Post-Quantum Cryptographic Hardware Primitives on FPGAs

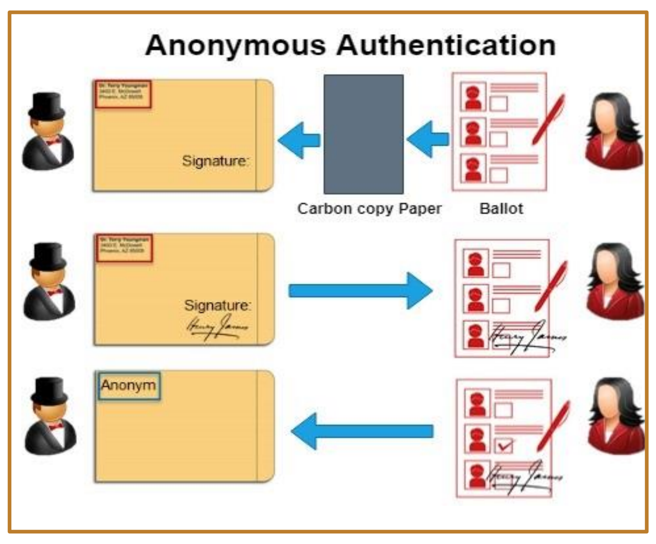
Rashmi Agrawal, Lake Bu, and Michel A. Kinsy

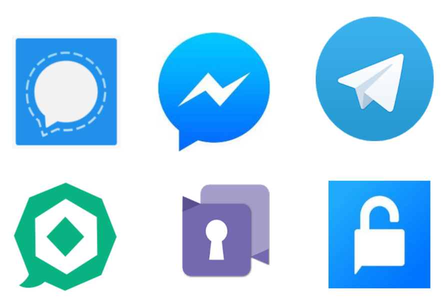
BOSTON UNIVERSITY

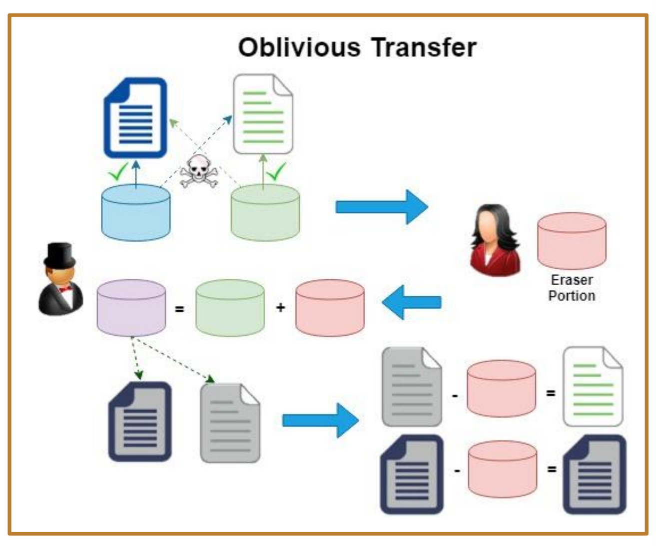
ASCS ADAPTIVE & SECURE COMPUTING SYSTEMS LABORATORY

Applications

- HTTPS
- Digitally signed PDFs
- Homomorphic Encryption
- Secure IMs: Signal, FB Messenger, Telegram, Cyphr, Silence, etc.

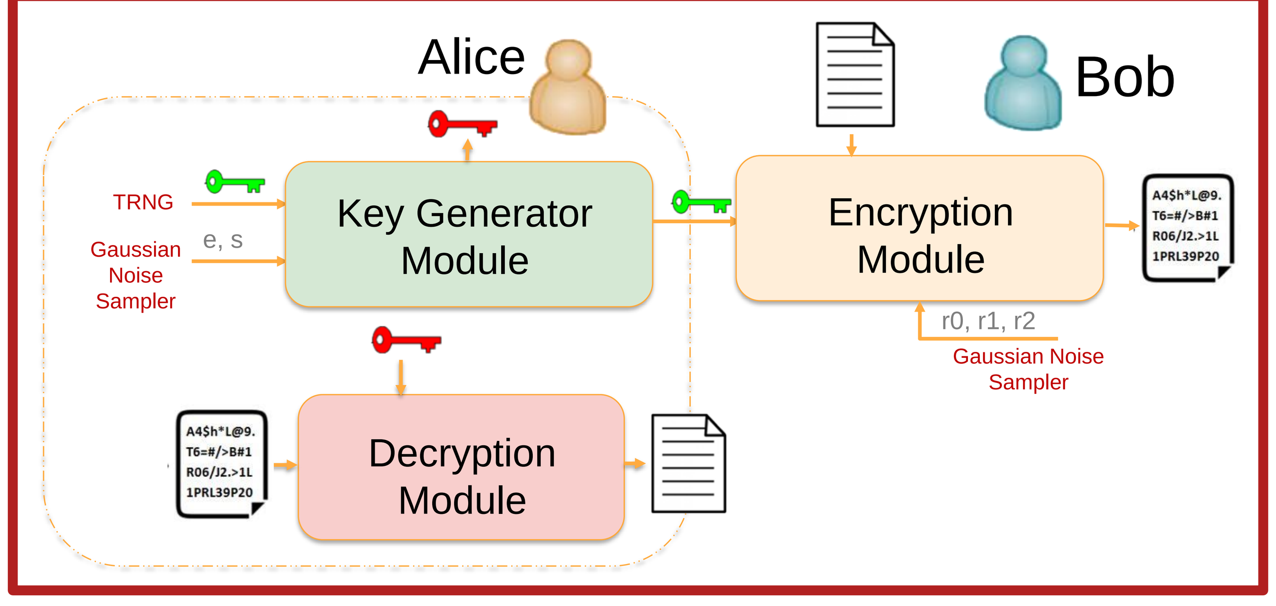




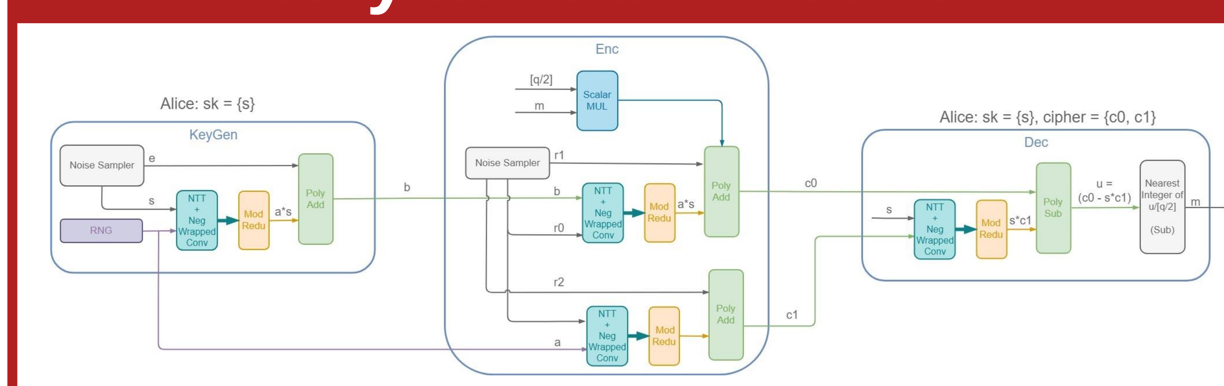


- Tor Browser
- Next-Gen Blockchain
- Secure DNA Query
- Privacy Preserving Machine Learning

Public-Key Cryptosystem



Key Modules of PKC



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
- Symbol: within the prime number 1,049,089

Polynomial Multiplication

Approach-1

- Naïve Convolution with Polynomial Reduction
- Complexity: $O(N^2)$

Approach-2

- Number-Theoretic Transform over finite field
- Negative Wrapped Convolution
- Optimized for FPGA platforms
- Complexity:
  - O(N log N)

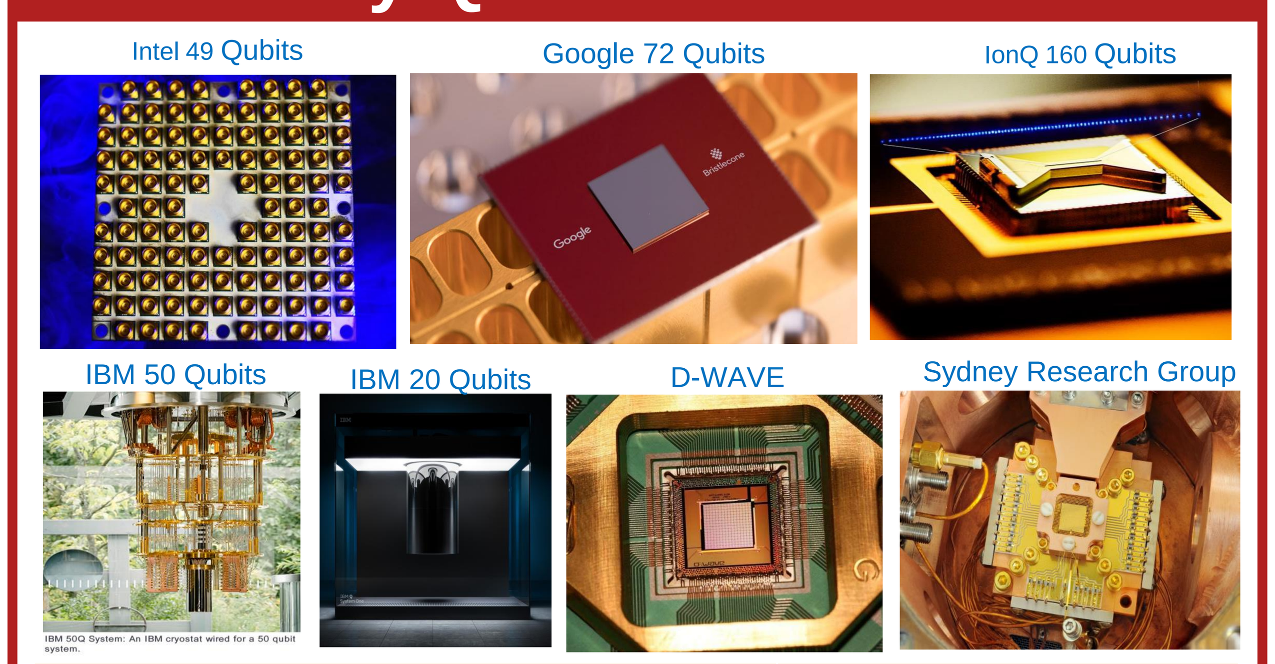
Algorithm Polynomial multiplication using FFT

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 $a_i, b_i, c_i \in \mathbb{Z}_p, i = 0, 1, \ldots, n-1$.

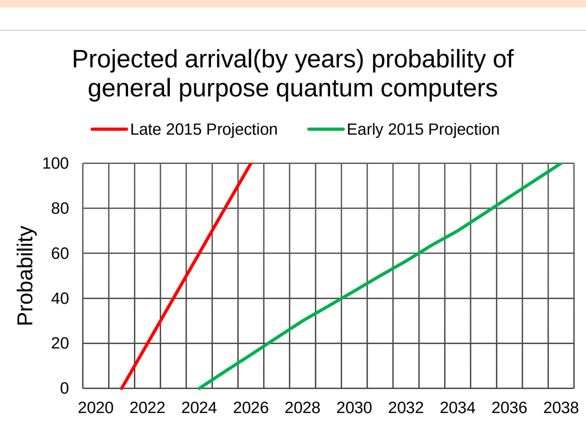
Input: $\mathbf{a}, \mathbf{b}, \omega, \omega^{-1}, \phi, \phi^{-1}, n, n^{-1}, p$.
Output: $\mathbf{c}$ where $c(x) = a(x) \cdot b(x) \bmod \langle x^n + 1 \rangle$.

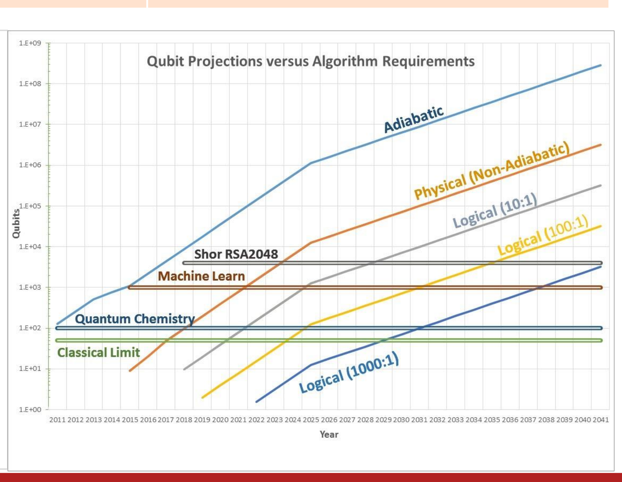
Precompute: $\omega^i, \omega^{-i}, \phi^i, \phi^{-i}$ where $i = 0, 1, \ldots, n-1$
for $i = 0$ **to** $n-1$ **do**
Negative Wrapped Convolution (NWC) → $\bar{a}_i \leftarrow a_i \phi^i \bmod p$
$\bar{b}_i \leftarrow b_i \phi^i \bmod p$
end for
Number-Theoretic Transform (NTT) → $\bar{\mathbf{A}} \leftarrow \mathrm{FFT}_\omega^n(\bar{\mathbf{a}})$
$\bar{\mathbf{B}} \leftarrow \mathrm{FFT}_\omega^n(\bar{\mathbf{b}})$
for $i = 0$ **to** $n-1$ **do**
Component-wise multiplication → $\bar{C}_i \leftarrow \bar{A}_i \bar{B}_i \bmod p$
end for
Inverse NTT → $\bar{\mathbf{c}} \leftarrow \mathrm{IFFT}_\omega^n(\bar{\mathbf{C}})$
for $i = 0$ **to** $n-1$ **do**
Inverse NWC → $c_i \leftarrow \bar{c}_i \phi^{-i} \bmod p$
end for
return $\mathbf{c}$

Why Quantum-Proof?



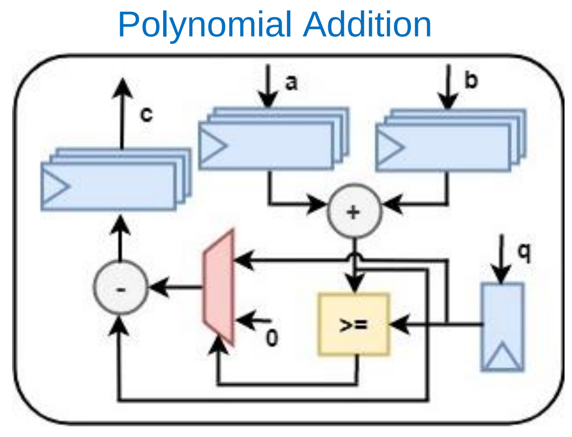
Algorithm	Secure in Post-quantum Era?
RSA-1024, -2048, -4096	No
Elliptic Curve Crypto (ECC) -256, -521	No
Diffie-Hellman	No
ECC Diffie-Hellman	No
AES-128, -192	No



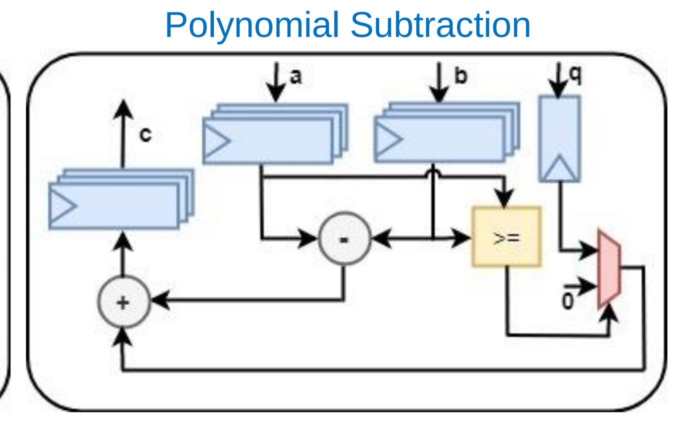


Implementation

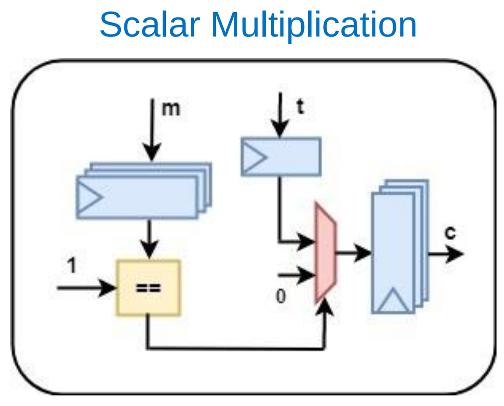
Polynomial Addition



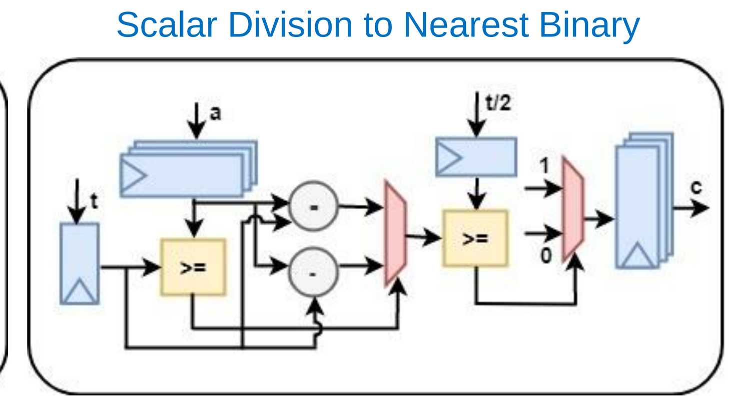
Polynomial Subtraction



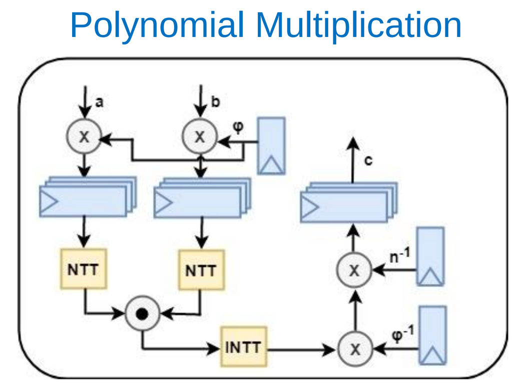
Scalar Multiplication



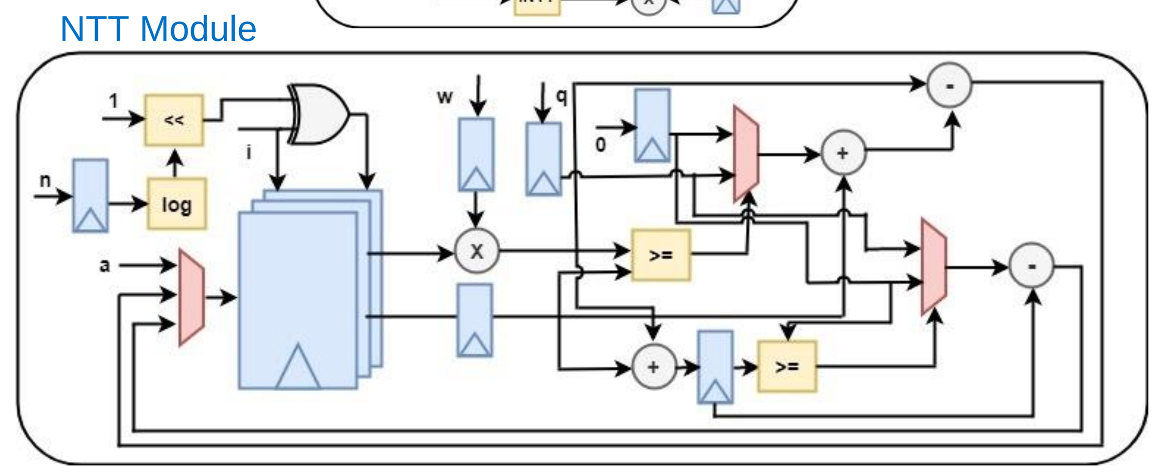
Scalar Division to Nearest Binary



Polynomial Multiplication



NTT Module



NIST's Standardization Steps

Public-Key Encryption		Key Establishment	
NTRU Prime	(R-lattice)	NewHope	(R-LWE)
NTRU *European Encryption Standard*	(R-lattice)	NTRU *European Encryption Standard*	(R-lattice)
LAC	(R-LWE)	FrodoKEM	(R-LWE)
SABER	(ModLWR)	CRYSTALS-KYBER	(R-LWE)
Round5	(R-LWR)	SABER	(Mod-LWR)
		Three Bears	(Mod-LWE)

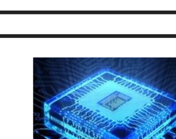
Design Workflow

Algorithmic optimizations for hardware implementation

Basic operator implementation i.e. polynomial addition, subtraction, multiplication, division

FSM design of three sub-modules i.e. Key Generation, Encryption and Decryption

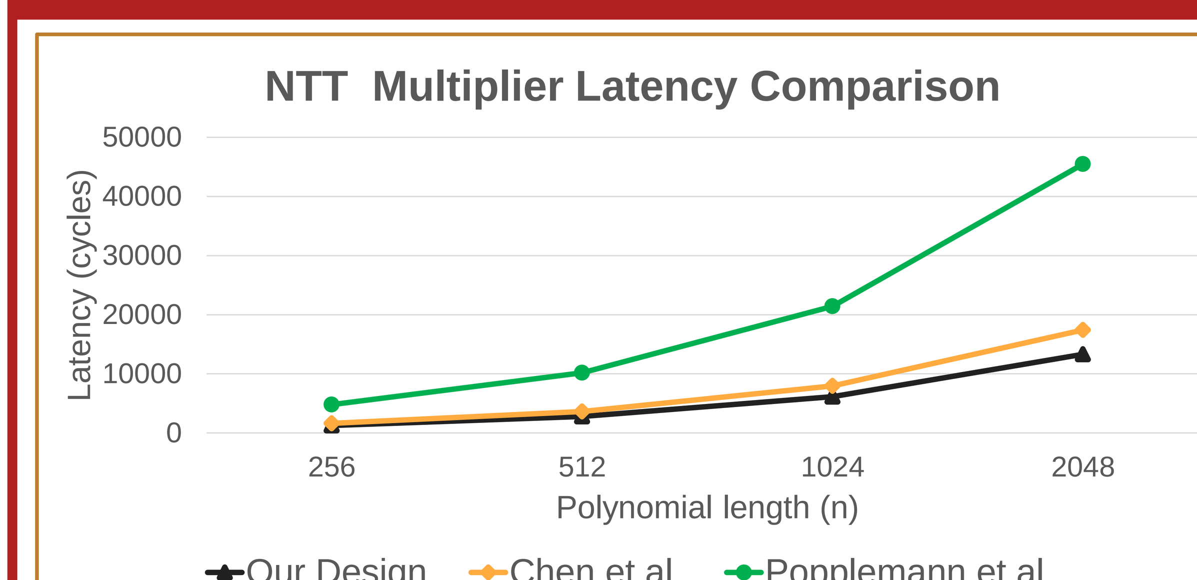
Integration, testing, and final deliverable

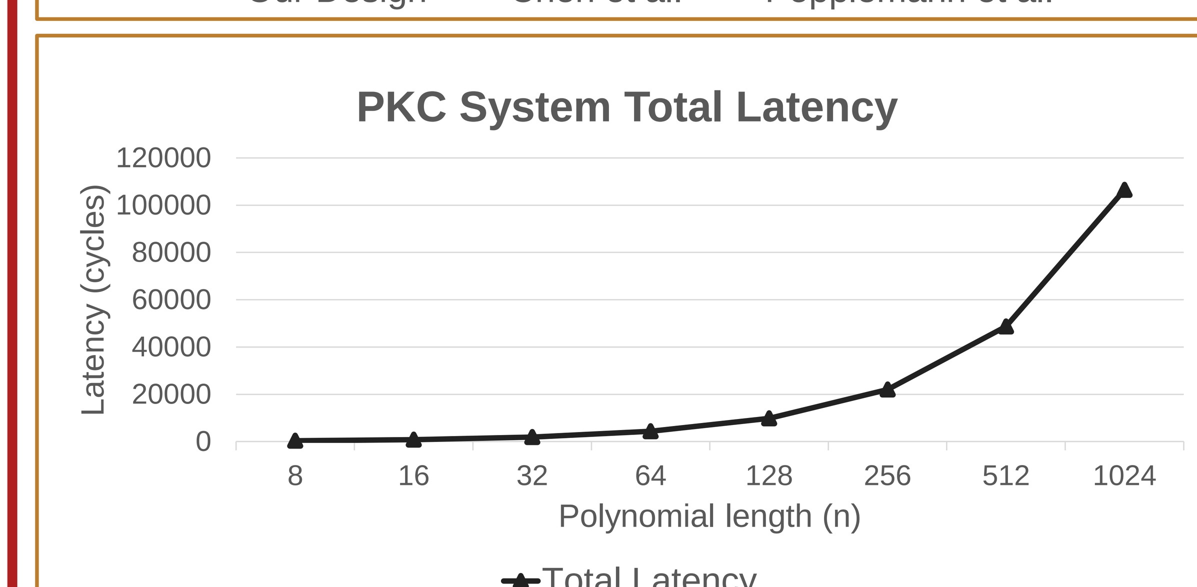


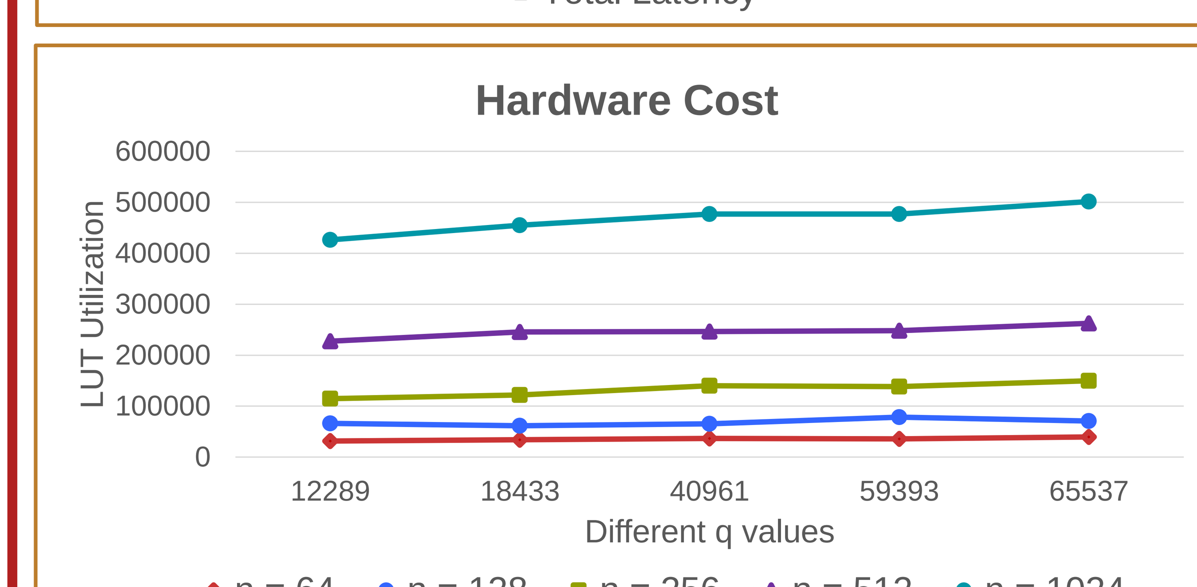
Why Ring-Learning with Errors?

- A branch of lattice-based cryptosystems
- Able to perform
  - Public-key encryption
  - Key-exchange
  - Digital Signature
- Able to build Somewhat Homomorphic Encryption (SHE)
- Used for quantum computation verification
- Smaller key size (7k~15k bits vs. 1MB for code-based & 1TB for "post-quantum RSA")
- Simpler computation and circuits

Results







Latency Equations based on {q, n}

Operation	Latency
KeyGen	$3n + \frac{3n}{2}\log n$
Enc	$7n + 2n\log n$
Dec	$4n + n\log n$

Area Equations based on {q, n}

Resource	Cost
LUTs	$O(n\, log\, n \log q)$
Registers	$O(n\, log\, n \log q)$

Hardware Cost with different n and q values

n	q	LUTs	Registers	DSP	BRAM
32	193	4352	169	4	0
64	257	5828	215	4	0
128	769	5420	211	26	3.5
256	10753	8311	394	26	3.5
512	12289	11504	674	26	3.5
1024	65537	21423	1304	30	3.5