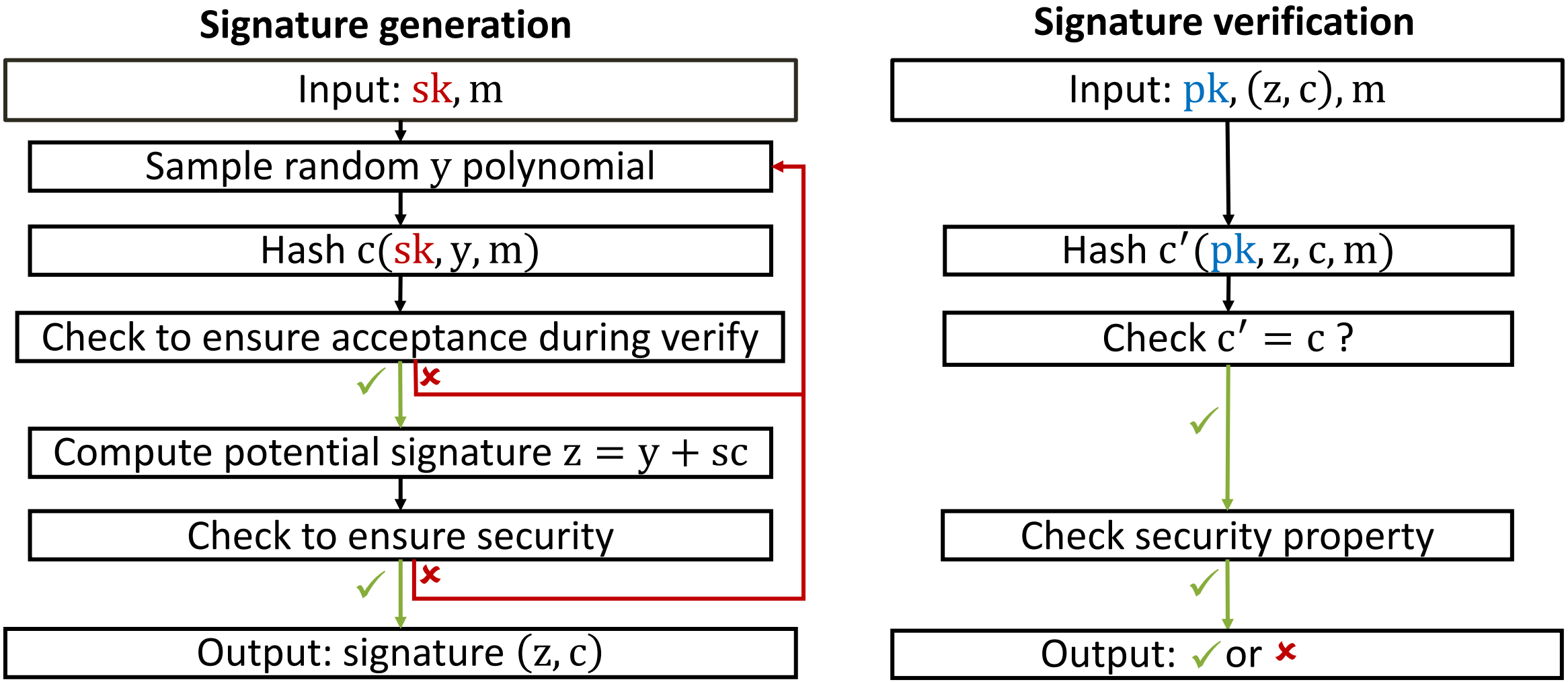
qTESLA's sign and verify



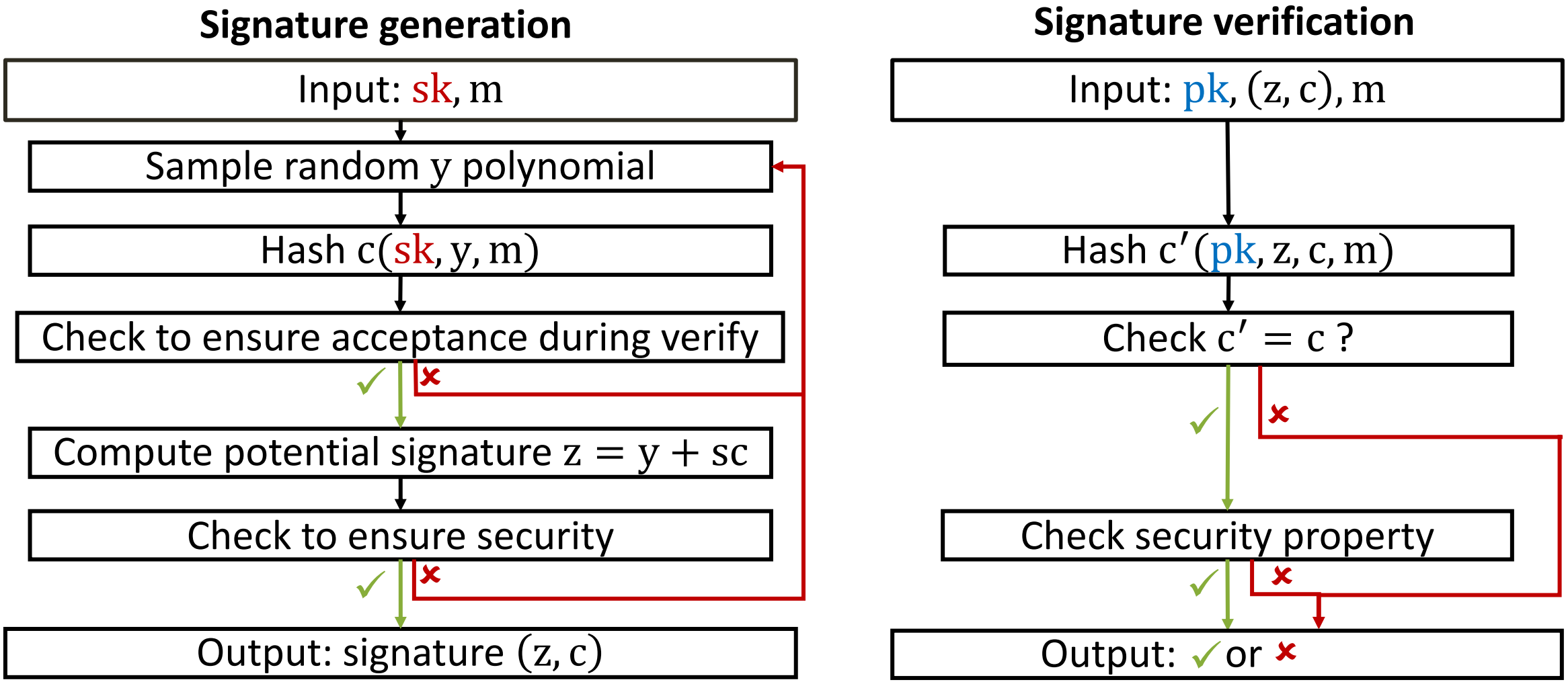
qTESLA's sign and verify



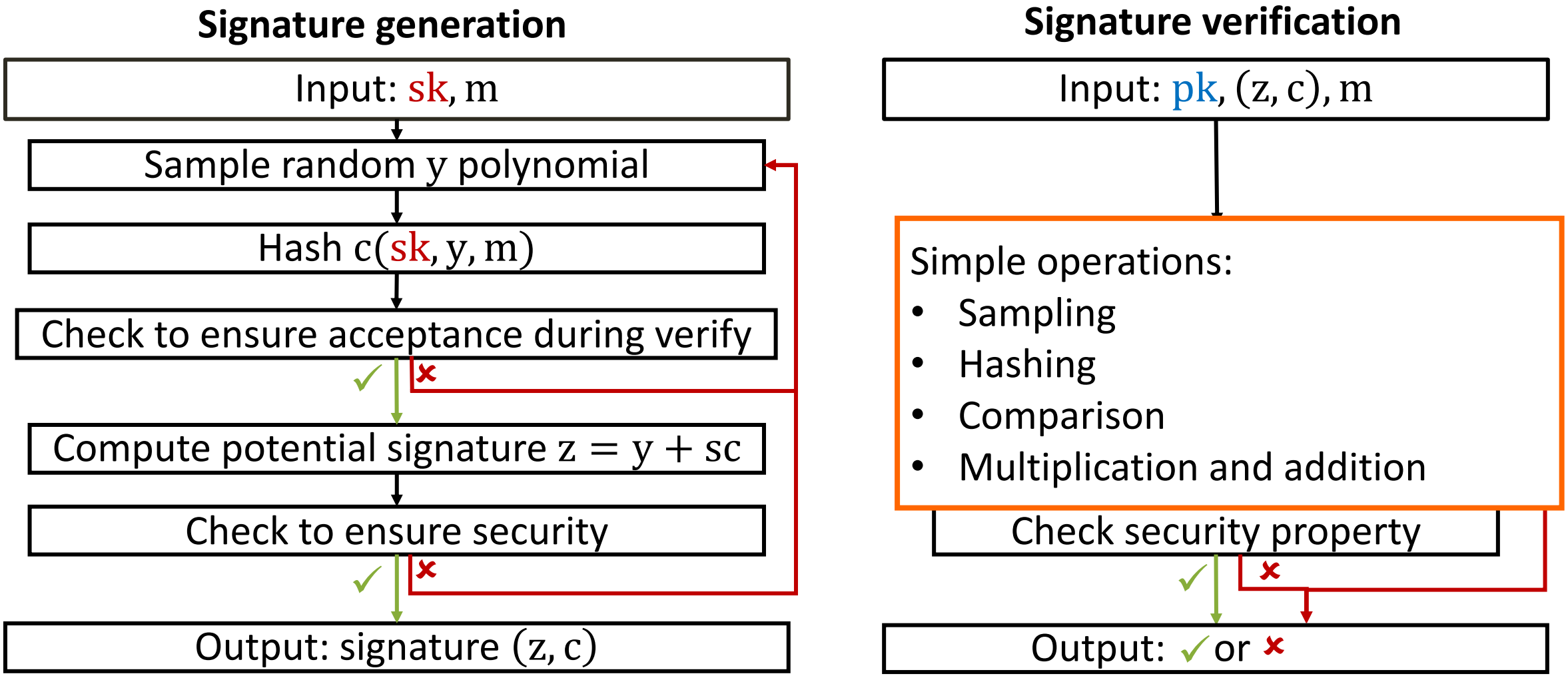
qTESLA's sign and verify



qTESLA's sign and verify

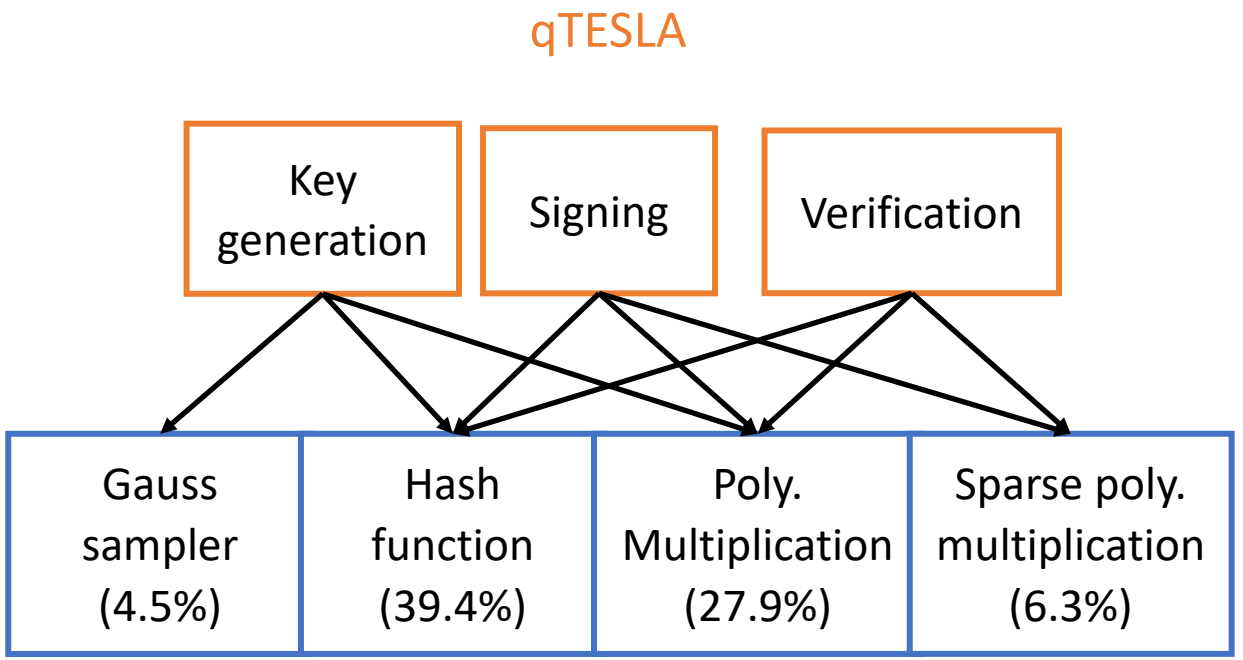


qTESLA's sign and verify



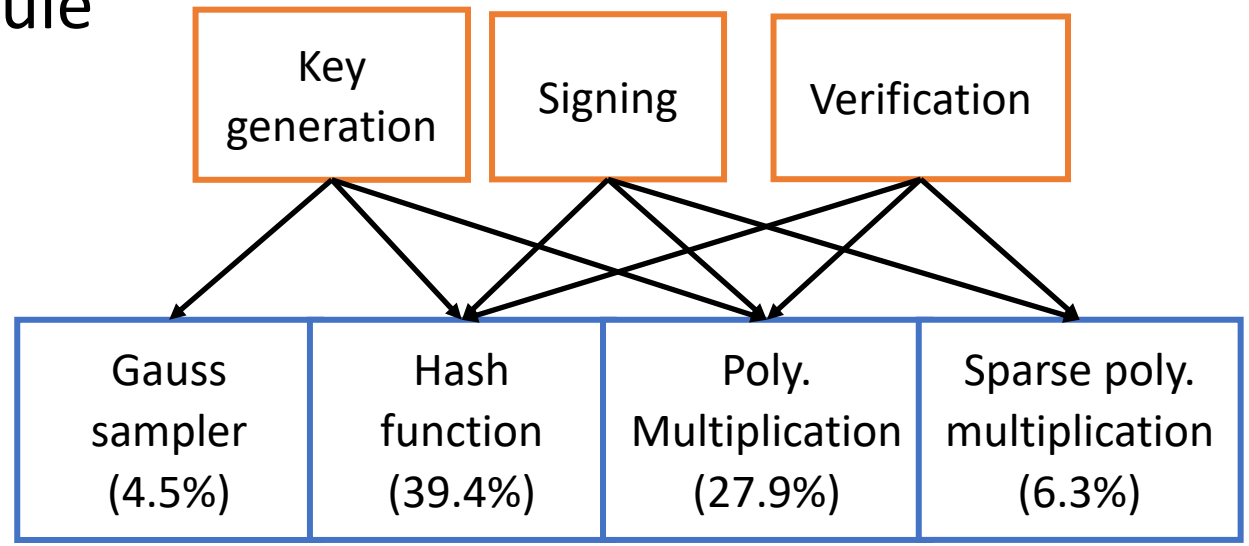
Hardware blocks for lattice-based schemes

Lattice-based hardware blocks



Lattice-based hardware blocks

- A unified hardware core for both SHAKE-128/256 and cSHAKE-128/256
- A novel, parameterized binary-search CDT sampler in hardware
- A novel, fully pipelined NTT-based polynomial multiplier
- A parameterized sparse polynomial multiplier qTESLA
- A lightweight Hmax-Sum module



Lattice-based hardware blocks

- A unified hardware core for both SHAKE-128/256 and cSHAKE-128/256
- A novel, parameterized binary-search CDT sampler in hardware
- A novel, fully pipelined NTT-based polynomial multiplier
- A parameterized sparse polynomial multiplier
- A lightweight Hmax-Sum module

A new binary-search CDT sampler

Input: a random number x of precision β generated by a PRNG

Output: a signed Gaussian sample s

- Pre-computed CDT table of depth t and width β
- Split CDT into two power-of-two parts, with the ending index of the first part and the starting index of the second part as: $end_1 = 2^{\lceil \log_2 t \rceil - 1} - 1, start_2 = t - 2^{\lceil \log_2 t \rceil - 1}$

If $x <$ CDT$[end_1 + 1]$ then

$min = 0, max = end_1 + 1$

Else

$min = start_2, max = t$

While $min + 1 \neq max$ do

if $x <$ CDT$[(min + max)/2]$ then

$max = (min + max)/2$

else

$min = (min + max)/2$

Return $s =$ MSB(x) ? $(-min) : min$

A new binary-search CDT sampler

Input: a random number x of precision β generated by a PRNG

Output: a signed Gaussian sample s

- Pre-computed CDT table of depth t and width β
- Split CDT into two power-of-two parts, with the ending index of the first part and the starting index of the second part as: $end_1 = 2^{\lceil \log_2 t \rceil - 1} - 1, start_2 = t - 2^{\lceil \log_2 t \rceil - 1}$

If $x <$ CDT$[end_1 + 1]$ then

$min = 0, max = end_1 + 1$

Else

$min = start_2, max = t$

While $min + 1 \neq max$ do

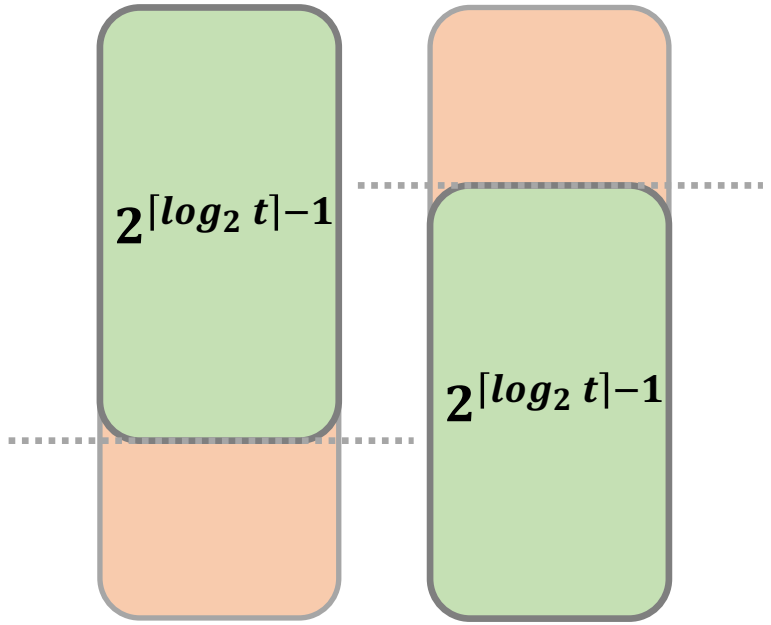
if $x <$ CDT$[(min + max)/2]$ then

$max = (min + max)/2$

else

$min = (min + max)/2$

Return $s =$ MSB(x) ? $(-min) : min$



A new binary-search CDT sampler

Input: a random number x of precision β generated by a PRNG

Output: a signed Gaussian sample s

- Pre-computed CDT table of depth t and width β
- Split CDT into two power-of-two parts, with the ending index of the first part and the starting index of the second part as: $end_1 = 2^{\lceil \log_2 t \rceil - 1} - 1, start_2 = t - 2^{\lceil \log_2 t \rceil - 1}$

If $x <$ CDT$[end_1 + 1]$ then

$min = 0, max = end_1 + 1$

Else

$min = start_2, max = t$

While $min + 1 \neq max$ do

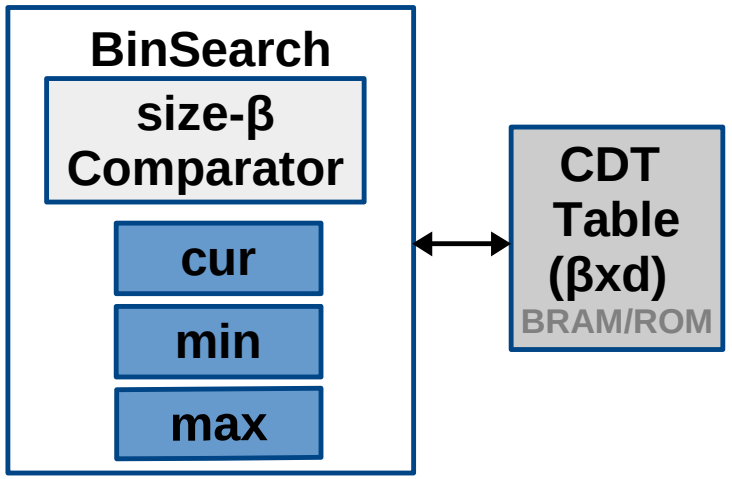
if $x <$ CDT$[(min + max)/2]$ then

$max = (min + max)/2$

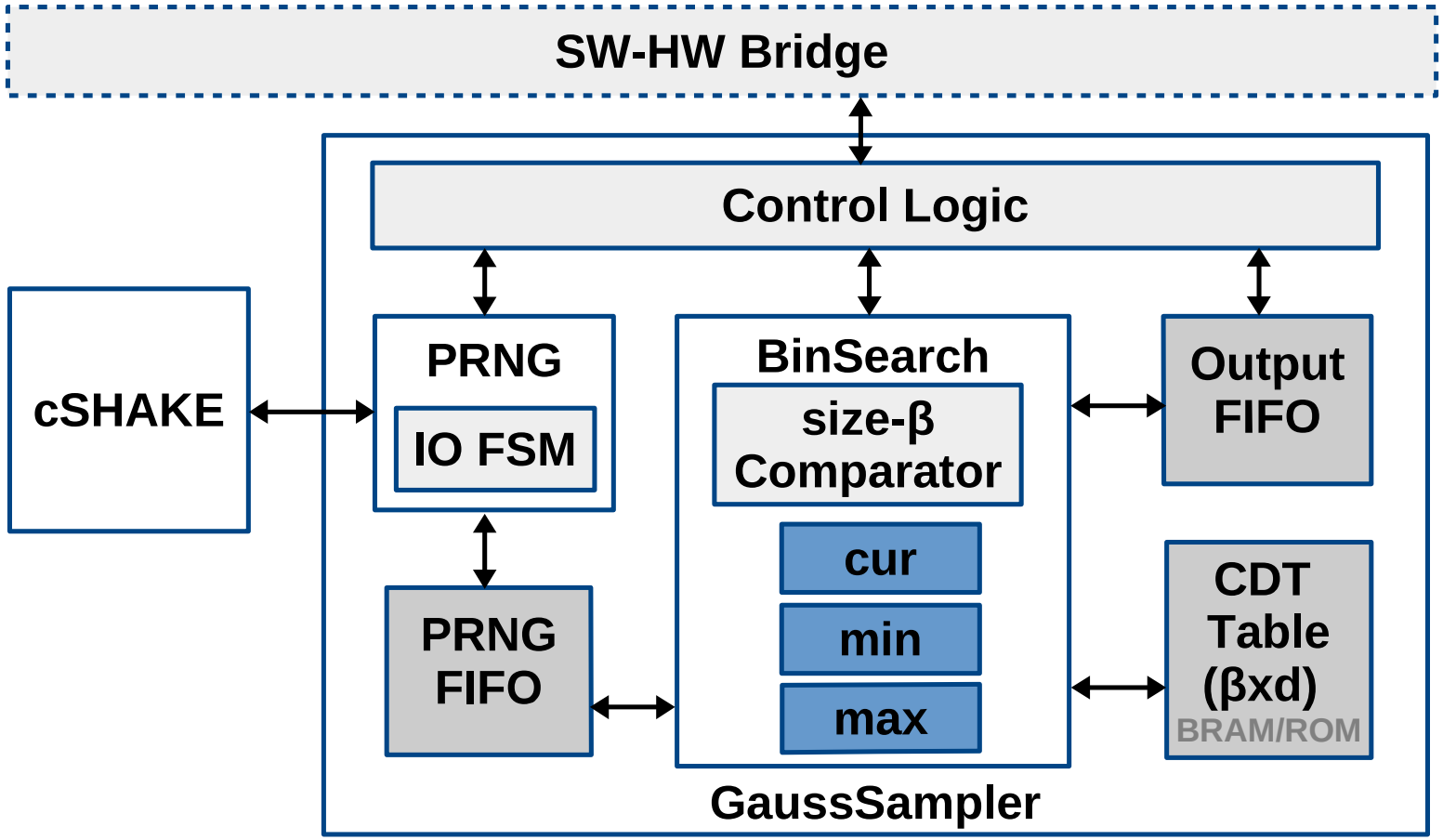
else

$min = (min + max)/2$

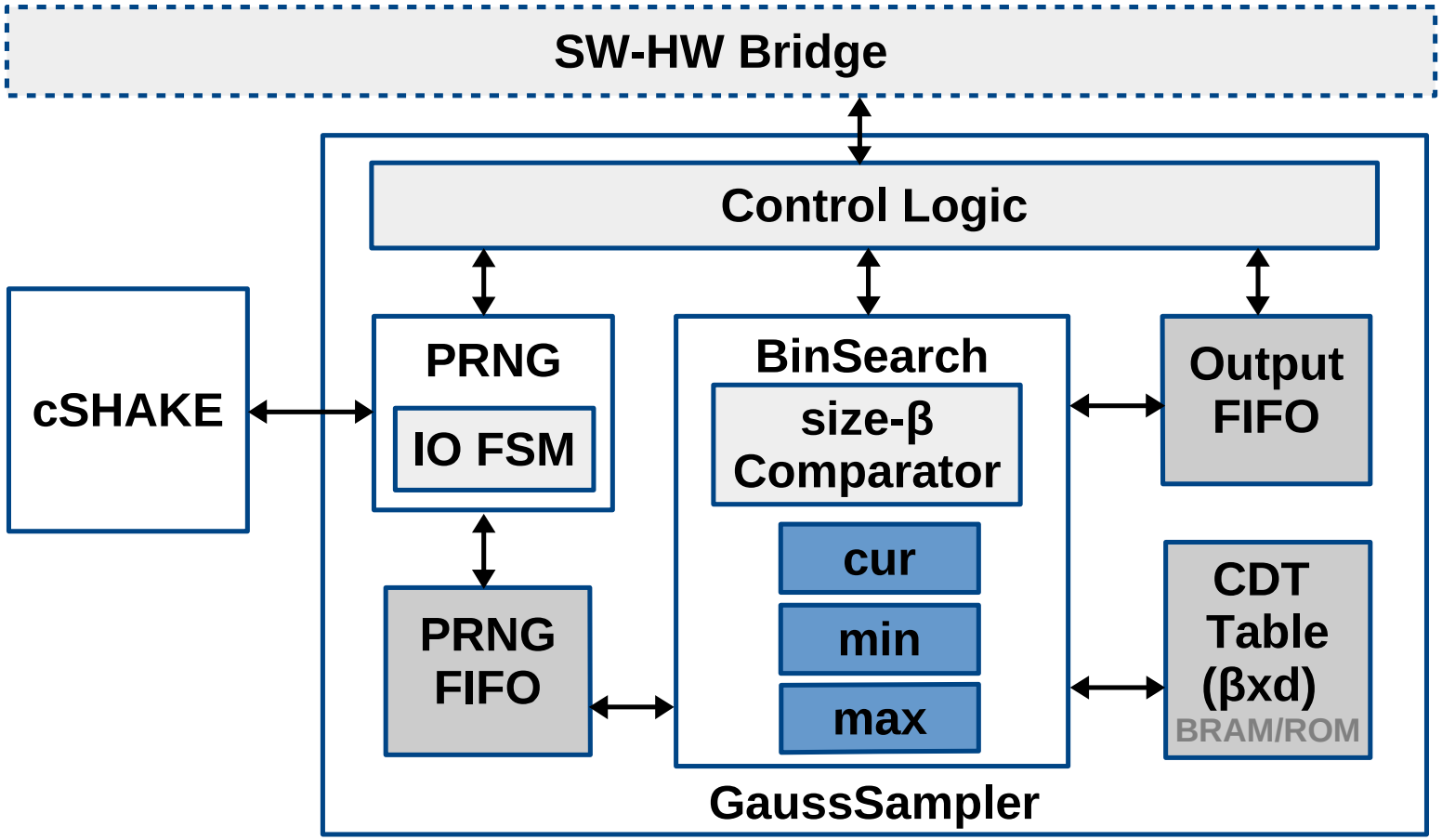
Return $s =$ MSB(x) ? $(-min)$: min



A new binary-search CDT sampler



A new binary-search CDT sampler



- ✓ Parameterized
  - σ: standard deviation
  - β: targeted precision
  - τ: tail-cut
  - b: batch size
- ✓ Wide Applicability
  - (small σ)
- ✓ Constant-time
- ✓ Lightweight
- ✓ Standard interface

Performance

Design	Batch size b	Device	Total cycles	PRNG cycles	Slices	LUTs	FFs	RAMs	Fmax (MHz)
Ours (qTESLA-p-I)	512	Artix-7	19,046	18,948	113	278	295	2.5	131
	1024	Artix-7	18,451	18,370	118	279	296	2.5	134
Ours (qTESLA-p-III)	512	Artix-7	83,040	82,952	217	485	487	4.5	128
	1024	Artix-7	81,904	81,860	191	450	487	4.5	123
	2048	Artix-7	81,335	81,314	191	470	490	4.5	123
Ours	512	Artix-7	9,506	9474	114	268	283	2.5	101
[HKR+18]	512	Virtex-7	2,560 (w/o PRNG)		15	53	17	1	193
[TWS19]	512	Artix-7	50,700 (w/o PRNG)		–	893	796	3	113

[HKR+18]: Howe, James, et al. "On practical discrete Gaussian samplers for lattice-based cryptography."
[TWS19]: Tian, Shanquan, et al. "Merge-exchange sort based discrete Gaussian sampler with fixed memory access pattern."

✓ Parameterized

Performance

Design	Batch size b	Device	Total cycles	PRNG cycles	Slices	LUTs	FFs	RAMs	Fmax (MHz)
Ours (qTESLA-p-I)	512	Artix-7	19,046	18,948	113	278	295	2.5	131
	1024	Artix-7	18,451	18,370	118	279	296	2.5	134
Ours (qTESLA-p-III)	512	Artix-7	83,040	82,952	217	485	487	4.5	128
	1024	Artix-7	81,904	81,860	191	450	487	4.5	123
	2048	Artix-7	81,335	81,314	191	470	490	4.5	123
Ours	512	Artix-7	9,506	9474	114	268	283	2.5	101
[HKR+18]	512	Virtex-7	2,560 (w/o PRNG)		15	53	17	1	193
[TWS19]	512	Artix-7	50,700 (w/o PRNG)		–	893	796	3	113

[HKR+18]: Howe, James, et al. "On practical discrete Gaussian samplers for lattice-based cryptography."
[TWS19]: Tian, Shanquan, et al. "Merge-exchange sort based discrete Gaussian sampler with fixed memory access pattern."

✓ Parameterized
✓ Lightweight

Performance

Design	Batch size b	Device	Total cycles	PRNG cycles	Slices	LUTs	FFs	RAMs	Fmax (MHz)
Ours (qTESLA-p-I)	512	Artix-7	19,046	18,948	113	278	295	2.5	131
	1024	Artix-7	18,451	18,370	118	279	296	2.5	134
Ours (qTESLA-p-III)	512	Artix-7	83,040	82,952	217	485	487	4.5	128
	1024	Artix-7	81,904	81,860	191	450	487	4.5	123
	2048	Artix-7	81,335	81,314	191	470	490	4.5	123
Ours	512	Artix-7	9,506	9474	114	268	283	2.5	101
[HKR+18]	512	Virtex-7	2,560 (w/o PRNG)		15	53	17	1	193
[TWS19]	512	Artix-7	50,700 (w/o PRNG)		–	893	796	3	113

[HKR+18]: Howe, James, et al. "On practical discrete Gaussian samplers for lattice-based cryptography."
[TWS19]: Tian, Shanquan, et al. "Merge-exchange sort based discrete Gaussian sampler with fixed memory access pattern."

Performance

- ✓ Parameterized
- ✓ Lightweight
- ✓ Computation cycles perfectly hidden by PRNG
- ✓ Cryptographically strong cSHAKE

Design	Batch size b	Device	Total cycles	PRNG cycles	Slices	LUTs	FFs	RAMs	Fmax (MHz)
Ours (qTESLA-p-I)	512	Artix-7	19,046	18,948	113	278	295	2.5	131
	1024	Artix-7	18,451	18,370	118	279	296	2.5	134
Ours (qTESLA-p-III)	512	Artix-7	83,040	82,952	217	485	487	4.5	128
	1024	Artix-7	81,904	81,860	191	450	487	4.5	123
	2048	Artix-7	81,335	81,314	191	470	490	4.5	123
Ours	512	Artix-7	9,506	9474	114	268	283	2.5	101
[HKR+18]	512	Virtex-7	2,560 (w/o PRNG)		15	53	17	1	193
[TWS19]	512	Artix-7	50,700 (w/o PRNG)		–	893	796	3	113

[HKR+18]: Howe, James, et al. "On practical discrete Gaussian samplers for lattice-based cryptography."
[TWS19]: Tian, Shanquan, et al. "Merge-exchange sort based discrete Gaussian sampler with fixed memory access pattern."

Performance

Related work:

✗ Hard to port to other platforms/applications

No support for standard interface

Sequential search and PRNG, no synchronization between modules

✗ Less cryptographically secure PRNG

Design	Batch size b	Device	Total cycles	PRNG cycles	Slices	LUTs	FFs	RAMs	Fmax (MHz)
Ours (qTESLA-p-I)	512	Artix-7	19,046	18,948	113	278	295	2.5	131
	1024	Artix-7	18,451	18,370	118	279	296	2.5	134
Ours (qTESLA-p-III)	512	Artix-7	83,040	82,952	217	485	487	4.5	128
	1024	Artix-7	81,904	81,860	191	450	487	4.5	123
	2048	Artix-7	81,335	81,314	191	470	490	4.5	123
Ours	512	Artix-7	9,506	9,474	114	268	283	2.5	101
[HKR+18]	512	Virtex-7	2,560 (w/o PRNG)		15	53	17	1	193
[TWS19]	512	Artix-7	50,700 (w/o PRNG)		–	893	796	3	113

[HKR+18]: Howe, James, et al. "On practical discrete Gaussian samplers for lattice-based cryptography."
[TWS19]: Tian, Shanquan, et al. "Merge-exchange sort based discrete Gaussian sampler with fixed memory access pattern."

NTT-based polynomial multiplier

Unified NTT algorithm

Forward NTT	Inverse NTT
Cooley-Tukey (CT) butterfly	CT butterfly

✓ One hardware module

✗ Extra computations
- Pre-scaling
- Bit-reversal
- Post-scaling

Separated NTT algorithm

Forward NTT	Inverse NTT
CT butterfly	Gentlemen-Sande (GS) butterfly

✓ No extra computations

✗ Two hardware modules

NTT-based polynomial multiplier

Our approach

Unified NTT algorithm		**Separated NTT algorithm**		**CT-GS NTT algorithm**	
Forward NTT	Inverse NTT	Forward NTT	Inverse NTT	Forward NTT	Inverse NTT
Cooley-Tukey (CT) butterfly	CT butterfly	CT butterfly	Gentlemen-Sande (GS) butterfly	CT-GS butterfly	

✓ One hardware module

✗ Extra computations
- Pre-scaling
- Bit-reversal
- Post-scaling

✓ No extra computations

✗ Two hardware modules

✓ No extra computations

✓ One hardware module

Memory efficient CT-GS NTT algorithm

Input: $a = \sum_{i=0}^{n-1} a_i x^i \in R_q$, with $a_i \in \mathbb{Z}_q$; pre-computed twiddle factors W

Output: $NTT(a)$ or $NTT^{-1}(a) \in R_q$

$m_0 = n/2$ or 1; $m_1 = 1/2$ or 2; $n_0 = 1$ or 0; $n_1 = n$ or $n/2$

$k = 0, j = 0$

Unified CT-GS algorithm → One hardware module

For $m = m_0; n_0 < m < n_1; m = m \cdot m_1$ do

For $i = 0; i < n/2; i = j + m/2$ do

$w = W[k]$

For $j = i; j < i + m/2; j = j + 1$ do

$(t_1, u_1) = (a[j], a[j + m])$;	$(t_2, u_2) = (a[j + m \cdot m_1], a[j + m + m \cdot m_1])$
$r_1 = w \cdot u_1$ or $w \cdot (t_1 - u_1)$;	$r_2 = w \cdot u_2$ or $w \cdot (t_2 - u_2)$
$a[j] = t_1 + r_1$ *or* $t_1 + u_1$;	$a[j + m] = t_1 - r_1$ *or* r_1
$a[j + m \cdot m_1] = t_2 + r_2$ *or* $t_2 + u_2$;	$a[j + m + m \cdot m_1] = t_2 - r_2$ *or* r_2
$mem[j] = (a[j], a[j + m \cdot m_1])$;	$mem[j + m \cdot m_1] = (a[j + m], a[j + m + m \cdot m_1])$

$k = k + 1$

// repeat for the Last NTT round

Return a

Memory efficient CT-GS NTT algorithm

Input: $a = \sum_{i=0}^{n-1} a_i x^i \in R_q$, with $a_i \in \mathbb{Z}_q$; pre-computed twiddle factors W

Output: $NTT(a)$ or $NTT^{-1}(a) \in R_q$

$m_0 = n/2$ or 1; $m_1 = 1/2$ or 2; $n_0 = 1$ or 0; $n_1 = n$ or $n/2$

$k = 0, j = 0$

For $m = m_0; n_0 < m < n_1; m = m \cdot m_1$ do

  For $i = 0; i < n/2; i = j + m/2$ do

    $w = W[k]$

    For $j = i; j < i + m/2; j = j + 1$ do

      $(t_1, u_1) = (a[j], a[j + m])$; $(t_2, u_2) = (a[j + m \cdot m_1], a[j + m + m \cdot m_1])$

      $r_1 = w \cdot u_1$ or $w \cdot (t_1 - u_1)$; $r_2 = w \cdot u_2$ or $w \cdot (t_2 - u_2)$

      $a[j] = t_1 + r_1$ *or* $t_1 + u_1$; $a[j + m] = t_1 - r_1$ *or* r_1

      $a[j + m \cdot m_1] = t_2 + r_2$ *or* $t_2 + u_2$; $a[j + m + m \cdot m_1] = t_2 - r_2$ *or* r_2

      $mem[j] = (a[j], a[j + m \cdot m_1])$; $mem[j + m \cdot m_1] = (a[j + m], a[j + m + m \cdot m_1])$

    $k = k + 1$

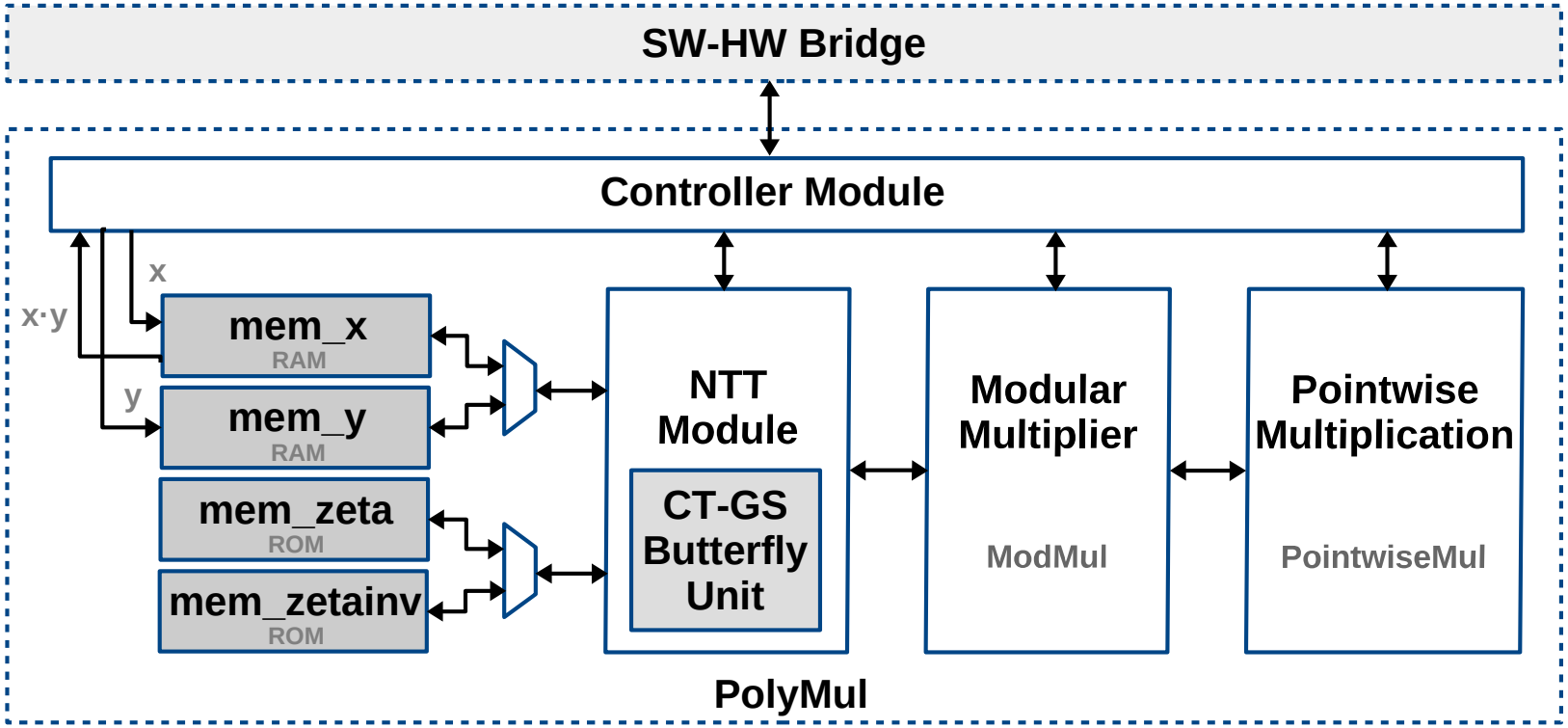
// repeat for the Last NTT round

Return a

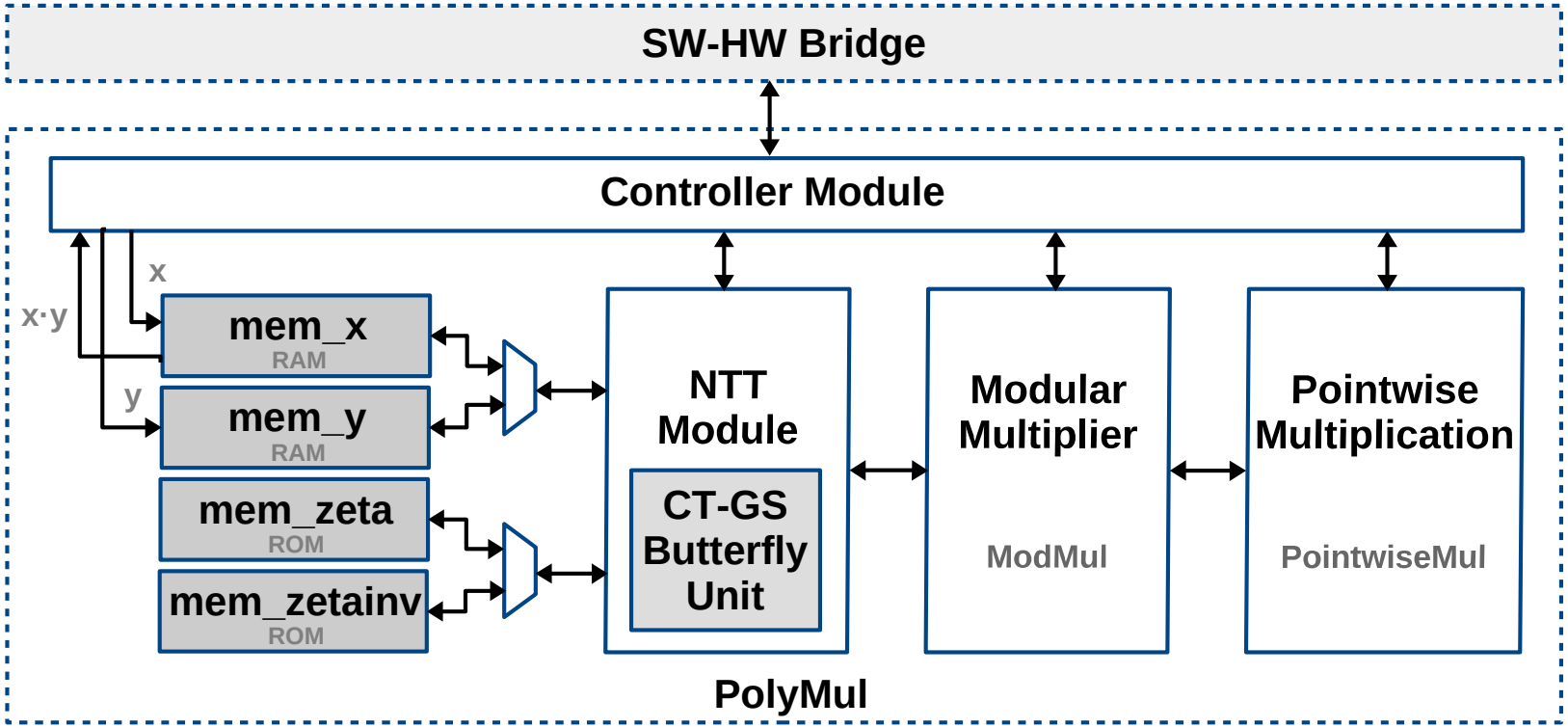
Unified CT-GS algorithm → One hardware module

Efficient memory access scheme → Memory efficiency
Fully pipelined architecture

NTT-based polynomial multiplier



NTT-based polynomial multiplier



- ✓ Parameterized
  - n: polynomial length
  - q: modulus
- ✓ Wide applicability
  - $(q \equiv 1 \; mod \; 2n)$
- ✓ Fully pipelined
- ✓ Constant-time
- ✓ Standard interface

Performance

Design	Parameters	Tunable (n, q)	Platform	Cycles	Slices	LUTs	FFs	DSPs	Fmax (MHz)
Ours	(1024, 343576577)	$\surd, \surd$	Artix-7	11,455	582	1977	991	6	124
	(2048, 856145921)		Artix-7	24,785	555	1981	1021	8	124
Ours	(1024, 12289)	$\surd, \surd$	Artix-7	11,455	271	944	467	3	141
[KLC+17]		$\surd, -$	Artix-7	5494	–	2832	1381	8	150
Ours	(2048, 65537)	$\surd, \surd$	Spartan-6	24,785	543	1601	368	8	90
[DB16]		$-, -$	Spartan-6	25,654	269	–	–	9	207

[HKR+18]: Howe, James, et al. "On practical discrete Gaussian samplers for lattice-based cryptography."
[TWS19]: Tian, Shanquan, et al. "Merge-exchange sort based discrete Gaussian sampler with fixed memory access pattern."

Performance

✓ Parameterized

Design	Parameters	Tunable (n, q)	Platform	Cycles	Slices	LUTs	FFs	DSPs	Fmax (MHz)
Ours	(1024, 343576577)	√, √	Artix-7	11,455	582	1977	991	6	124
	(2048, 856145921)		Artix-7	24,785	555	1981	1021	8	124
Ours	(1024, 12289)	√, √	Artix-7	11,455	271	944	467	3	141
[KLC+17]		√, –	Artix-7	5494	–	2832	1381	8	150
Ours	(2048, 65537)	√, √	Spartan-6	24,785	543	1601	368	8	90
[DB16]		–, –	Spartan-6	25,654	269	–	–	9	207

[HKR+18]: Howe, James, et al. "On practical discrete Gaussian samplers for lattice-based cryptography."
[TWS19]: Tian, Shanquan, et al. "Merge-exchange sort based discrete Gaussian sampler with fixed memory access pattern."

Performance

- ✓ Parameterized
- ✓ Achieves theoretical cycles limit $(n \cdot \log_2 n + \frac{n}{2})$

Design	Parameters	Tunable (n, q)	Platform	Cycles	Slices	LUTs	FFs	DSPs	Fmax (MHz)
Ours	(1024, 343576577)	√, √	Artix-7	11,455	582	1977	991	6	124
	(2048, 856145921)		Artix-7	24,785	555	1981	1021	8	124
Ours	(1024, 12289)	√, √	Artix-7	11,455	271	944	467	3	141
[KLC+17]		√, –	Artix-7	5494	–	2832	1381	8	150
Ours	(2048, 65537)	√, √	Spartan-6	24,785	543	1601	368	8	90
[DB16]		–, –	Spartan-6	25,654	269	–	–	9	207

[HKR+18]: Howe, James, et al. "On practical discrete Gaussian samplers for lattice-based cryptography."
[TWS19]: Tian, Shanquan, et al. "Merge-exchange sort based discrete Gaussian sampler with fixed memory access pattern."

Performance

- ✓ Parameterized
- ✓ Achieves theoretical cycles limit $(n \cdot \log_2 n + \frac{n}{2})$
- ✓ Good time-area product

Design	Parameters	Tunable (n, q)	Platform	Cycles	Slices	LUTs	FFs	DSPs	Fmax (MHz)
Ours	(1024, 343576577)	√, √	Artix-7	11,455	582	1977	991	6	124
	(2048, 856145921)		Artix-7	24,785	555	1981	1021	8	124
Ours	(1024, 12289)	√, √	Artix-7	11,455	271	944	467	3	141
[KLC+17]		√, –	Artix-7	5494	–	2832	1381	8	150
Ours	(2048, 65537)	√, √	Spartan-6	24,785	543	1601	368	8	90
[DB16]		–, –	Spartan-6	25,654	269	–	–	9	207

[HKR+18]: Howe, James, et al. "On practical discrete Gaussian samplers for lattice-based cryptography."
[TWS19]: Tian, Shanquan, et al. "Merge-exchange sort based discrete Gaussian sampler with fixed memory access pattern."

Performance

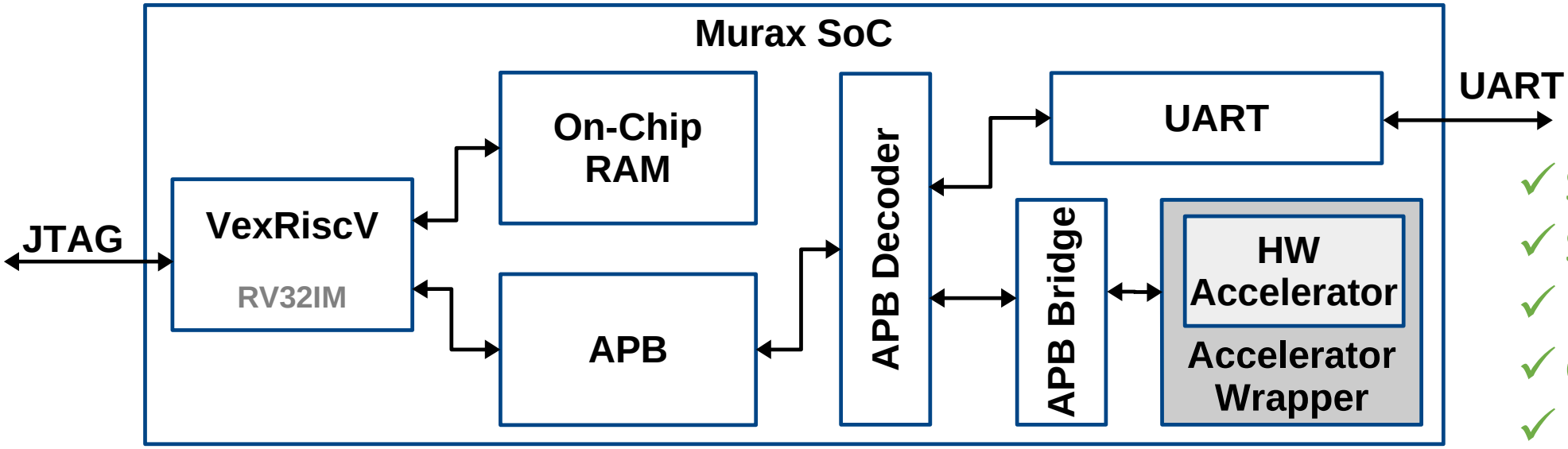
- ✓ Parameterized
- ✓ Achieves theoretical cycles limit $(n \cdot \log_2 n + \frac{n}{2})$
- ✓ Good time-area product

Design	Parameters	Tunable (n, q)	Platform	Cycles	Slices	LUTs	FFs	DSPs	Fmax (MHz)
Ours	(1024, 343576577)	√, √	Artix-7	11,455	582	1977	991	6	124
	(2048, 856145921)		Artix-7	24,785	555	1981	1021	8	124
Ours	(1024, 12289)	√, √	Artix-7	11,455	271	944	467	3	141
[KLC+17]		√, –	Artix-7	5494	–	2832	1381	8	150
Ours	(2048, 65537)	√, √	Spartan-6	24,785	543	1601	368	8	90
[DB16]		–, –	Spartan-6	25,654	269	–	–	9	207

[HKR+18]: Howe, James, et al. "On practical discrete Gaussian samplers for lattice-based cryptography."
[TWS19]: Tian, Shanquan, et al. "Merge-exchange sort based discrete Gaussian sampler with fixed memory access pattern."

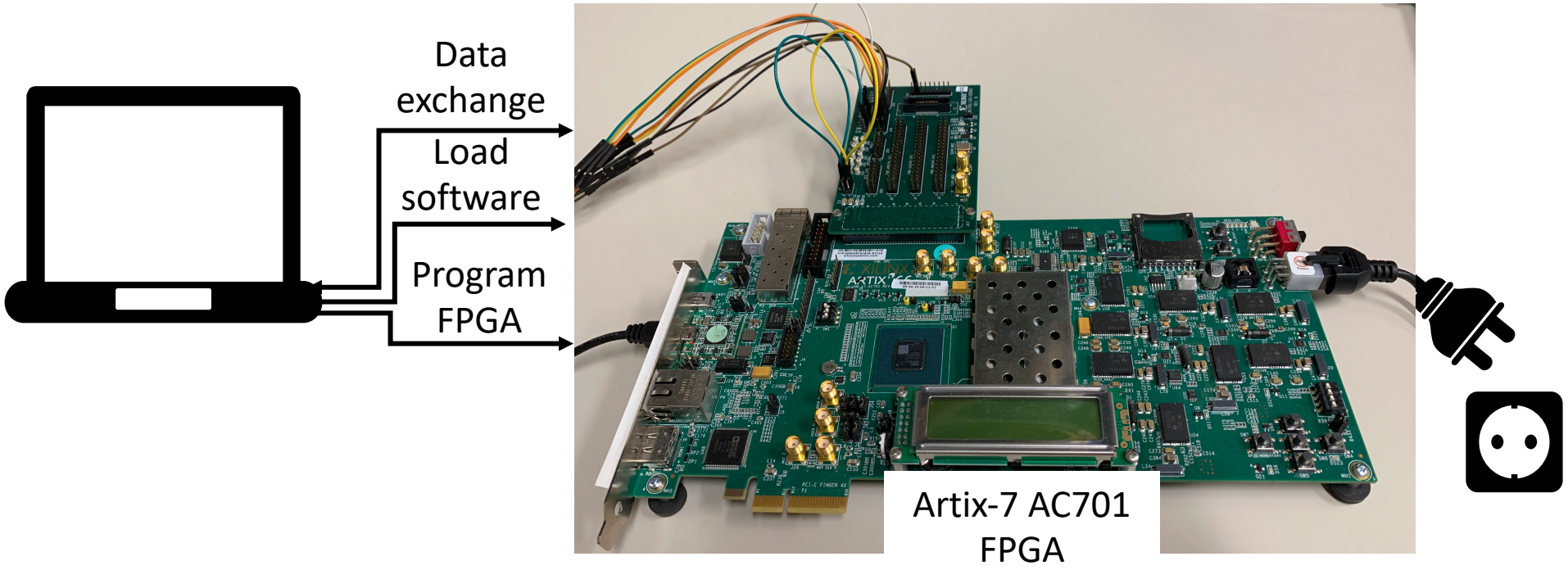
Software-hardware co-design of qTESLA on RISC-V

Software-hardware co-design on RISC-V



- ✓ Standard ISA
- ✓ Standard interface
- ✓ Lightweight
- ✓ Good performance
- ✓ Fully open-source

Experimental setup for qTESLA on FPGA



- Artix-7 FPGA board from Xilinx (recommended by NIST)

Performance of core functions in qTESLA

Function	SW Cycles	HW Cycles	HW-SW Cycles	IO Overhead(%)	Speedup SW/HW	Speedup SW/HW-SW
qTESLA-p-I						
SHAKE128	44,683	505	1586	214.1	88.5	28.2
GaussSampler	3,540,807	18,451	26,286	42.5	191.9	134.7
GaussSampler + HmaxSum	4,009,628	29,293	29,397	0.4	136.9	136.4
PolyMul	558,365	11,455	31,473	174.8	48.7	17.7
SparseMul8	365,207	7225	28,181	290.1	50.5	13.0
SparseMul32	571,165	7225	28,180	290.0	79.1	20.3
qTESLA-p-III						
SHAKE256	45,581	473	1562	230.2	96.4	29.2
GaussSampler	16,707,765	81,335	81,505	0.2	205.4	205.0
GaussSampler + HmaxSum	18,195,064	88,676	88,748	0.1	205.2	205.0
PolyMul	1,179,949	24,785	63,743	157.2	47.6	18.5
SparseMul8	1,160,213	20,561	62,216	202.6	56.4	18.6
SparseMul32	1,780,940	20,561	62,226	202.6	86.6	28.6

Performance of core functions in qTESLA

Function	SW Cycles	HW Cycles	HW-SW Cycles	IO Overhead(%)	Speedup SW/HW	Speedup SW/HW-SW
qTESLA-p-I						
SHAKE128	44,683	505	1586	214.1	88.5	28.2
GaussSampler	3,540,807	18,451	26,286	42.5	191.9	134.7
GaussSampler + HmaxSum	4,009,628	29,293	29,397	0.4	136.9	136.4
PolyMul	558,365	11,455	31,473	174.8	48.7	17.7
SparseMul8	365,207	7225	28,181	290.1	50.5	13.0
SparseMul32	571,165	7225	28,180	290.0	79.1	20.3
qTESLA-p-III						
SHAKE256	45,581	473	1562	230.2	96.4	29.2
GaussSampler	16,707,765	81,335	81,505	0.2	205.4	205.0
GaussSampler + HmaxSum	18,195,064	88,676	88,748	0.1	205.2	205.0
PolyMul	1,179,949	24,785	63,743	157.2	47.6	18.5
SparseMul8	1,160,213	20,561	62,216	202.6	56.4	18.6
SparseMul32	1,780,940	20,561	62,226	202.6	86.6	28.6

Performance of core functions in qTESLA

Function	SW Cycles	HW Cycles	HW-SW Cycles	IO Overhead(%)	Speedup SW/HW	Speedup SW/HW-SW
qTESLA-p-I						
SHAKE128	44,683	505	1586	214.1	88.5	28.2
GaussSampler	3,540,807	18,451	26,286	42.5	191.9	134.7
GaussSampler + HmaxSum	4,009,628	29,293	29,397	0.4	136.9	136.4
PolyMul	558,365	11,455	31,473	174.8	48.7	17.7
SparseMul8	365,207	7225	28,181	290.1	50.5	13.0
SparseMul32	571,165	7225	28,180	290.0	79.1	20.3
qTESLA-p-III						
SHAKE256	45,581	473	1562	230.2	96.4	29.2
GaussSampler	16,707,765	81,335	81,505	0.2	205.4	205.0
GaussSampler + HmaxSum	18,195,064	88,676	88,748	0.1	205.2	205.0
PolyMul	1,179,949	24,785	63,743	157.2	47.6	18.5
SparseMul8	1,160,213	20,561	62,216	202.6	56.4	18.6
SparseMul32	1,780,940	20,561	62,226	202.6	86.6	28.6

Performance of qTESLA: key generation

Design	Cycles	Fmax (MHz)	Time (ms)	Time×Area (slice×ms)	Speedup
qTESLA-p-I					
Murax	48,529,602	156	310.9	328,299	1.00
+ SHAKE	18,214,784	145	125.5	164,472	2.48
+ GaussSampler (incl. SHAKE)	6,653,608	137	48.7	73,436	6.38
+ GaussSampler + HmaxSum	2,525,853	126	20.1	30,792	15.47
+ PolyMul (incl. ModMul)	46,933,182	126	373.8	602,596	0.83
+ SparseMul	48,529,602	134	361.8	424,795	0.86
+ All	925,431	121	7.7	18,651	40.64
qTESLA-p-III					
Murax	297,103,198	156	1903.3	2,009,841	1.00
+ SHAKE	120,731,265	145	831.5	1,090,134	2.29
+ GaussSampler (incl. SHAKE)	28,394,350	126	224.8	350,687	8.47
+GaussSampler + HmaxSum	6,494,464	126	51.7	83,606	36.79
+ PolyMul (incl. ModMul)	292,924,220	125	2340.8	3,829,482	0.81
+ SparseMul	297,103,153	161	1849.8	2,199,459	1.03
+ All	2,305,220	121	19.0	47,001	100.14

Performance of qTESLA: key generation

Design	Cycles	Fmax (MHz)	Time (ms)	Time×Area (slice×ms)	Speedup
qTESLA-p-I					
Murax	48,529,602	156	310.9	328,299	1.00
+ SHAKE	18,214,784	145	125.5	164,472	2.48
+ GaussSampler (incl. SHAKE)	6,653,608	137	48.7	73,436	6.38
+ GaussSampler + HmaxSum	2,525,853	126	20.1	30,792	15.47
+ PolyMul (incl. ModMul)	46,933,182	126	373.8	602,596	0.83
+ SparseMul	48,529,602	134	361.8	424,795	0.86
+ All	925,431	121	7.7	18,651	40.64
qTESLA-p-III					
Murax	297,103,198	156	1903.3	2,009,841	1.00
+ SHAKE	120,731,265	145	831.5	1,090,134	2.29
+ GaussSampler (incl. SHAKE)	28,394,350	126	224.8	350,687	8.47
+GaussSampler + HmaxSum	6,494,464	126	51.7	83,606	36.79
+ PolyMul (incl. ModMul)	292,924,220	125	2340.8	3,829,482	0.81
+ SparseMul	297,103,153	161	1849.8	2,199,459	1.03
+ All	2,305,220	121	19.0	47,001	100.14

Performance of qTESLA: sign

Design	Cycles	Fmax (MHz)	Time (ms)	Time×Area (slice×ms)	Speedup
qTESLA-p-I					
Murax	47,180,534	156	302.2	319,171	1.00
+ SHAKE	22,914,215	145	157.8	206,905	1.91
+ GaussSampler (incl. SHAKE)	23,348,731	137	171.0	257,697	1.77
+ GaussSampler + HmaxSum	24,580,234	126	195.6	299,647	1.55
+ PolyMul (incl. ModMul)	34,013,026	126	270.9	436,707	1.12
+ SparseMul	41,356,497	134	308.4	362,007	0.98
+ All	4,165,160	121	34.4	83,944	8.78
qTESLA-p-III					
Murax	105,525,187	156	676.0	713,865	1.00
+ SHAKE	54,013,152	145	372.0	487,710	1.82
+ GaussSampler (incl. SHAKE)	55,308,030	126	437.9	683,084	1.54
+ GaussSampler + HmaxSum	53,024,762	126	422.4	682,611	1.60
+ PolyMul (incl. ModMul)	78,377,348	125	626.3	1,024,655	1.08
+ SparseMul	86,540,888	161	538.8	640,664	1.25
+ All	7,745,088	121	63.9	157,916	10.59

Performance of qTESLA: verify

Design	Cycles	Fmax (MHz)	Time (ms)	Time×Area (slice×ms)	Speedup
qTESLA-p-I					
Murax	17,871,157	156	114.5	120,895	1.00
+ SHAKE	4,625,094	145	31.9	41,763	3.59
+ GaussSampler (incl. SHAKE)	4,625,505	137	33.9	51,052	3.38
+ GaussSampler + HmaxSum	4,623,861	126	36.8	56,368	3.11
+ PolyMul (incl. ModMul)	16,274,763	126	129.6	208,960	0.88
+ SparseMul	15,793,283	134	117.8	138,243	0.97
+ All	946,520	121	7.8	19,076	14.63
qTESLA-p-III					
Murax	48,309,625	156	309.5	326,810	1.00
+ SHAKE	14,899,621	145	102.6	134,535	3.02
+ GaussSampler (incl. SHAKE)	14,892,149	126	117.9	183,927	2.63
+ GaussSampler + HmaxSum	14,889,776	126	118.6	191,684	2.61
+ PolyMul (incl. ModMul)	44,130,687	125	352.6	576,934	0.88
+ SparseMul	39,915,065	161	248.5	295,490	1.25
+ All	2,315,950	121	19.1	47,220	16.21

Comparison with related work

Design	Platform	Freq. (MHz)	KeyGen./Sign/Verify $\times 10^3$ Cycles	KeyGen./Sign/Verify Time (ms)
NIST Security Level = 1				
qTESLA-p-I, our	Murax+HW^p	121	925/4165/947	7.7/34.4/7.8
qTESLA-p-I, —	Cortex-M4	—	—	—
Dilithium-II [KRS+19]	Cortex-M4	168	1400/6157/1461	8.3/36.6/8.7
NIST Security Level = 3				
qTESLA-p-III, our	Murax+HW^p	121	2305/7745/2316	19.0/63.9/19.1
qTESLA-p-III, —	Cortex-M4	—	—	—
Dilithium-III [KRS+19]	Cortex-M4	168	2282/9289/2229	13.6/55.3/13.3
Falcon-512 [KRS+19]	Cortex-M4	168	197,794/38,090/474	1177.3/226.7/2.8
Designs not matching latest NIST Security Levels				
qTESLA-I^o, our	Murax+HW^p	125	181/781/137	1.4/6.2/1.1
qTESLA-I^o [BUC19]	RISC-V+HW^p	10	4847/168/39	484.7/16.8/3.9
qTESLA-I^o [KRS+19]	Cortex-M4	168	6748/5831/788	40.2/34.7/4.7

[KRS+19]: Kannwischer, Matthias, et al. "pqm4: testing and benchmarking NIST PQC on ARM Cortex-M4.."
[BUC19]: Banerjee, Utsav, et al. "A configurable crypto-processor for post-quantum lattice-based protocols ."

Comparison with related work

Design	Platform	Freq. (MHz)	KeyGen./Sign/Verify $\times 10^3$ Cycles	KeyGen./Sign/Verify Time (ms)
NIST Security Level = 1				
qTESLA-p-I, our	Murax+HW^p	121	925/4165/947	7.7/34.4/7.8
qTESLA-p-I, —	Cortex-M4	—	—	—
Dilithium-II [KRS+19]	Cortex-M4	168	1400/6157/1461	8.3/36.6/8.7
NIST Security Level = 3				
qTESLA-p-III, our	Murax+HW^p	121	2305/7745/2316	19.0/63.9/19.1
qTESLA-p-III, —	Cortex-M4	—	—	—
Dilithium-III [KRS+19]	Cortex-M4	168	2282/9289/2229	13.6/55.3/13.3
Falcon-512 [KRS+19]	Cortex-M4	168	197,794/38,090/474	1177.3/226.7/2.8
Designs not matching latest NIST Security Levels				
qTESLA-I^o, our	Murax+HW^p	125	181/781/137	1.4/6.2/1.1
qTESLA-I^o [BUC19]	RISC-V+HW^p	10	4847/168/39	484.7/16.8/3.9
qTESLA-I^o [KRS+19]	Cortex-M4	168	6748/5831/788	40.2/34.7/4.7

[KRS+19]: Kannwischer, Matthias, et al. "pqm4: testing and benchmarking NIST PQC on ARM Cortex-M4.."
[BUC19]: Banerjee, Utsav, et al. "A configurable crypto-processor for post-quantum lattice-based protocols ."

Comparison with related work

Design	Platform	Freq. (MHz)	KeyGen./Sign/Verify $\times 10^3$ Cycles	KeyGen./Sign/Verify Time (ms)
NIST Security Level = 1				
qTESLA-p-I, our	Murax+HW^p	121	925/4165/947	7.7/34.4/7.8
qTESLA-p-I, —	Cortex-M4	—	—	—
Dilithium-II [KRS+19]	Cortex-M4	168	1400/6157/1461	8.3/36.6/8.7
NIST Security Level = 3				
qTESLA-p-III, our	Murax+HW^p	121	2305/7745/2316	19.0/63.9/19.1
qTESLA-p-III, —	Cortex-M4	—	—	—
Dilithium-III [KRS+19]	Cortex-M4	168	2282/9289/2229	13.6/55.3/13.3
Falcon-512 [KRS+19]	Cortex-M4	168	197,794/38,090/474	1177.3/226.7/2.8
Designs not matching latest NIST Security Levels				
qTESLA-I^o, our	Murax+HW^p	125	181/781/137	1.4/6.2/1.1
qTESLA-I^o [BUC19]	RISC-V+HW^p	10	4847/168/39	484.7/16.8/3.9
qTESLA-I^o [KRS+19]	Cortex-M4	168	6748/5831/788	40.2/34.7/4.7

[KRS+19]: Kannwischer, Matthias, et al. "pqm4: testing and benchmarking NIST PQC on ARM Cortex-M4.."
[BUC19]: Banerjee, Utsav, et al. "A configurable crypto-processor for post-quantum lattice-based protocols ."

Summary

- Design and implement various hardware accelerators for lattice-based schemes:
  - SHAKE-128/256 and cSHAKE-128/256
  - Binary-search CDT sampler
  - NTT-based polynomial multiplier
  - Sparse polynomial multiplier
  - Hmax-Sum module
- Prototype of the full qTESLA scheme as a software-hardware co-design on a RISC-V platform.
- Fully open-source code: https://caslab.csl.yale.edu/code/qtesla-hw-sw-platform/

Thank you!