Open-Source FPGA Implementation of Post-Quantum Cryptographic Hardware Primitives

Rashmi Agrawal, Bu Lake, Alan Ehret, and Michel Kinsy
Adaptive & Secure Computing Systems Lab
Department of Electrical & Computer Engineering
Boston University

Presentation Outline

- Motivation: why quantum-proof?
- NIST: steps towards standardization
- State of the Art: main algorithm
- FPGA-based Implementation: primitives
- Evaluation: cost and performance
- Key Contributions: conclusion

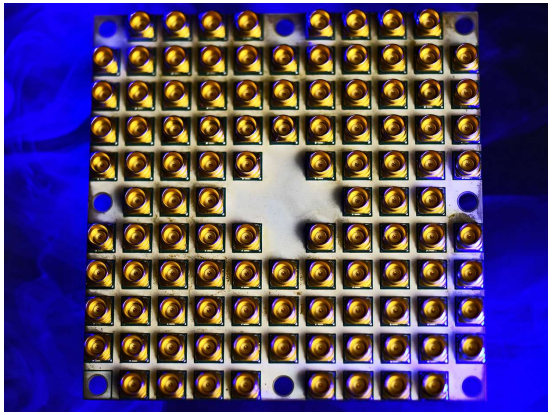
Presentation Outline

- Motivation: why quantum-proof?
- NIST: steps towards standardization
- State of the Art: main algorithm
- FPGA-based Implementation: primitives
- Evaluation: cost and performance
- Key Contributions: conclusion

Ongoing Development

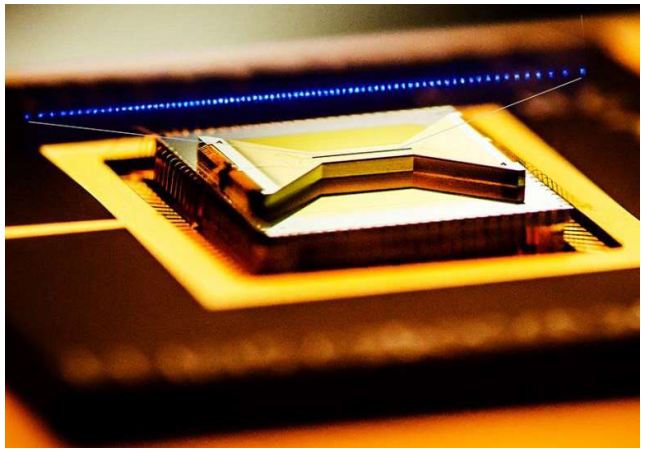
IBM 50Q System: An IBM cryostat wired for a 50 qubit system.

IBM's Q System 50 Qubits, 20 Qubits



Intel's Tangle lake 49 Qubits

Google's Bristlecone – 72 Qubits



IonQ 160 Qubits

With Quantum Supremacy…

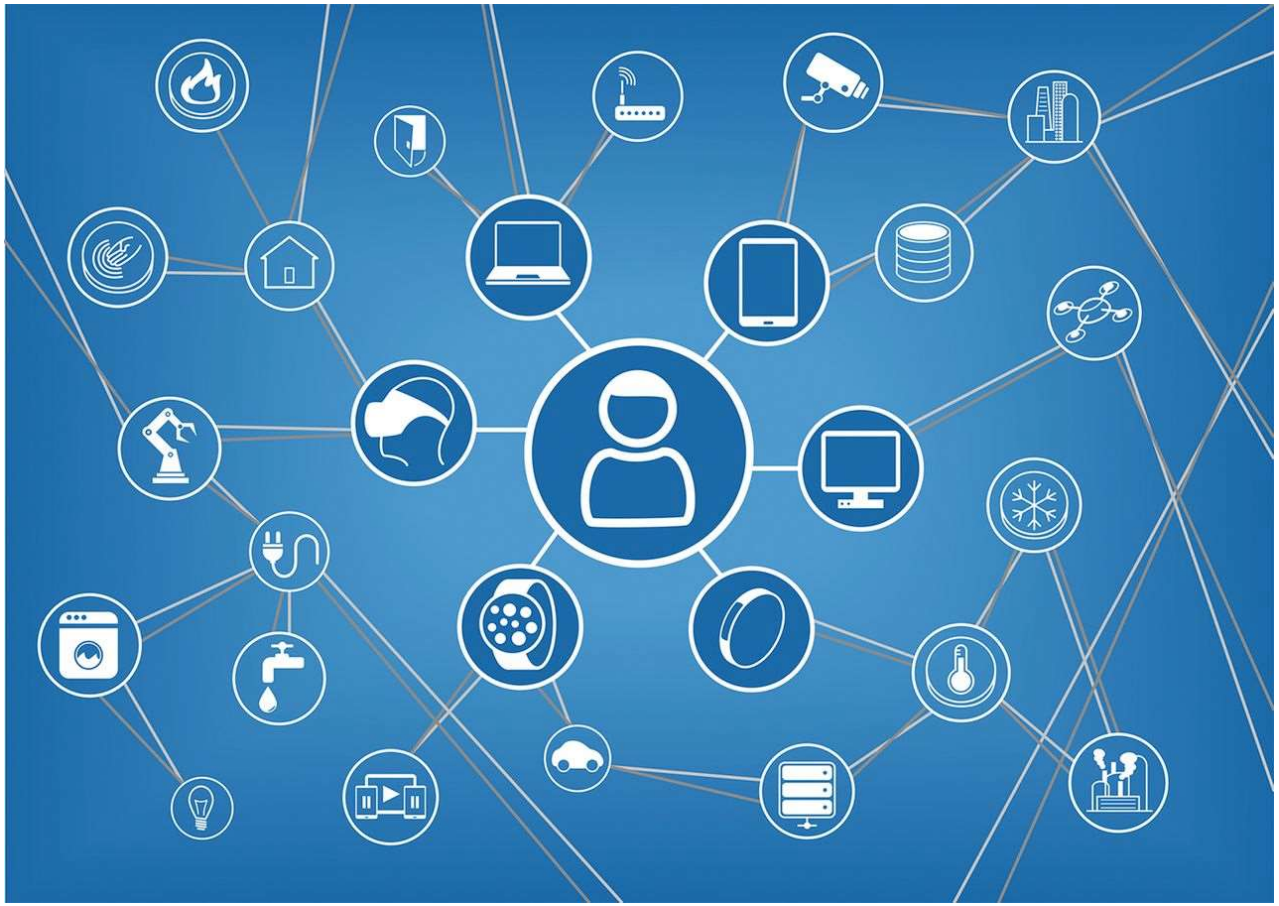
- What is *NOT* considered as post-quantum secure?

Algorithm	Secure in Post-quantum Era?
RSA-1024, -2048, -4096	No
Elliptic Curve Crypto (ECC)-256, -521	No
Diffie-Hellman	No
ECC Diffie-Hellman	No
AES-128, -192	No

[1]

[1] https://www.nist.gov/

How does this impacts us?



Question

- Can we <u>increase the key size</u> of some popular encryption schemes, so that they can be post-quantum secure?
  - Maybe yes, maybe no

Table II. Equivalent Security Levels of AES and RSA under Attacks from Classic and Quantum Computers *

Attack Platform	Symmetric Encryption			Asymmetric (Public-key) Encryption		
	Algorithm	Key Size	Security Level	Algorithm	Key Size	Security Level
Classic Computers	AES-128	128	128	RSA-2048	2,048	112
	AES-256	256	256	RSA-15360	15,360	256
Quantum Computers	AES-128	128	64	RSA-2048	2,048	25
	AES-256	256	128	RSA-15360	15,360	31

Grover's algorithm

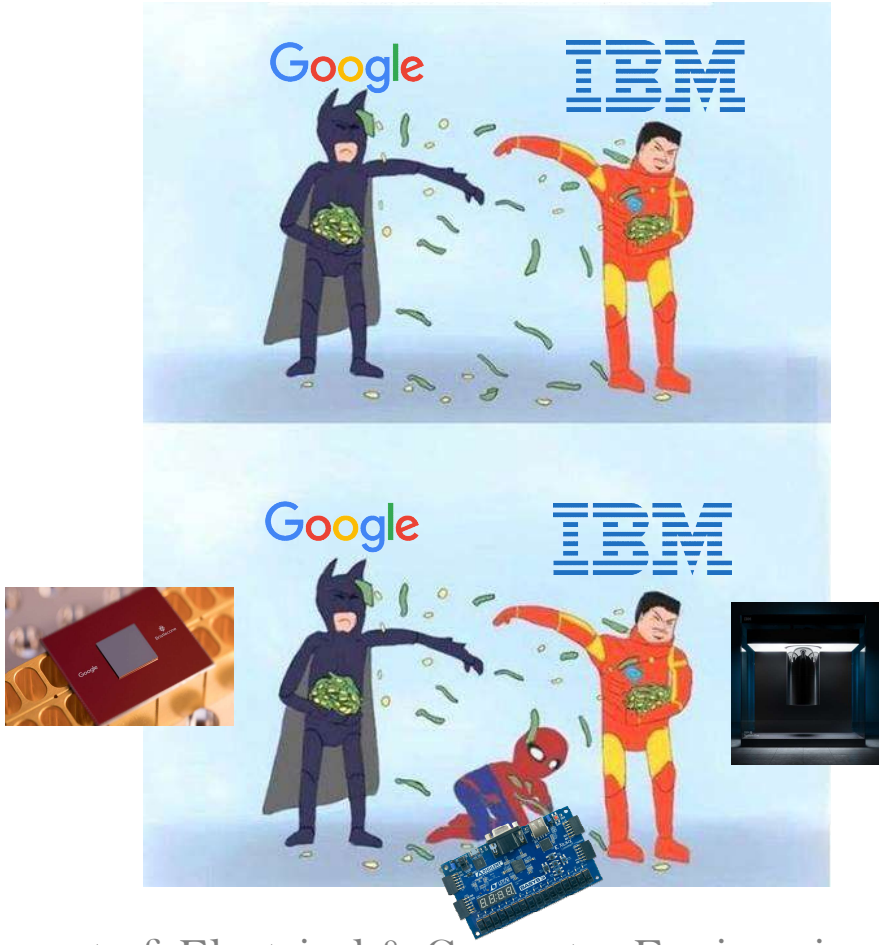
Shor's algorithm

* TechBeacon, Waiting for quantum computing: Why encryption has nothing to worry about, 2018

Quantum Computer-based Cryptography
vs
General Computer-based Quantum-proof Cryptography

Batman & Ironman
Vs
Spiderman

Quantum Computer-based Cryptography
vs
General Computer-based Quantum-proof Cryptography



Presentation Outline

- Motivation: why quantum-proof?
- NIST: steps towards standardization
- State of the Art: main algorithm
- FPGA-based Implementation: primitives
- Evaluation: cost and performance
- Key Contributions: conclusion

Post-Quantum Cryptography (PQC) Standardization (Round -1)

- NIST
  - Jan 2017 – Dec 2018
  - Evaluating 69 (5 withdrawn) submissions of PQC, to bring up a standard (just like AES or RSA):
    - 21 lattice-based
    - 18 code-based
    - Some hash-based
    - Some others

[1]

Algorithm	Algorithm Information *KAT files are included in zip file unless they were too large*	Submitters	Comments
BIG QUAKE	Zip File (4MB) IP Statements Website	Alain Couvreur Magali Bardet Elise Barelli Olivier Blazy Rodolfo Canto-Torres Philippe Gaborit Ayoub Otmani Nicolas Sendrier Jean-Pierre Tillich	Submit Comment View Comments
BIKE	Zip File (10MB) IP Statements Website	Nicolas Aragon Paulo Barreto Slim Bettaieb Loic Bidoux	Submit Comment View Comments
CFPKM	Zip File (<1MB) IP Statements Website	O. Chakraborty J.-C. Faugere L. Perret	Submit Comment View Comments
Classic McEliece	Zip File (<1MB) KAT Files (26MB) IP Statements Website	Daniel J. Bernstein Tung Chou Tanja Lange Ingo von Maurich Rafael Misoczki Ruben Niederhagen Edoardo Persichetti Christiane Peters Peter Schwabe Nicolas Sendrier Jakub Szefer Wen Wang	Submit Comment View Comments
Compact LWE	Zip File (1MB) IP Statements Website	Dongxi Liu Nan Li Jongkil Kim Surya Nepal	Submit Comment View Comments

Submission deadline Nov 30, 2017. List updated Dec 20, 2018.

[1] https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions

Post-Quantum Cryptography (PQC) Standardization (Round -1)

- NIST
  - Jan 2017 – Dec 2018
  - Evaluating 69 (5 withdrawn) submissions of PQC, to bring up a standard (just like AES or RSA):
    - 21 lattice-based
    - 18 code-based
    - Some hash-based
    - Some others

[1]

Ring-Learning with Error (Ring-LWE)

Submission deadline Nov 30, 2017. List updated Dec 20, 2018.

[1] https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions

Post-Quantum Cryptography (PQC) Standardization (Round -2)

- NIST
  - Jan 30, 2019 published candidates of Round-2:
  - 26 candidates
  - Who survived?
    - 12 lattice-based
    - 8 code-based
    - some multivariate-based and hash based for digital signatures

PQC Standardization Process: Second Round Candidate Announcement

January 30, 2019

After over a year of evaluation, NIST would like to announce the candidates that will be moving on to the 2nd round of the NIST PQC Standardization Process.

The 17 Second-Round Candidate public-key encryption and key-establishment algorithms are:

- BIKE
- Classic McEliece
- CRYSTALS-KYBER
- FrodoKEM
- HQC
- LAC
- LEDAcrypt (merger of LEDAkem/LEDApkc)
- NewHope
- NTRU (merger of NTRUEncrypt/NTRU-HRSS-KEM)
- NTRU Prime
- NTS-KEM
- ROLLO (merger of LAKE/LOCKER/Ouroboros-R)
- Round5 (merger of Hila5/Round2)
- RQC
- SABER
- SIKE
- Three Bears

The 9 Second Round Candidates for digital signatures are:

- CRYSTALS-DILITHIUM
- FALCON
- GeMSS
- LUOV
- MQDSS

PARENT PROJECT

See: Post-Quantum Cryptography

TOPICS

Security and Privacy: post-quantum cryptography.

RELATED PAGES

News Item: NIST Publishes NISTIR 8240: PQC Round 1 Status Report

Post-Quantum Cryptography (PQC) Standardization (Round -2)

Sr. No.	Public-Key Encryption	
	Lattice-based/R-LWE	Code-based
1	NTRU Prime (R-lattice)	Classic McEliece (Binary Goppa)
2	NTRU (R-lattice)	HQC (BCH & Cyclic)
3	LAC (R-LWE)	RQC (Cyclic)
4	SABER (Mod-LWR)	LEDA (LDPC)
5	Round5 (R-LWR)	ROLLO (LAKE & LOCKER) (LRPC)

Post-Quantum Cryptography (PQC) Standardization (Round -2)

Sr. No.	Key Establishment/Encapsulation	
	Lattice-based/R-LWE	Code-based
1	NewHope (R-LWE)	BIKE (MDPC)
2	NTRU (R-lattice)	NTS-KEM (Binary Goppa)
3	FrodoKEM (R-LWE)	LEDA (LDPC)
4	CRYSTALS (R-LWE)	ROLLO (LRPC) (LAKE & LOCKER)
5	SABER (Mod-LWR)	
6	Three Bears (Mod-LWR)	

Post-Quantum Cryptography (PQC) Standardization (Round -2)

Sr. No.	Digital Signature		
	Lattice-based/R-LWE	Multivariate-based	Others
1	FALCON (NTRU R-lattice)	GeMSS	Picnic
2	qTESLA (R-LWE)	MQDSS	SPHINCS
3	CRYSTALS (R-LWE)	LUOV	
4		Rainbow	

Why Ring-LWE?

- Advantages
  1) Based on LWE - a branch of lattice-based cryptosystem

Learning with Error (LWE)

$$\underset{a}{\begin{bmatrix} 2 & 13 & 7 & 3 \\ 4 & 7 & 9 & 1 \\ 6 & 14 & 5 & 11 \\ 5 & 11 & 13 & 2 \end{bmatrix}} * \underset{s}{\begin{bmatrix} s1 \\ s2 \\ s3 \\ s4 \end{bmatrix}} + \underset{e}{\begin{bmatrix} e1 \\ e2 \\ e3 \\ e3 \end{bmatrix}} = \underset{b}{\begin{bmatrix} 13 \\ 12 \\ 3 \\ 9 \end{bmatrix}}$$

- An arbitrary number of equations, each distorted up to $\pm\alpha q$,
- How to find s?

$$(2s1 + 13s2 + 7s3 + 3s4) + e1 \approx 13 \pmod q$$
$$(4s1 + 7s2 + 9s3 + 1s4) + e2 \approx 12 \pmod q$$
$$(6s1 + 14s2 + 5s3 + 11s4) + e3 \approx 3 \pmod q$$
$$(5s1 + 11s2 + 13s3 + 2s4) + e4 \approx 9 \pmod q$$

Why Ring-LWE?

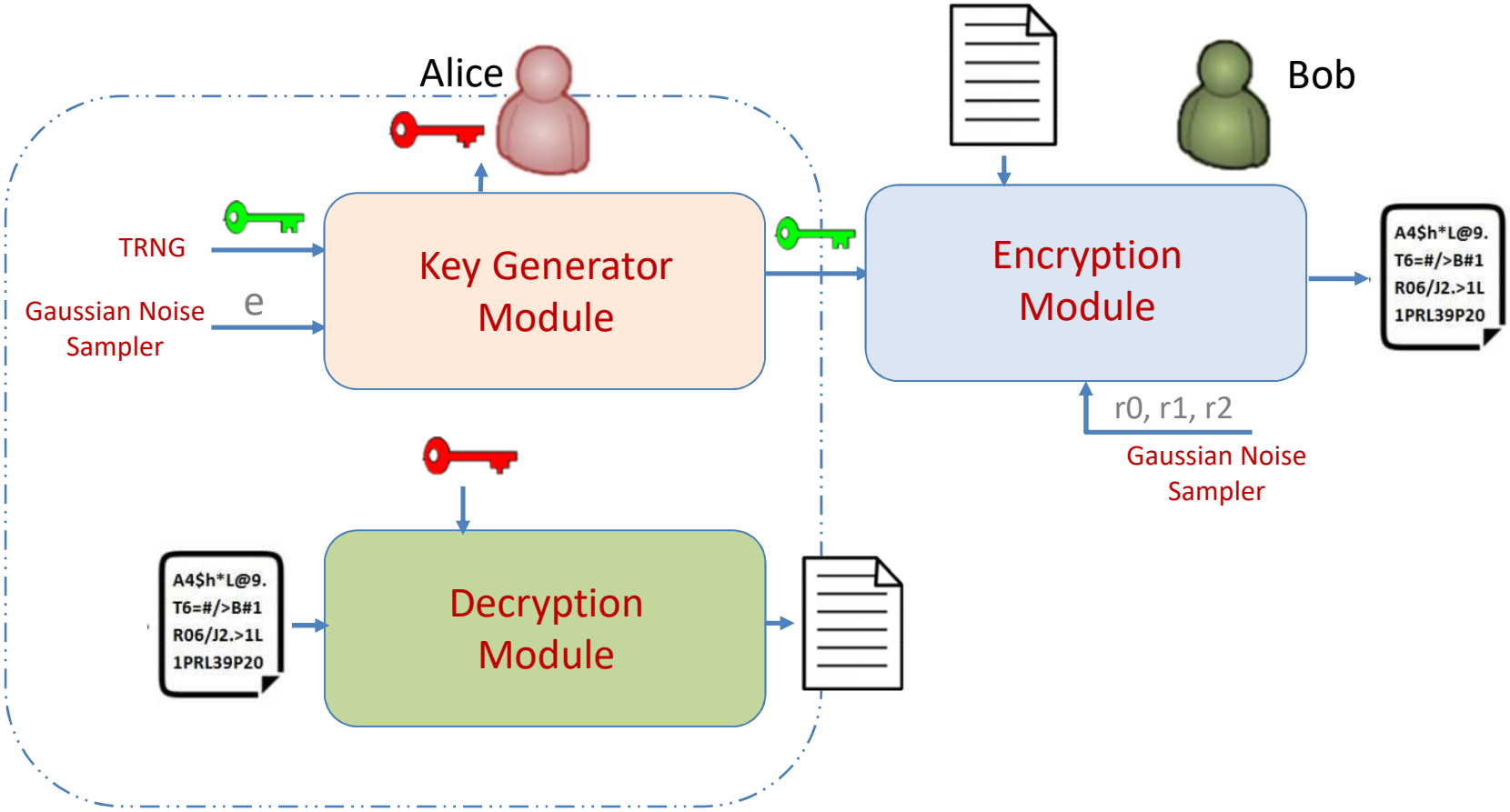
- Advantages
  1) Based on LWE - a branch of lattice-based cryptosystem
  2) Can perform
     - Public-key encryption
     - Key-exchange mechanism
     - Digital signature
  3) Can extend to somewhat homomorphic encryption (SHE)
  4) Smaller key size (7k~15k bits vs. 1MB for code-based & 1TB for "post-quantum RSA")
  5) Simpler computation & circuits

Presentation Outline

- Motivation: why quantum-proof?
- NIST: steps towards standardization
- State of the Art: main algorithm
- FPGA-based Implementation: primitives
- Evaluation: cost and performance
- Key Contributions: conclusion

Ring-Learning with Error (R-LWE)

- Public-Key Cryptosystem



Ring-Learning with Error (Ring-LWE)

- Public-key Cryptosystem (PKC)[1]
  - Setup (Alice)
    - Let q be a prime. In a ring R_q, picks a, s, e, where s, e are small polynomials
    - s.t. polynomial $b = a \cdot s + e$ (1)
    - Publishes $\{a, b\}$ as the public key, as well as $t = \left\lfloor \frac{q}{2} \right\rfloor$
    - Keeps s as the private key

[1] Oded Regev, "On lattices, learning with errors, random linear codes, and cryptography", 2005

Ring-Learning with Error (Ring-LWE)

- Public-key Cryptosystem (PKC)[1]
  - Setup (Alice)
    - Publishes {a, b = a·s+e} as the public key, as well as t = $\left\lfloor \frac{q}{2} \right\rfloor$.
    - Keeps s as the private key
  - Encryption (Bob to Alice):
    - Has a plaintext m (a binary string in Rq)
    - Picks small r0, r1, r2
    - Encryption using public key:
      - c0 = b · r0 + r2 + tm;
      - c1 = a · r0 + r1

[1] Oded Regev, "On lattices, learning with errors, random linear codes, and cryptography", 2005

Ring-Learning with Error (Ring-LWE)

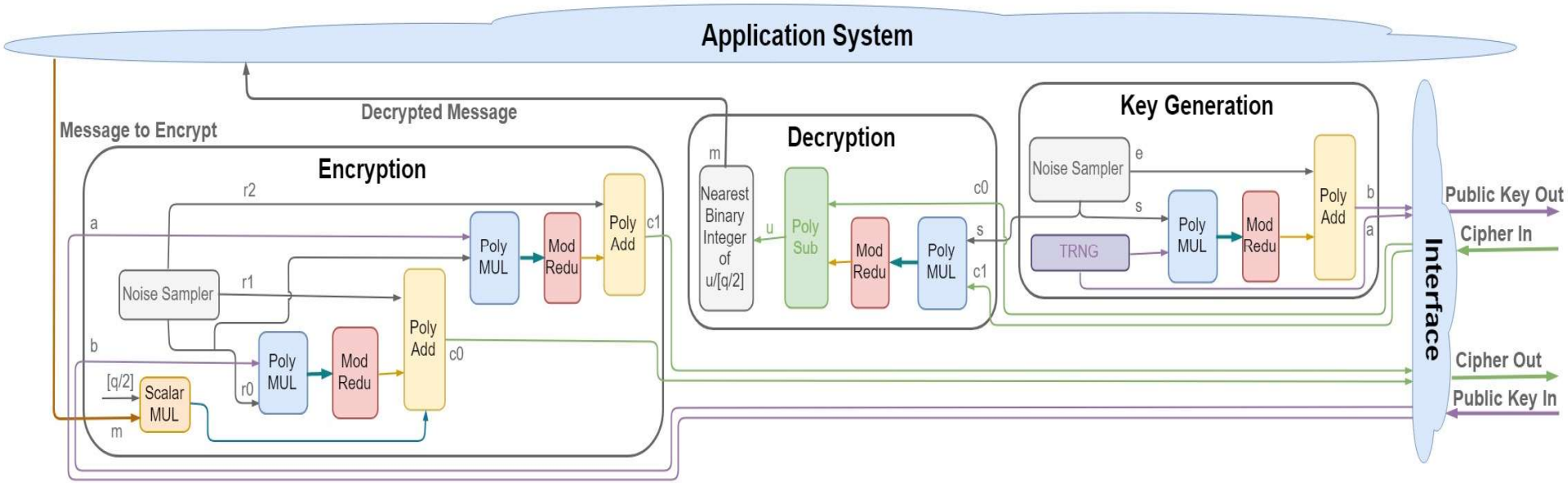
- Public-key Cryptosystem (PKC)[1]
  - Setup (Alice)
    - Publishes $\{a, b = a \cdot s + e\}$ as the public key, as well as $t = \left\lfloor \frac{q}{2} \right\rfloor$
    - Keeps s as the private key
  - Encryption (Bob to Alice):
    - Generates the cipher:
      - $c0 = b \cdot r0 + r2 + tm$;
      - $c1 = a \cdot r0 + r1$
  - Decryption (Alice computes):
    - $c0 - s \cdot c1 = b \cdot r0 + r2 + tm - s \cdot a \cdot r0 - s \cdot r1$ (2)
      $= tm + e \cdot r0 + r2 - s \cdot r1 = tm +$ "small"
    - $m = \lceil (c0 - s \cdot c1)/t \rfloor$

e, r0, r1, r2 will be eliminated easily by Alice, but they make attacker's life so much harder.

[1] Oded Regev, "On lattices, learning with errors, random linear codes, and cryptography", 2005

R-LWE Public Key Encryption Co-processor

- Public-key Cryptosystem (PKC)



R-LWE Public Key Encryption Co-processor

- Basic Operations

  (Every operation is modular)
  - Random Number Generator
  - Gaussian Noise Sampler
  - Polynomial Addition/Subtraction
  - Scalar Multiplication with a Binary Polynomial
  - Scalar Division to the Nearest Binary Integer
  - Polynomial Multiplication

- Size of the Polynomials/Vectors
  - Length: 256, 512, or 1024
  - Coefficients: within the prime number 1,049,089

R-LWE Public Key Encryption Co-processor

- Basic Operations

  (Every operation is modular)

  - Random Number Generator ✓
  - Gaussian Noise Sampler ✓
  - Polynomial Addition/Subtraction ✓
  - Scalar Multiplication with a Binary Polynomial ✓
  - Scalar Division to the Nearest Binary Integer ✓
    - **Can be done by 2 subtractions**
  - Polynomial Multiplication **hard**

- Size of the Polynomials/Vectors
  - Length: 256, 512, or 1024
  - Symbol: within the prime number 1,049,089

Presentation Outline

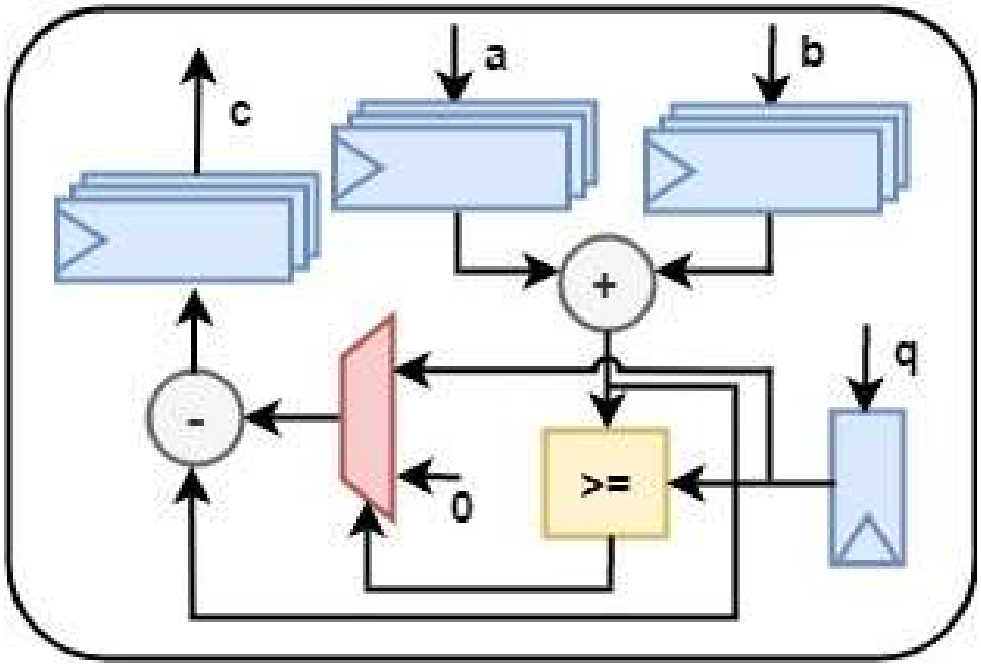
- Motivation: why quantum-proof?
- NIST: steps towards standardization
- State of the Art: main algorithm
- FPGA-based Implementation: primitives
- Evaluation: cost and performance
- Key Contributions: conclusion

Key Design Features

- Parameterized
  - Fully configurable parameters
    - Enable deployment in small devices like IoT as well as large platforms like Homomorphic Encryption
- Optimized
  - Fully optimized for reconfigurable hardware implementation
- Provides building blocks for other schemes
  - With little modifications to implement R-LWE schemes in NIST standardization process

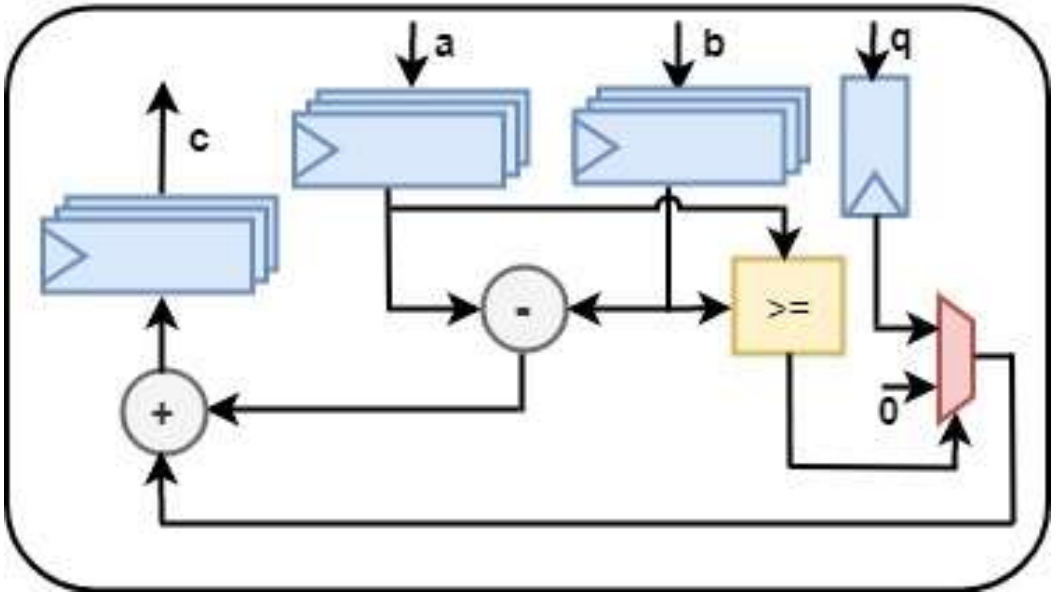
R-LWE Public Key Encryption Co-processor

- Polynomial Addition
  - If a = [a0, a1], b = [b0, b1], then:
    - c = a + b = [(a0+b0)%q, (a1+b1)%q]



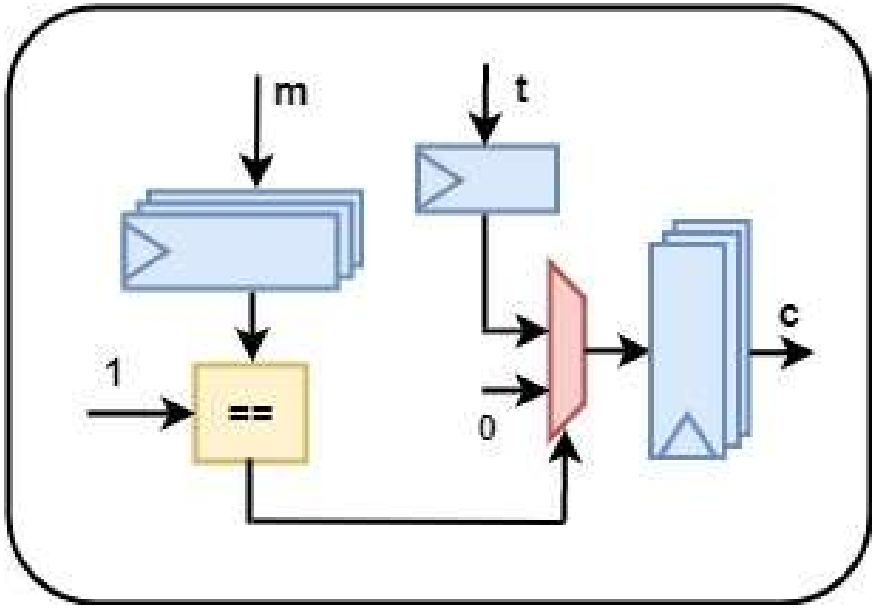
R-LWE Public Key Encryption Co-processor

- Polynomial Subtraction
  - If a = [a0, a1], b = [b0, b1], then:
    - c = a – b
      - c0 = (a0 – b0)%q
      - c0 = (a0 >= b0) ? (a0 – b0) : (q – (b0 – a0))



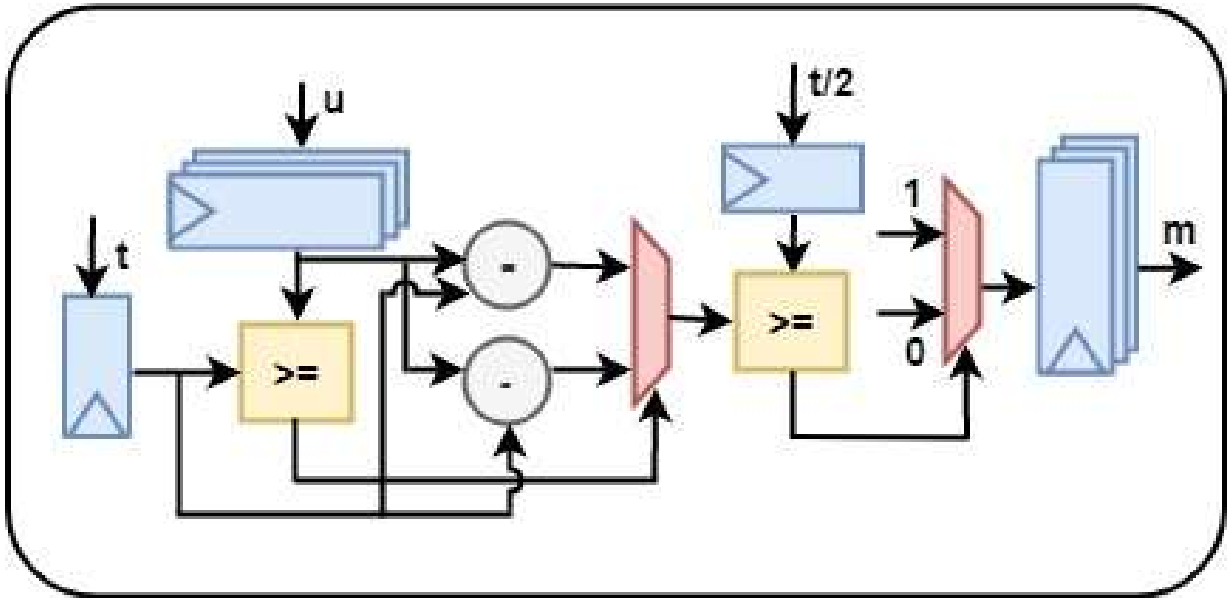
R-LWE Public Key Encryption Co-processor

- Scalar Multiplication
  - t *is* a constant and pre-computed, and
  - m the plaintext is a binary vector
    - c0 = (m[0] == 1) ? t : 0



R-LWE Public Key Encryption Co-processor

- Scalar division to the nearest binary integer
  - Denote $\mathrm{u} = (c_0 - s \cdot c_1)$
  - Compute $m = \lceil \mathrm{u}/t \rfloor$



R-LWE Public Key Encryption Co-processor

- Modular Polynomial Multiplication
  - Naïve Convolution then Polynomial Reduction
  - By FFT over finite field

Algorithm Polynomial multiplication using FFT

Let ω be a primitive n-th root of unity in $\mathbb{Z}_p$ and $\phi^2 \equiv \omega \bmod p$. Let $\mathbf{a} = (a_0, \ldots, a_{n-1})$, $\mathbf{b} = (b_0, \ldots, b_{n-1})$ and $\mathbf{c} = (c_0, \ldots, c_{n-1})$ be the coefficient vectors of degree n polynomials $a(x)$, $b(x)$, and $c(x)$, respectively, where $a_i, b_i, c_i \in \mathbb{Z}_p, i = 0, 1, \ldots, n-1$.

Input: $\mathbf{a}, \mathbf{b}, \omega, \omega^{-1}, \phi, \phi^{-1}, n, n^{-1}, p$.
Output: $\mathbf{c}$ where $c(x) = a(x) \cdot b(x) \bmod \langle x^n + 1 \rangle$.

```
Precompute: ω^i, ω^{-i}, φ^i, φ^{-i} where i = 0, 1, ..., n-1
for i = 0 to n-1 do
    ā_i ← a_i φ^i  mod p
    b̄_i ← b_i φ^i  mod p
end for
Ā ← FFT^n_ω(ā)
B̄ ← FFT^n_ω(b̄)
for i = 0 to n-1 do
    C̄_i ← Ā_i B̄_i  mod p
end for
c̄ ← IFFT^n_ω(C̄)
for i = 0 to n-1 do
    c_i ← c̄_i φ^{-i}  mod p
end for
return c
```

Negative Wrapped Convolution (NWC) → lines 3–4

Fast Number Theoretic Transform (NTT) → lines 6–7

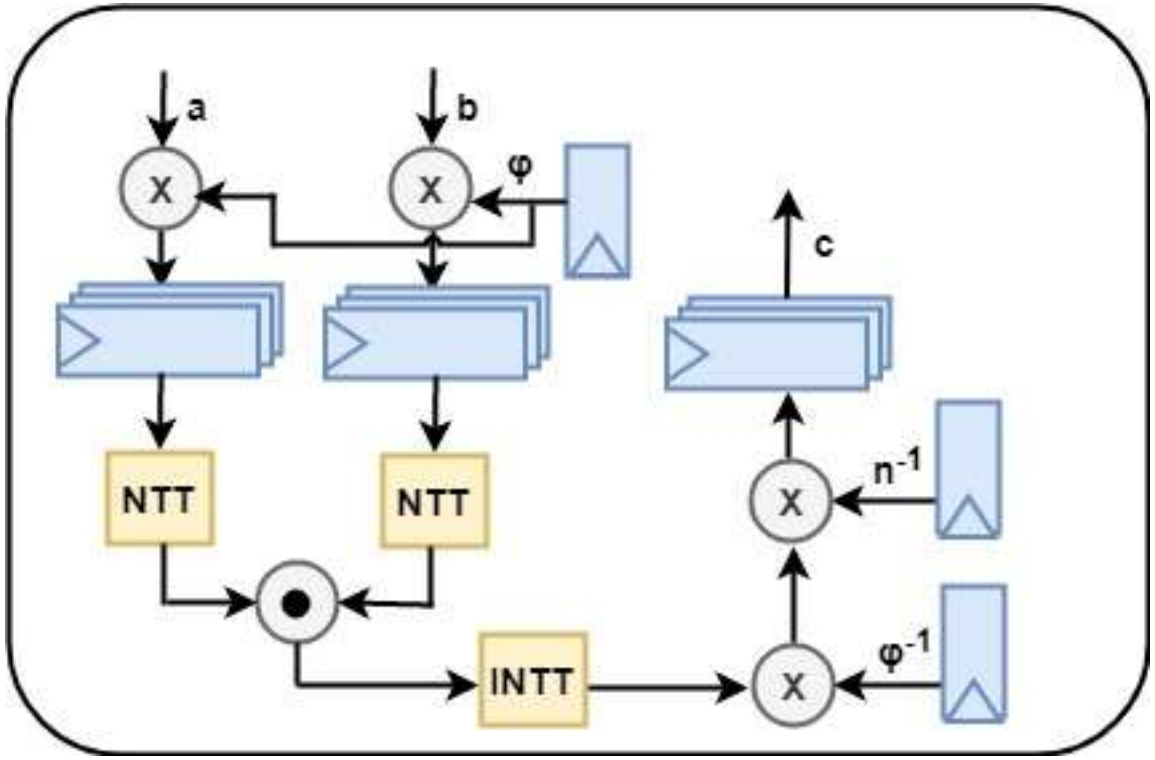
Component-wise multiplication → line 9

Inverse NTT → line 11

Inverse NWC → line 13

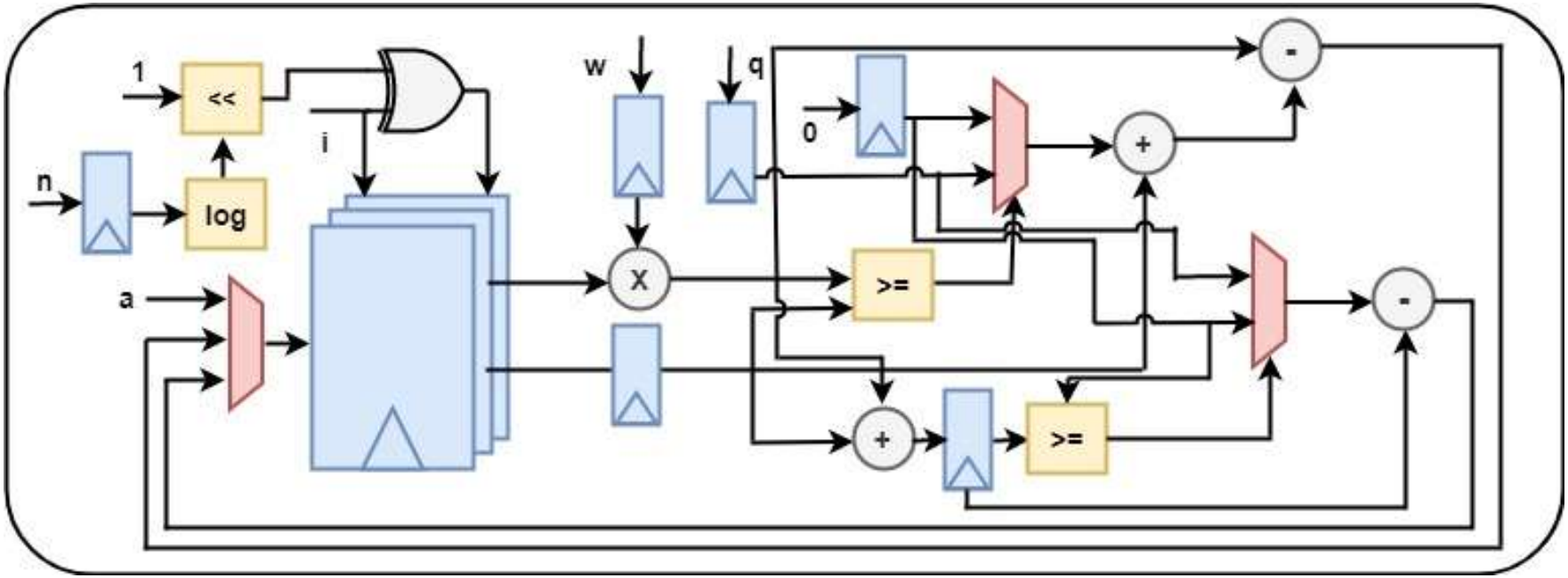
R-LWE Public Key Encryption Co-processor

- Modular Polynomial Multiplication



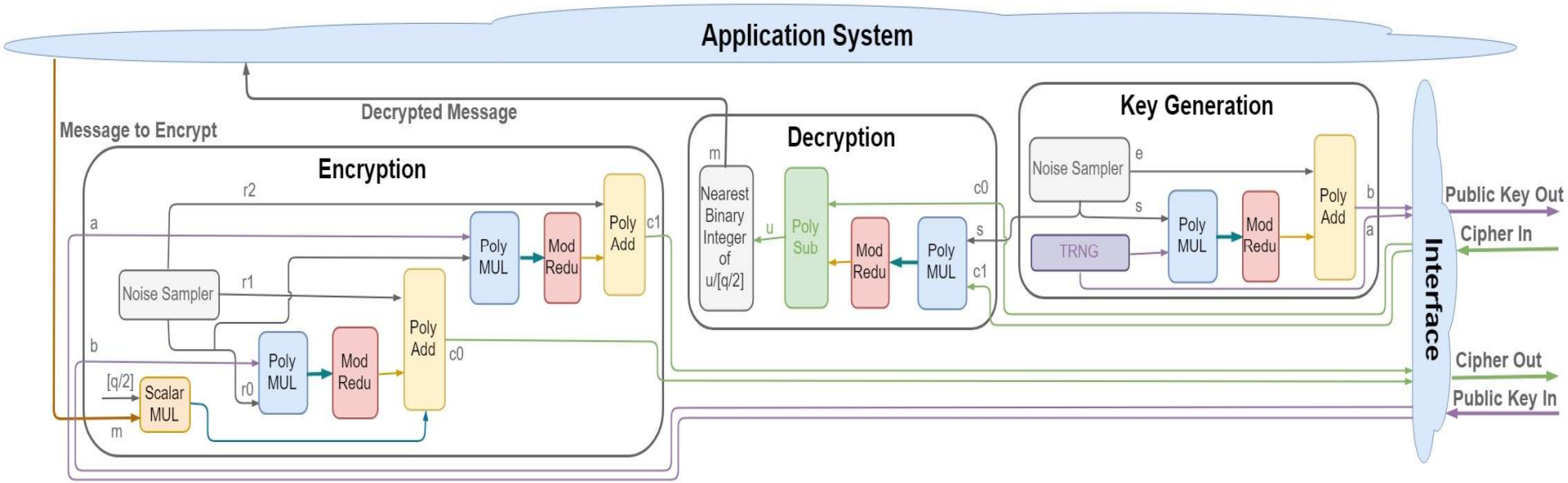
R-LWE Public Key Encryption Co-processor

- Modular Polynomial Multiplication
  - NTT Module



R-LWE Public Key Encryption Co-processor

- Public-key Cryptosystem (PKC)



Presentation Outline

- Motivation: why quantum-proof?
- NIST: steps towards standardization
- State of the Art: main algorithm
- FPGA-based Implementation: primitives
- Evaluation: cost and performance
- Key Contributions: conclusion

Performance Evaluation

- Target Platform
  - Xilinx Zynq-7000 FPGA
- Hardware Description Language
  - Verilog 2001
- Design Tool
  - Xilinx Vivado 2018.2 design suite

Correlation Between {q, n} and {Latency, Area}

Operation	Latency
KeyGen	$3n + \frac{3n}{2} log_2 n$
Enc	$7n + 2n\, log_2 n$
Dec	$4n + n\, log_2 n$
Resource	**Cost**
LUTs	$O(n\, log_2 n\, log_2 q)$
Registers	$O(n\, log_2 n\, log_2 q)$

Hardware Cost for PKC with q = 12,289

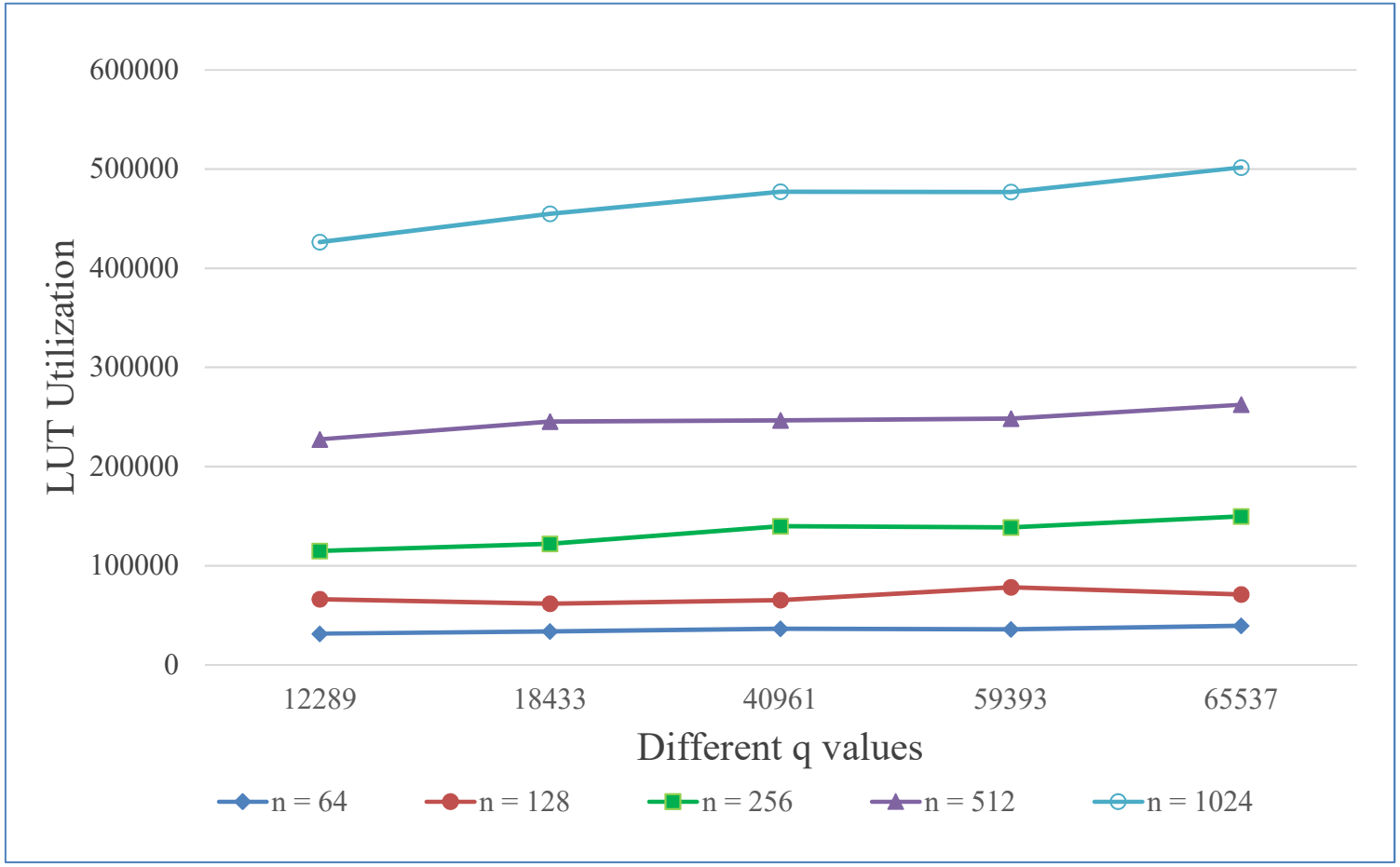
- LUTs Only Implementation

Length (n)	LUTs	Registers	DSP
128	66251	16805	26
256	114900	33138	26
512	227458	65643	26
1024	426402	130540	26

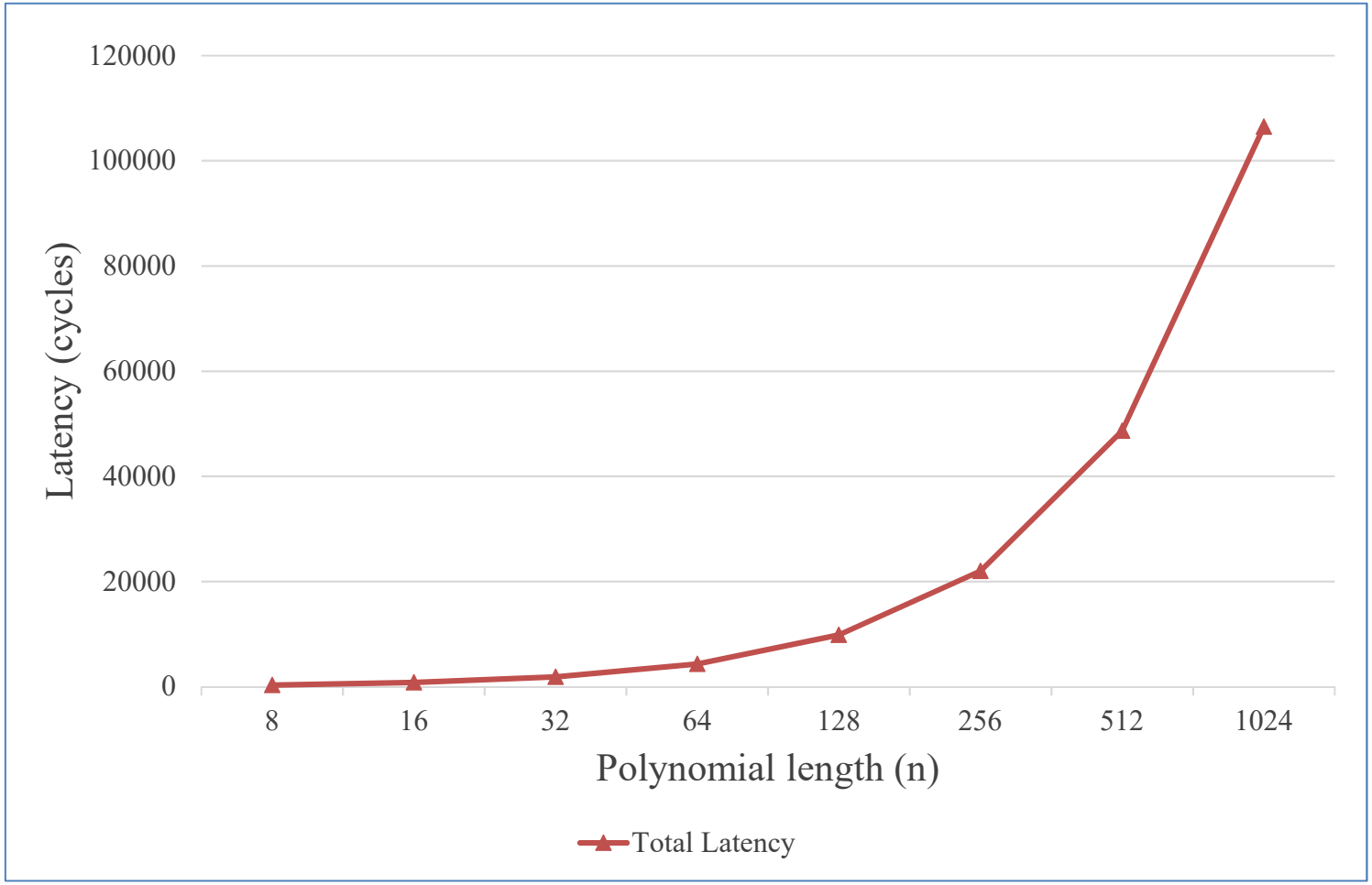
- BRAM Implementation

Length (n)	LUTs	Registers	DSP	BRAM
128	7376	221	26	3.5
256	9152	396	26	3.5
512	11504	674	26	3.5
1024	15717	1255	26	3.5

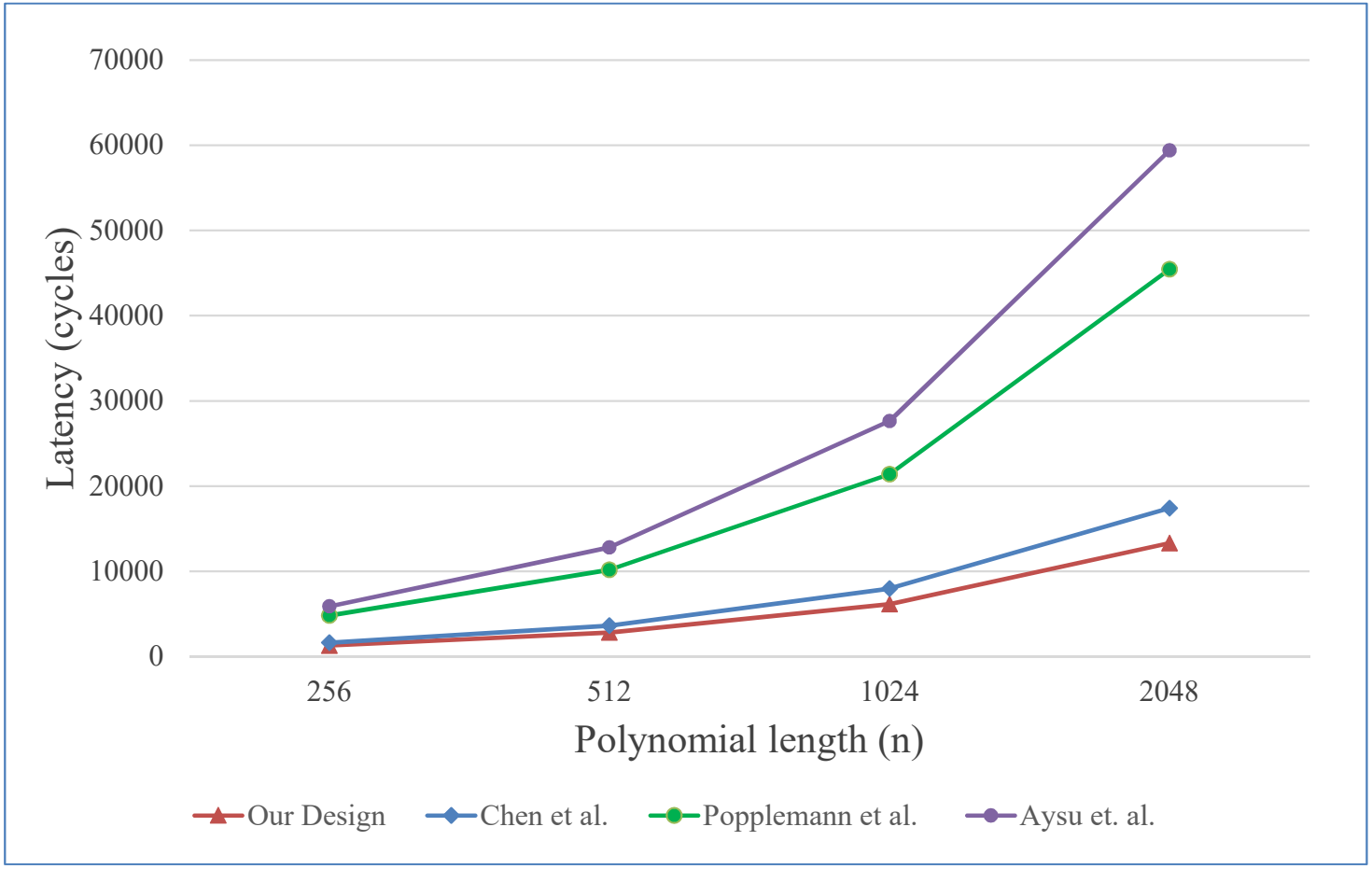
Hardware Cost: Varying q and n values



PKC System Total Latency



NTT Multiplier Latency Comparison



Presentation Outline

- Motivation: why quantum-proof?
- NIST: steps towards standardization
- State of the Art: main algorithm
- FPGA-based Implementation: primitives
- Evaluation: cost and performance
- Key Contributions: conclusion

Conclusion

- Implementation
  - FGPA-tailored implementation of primitives
- Optimization
  - Algorithmic optimizations to reduce hardware cost
- Open Source
  - Release of the synthesizable and fully verifiable Verilog code with following advantages:
    - Parameterization
      - Enable deployment in small devices as well as large platforms
    - Fast Polynomial Multiplier
      - Efficient n-point NTT multiplier

Acknowledgements

- All ASCS lab members

Thank you

- Code available at: http://ascslab.org/research/pqcp/index.html