Ring-Learning with Errors (R-LWE) Post-Quantum Cryptographic Hardware Primitives

Lake Bu, **Rashmi S. Agrawal (RSA)**, Hai Cheng, Michel A. Kinsy
Adaptive & Secure Computing Systems Lab
Boston University

Presentation Flow

- Motivation: why quantum-proof?
- State of the Art
- What we are proposing
  - Advance Reduction Multiplier (ARM)
- Applications
  - Oblivious Transfer – privacy-preserving ML
  - Zero-Knowledge Proof – a new generation of Blockchain

Presentation Flow

- Motivation: why quantum-proof?
- State of the Art
- What we are proposing
  - Advance Reduction Multiplier (ARM)
- Applications
  - Oblivious Transfer – privacy-preserving ML
  - Zero-Knowledge Proof – a new generation of Blockchain

When Quantum Computers Come …

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

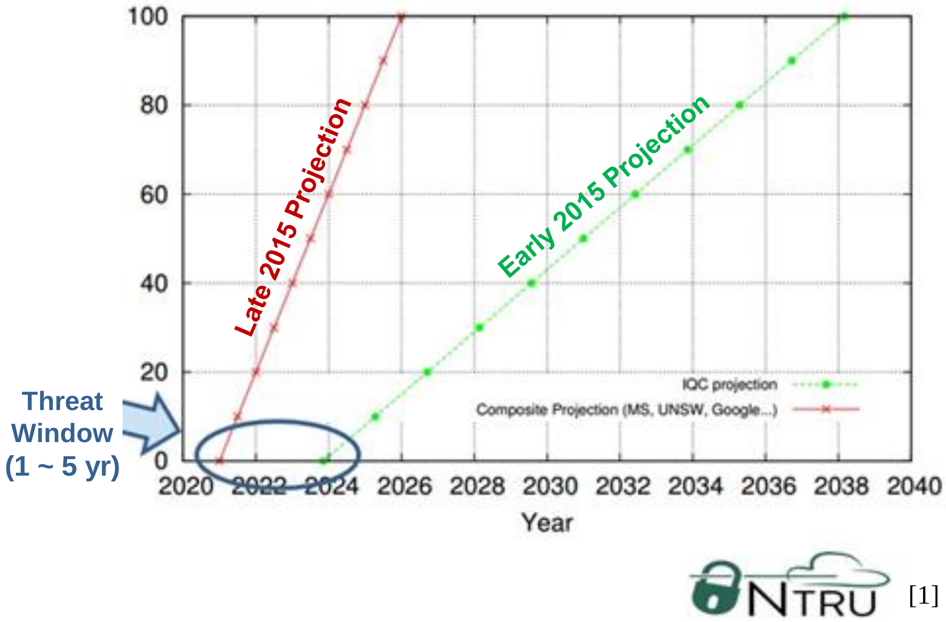
Algorithm	Secure in Post-quantum Era?
RSA-1024, -2048, -4096	No
Elliptic Curve Crypto (ECC)-256, -521	No
Diffie-Hellman	No
ECC Diffie-Hellman	No
AES-128, -192	No

[1]

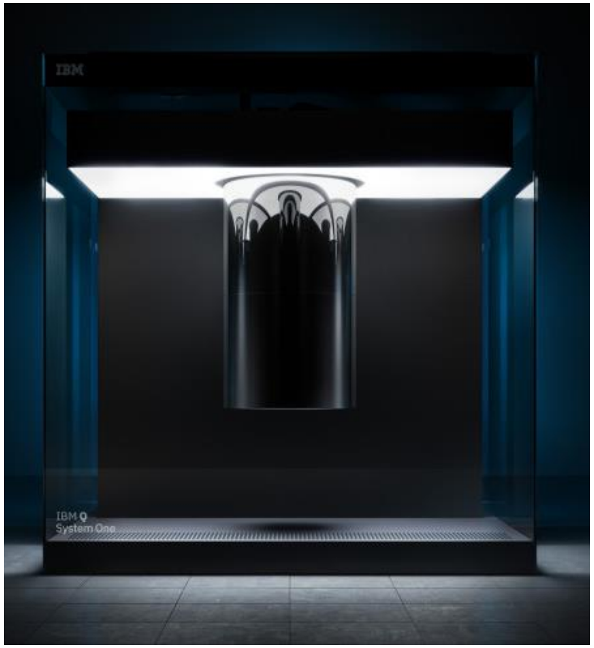
[1] https://www.nist.gov/

How Far Is It From Us?

- **The Projected Timeline of General Quantum Computers**



[1]



IBM Q System (20-qubit), Jan 2019

[1] Lee Wilson, The Rise of Quantum Computers – The Current State of Cryptographic Affairs, 2016

Presentation Flow

- Motivation: why quantum-proof?
- State of the Art
- What we are proposing
  - Advance Reduction Multiplier (ARM)
- Applications
  - Oblivious Transfer – privacy-preserving ML
  - Zero-Knowledge Proof – a new generation of Blockchain

Post-Quantum Cryptography (PQC) Standardization

- NIST
  - Jan 2017 - present
  - Evaluating 69 (5 withdrawn) submissions of PQC, to bring up a standard (just like AES or RSA):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

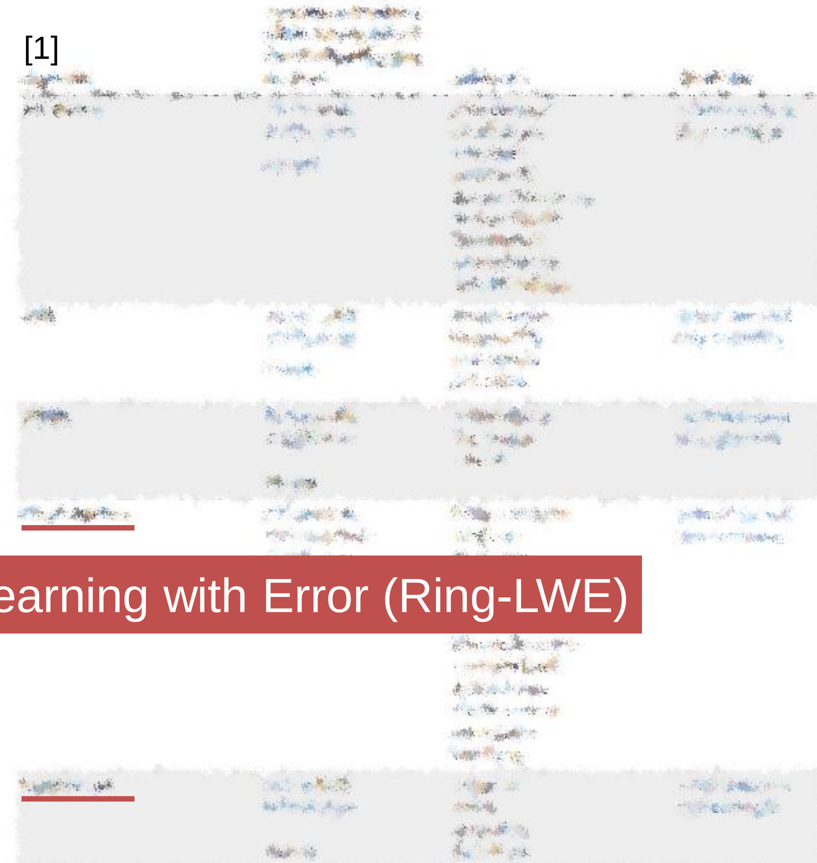
Submission deadline Nov 30, 2017. List updated Dec 20, 2018.

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

Post-Quantum Cryptography (PQC) Standardization

- NIST
  - Jan 2017 - present
  - 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

Why Ring-LWE?

- Advantages
  1) A branch of lattice-based cryptosystem
  2) Able to do public-key encryption
  3) Able to do key-exchange mechanism
  4) Able to build homomorphic encryption (HE)
  5) Used for quantum computation verification and HE[1]
  6) Smaller key size (7k~15k bits vs. 1MB for code-based & 1TB for “post-quantum RSA”)
  7) Simpler computation & circuits

[1] Urmila Mahadev, Classical Homomorphic Encryption for Quantum Circuits, 2018

Ring-Learning with Error (Ring-LWE)

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

Since e is small, $a \cdot s$ is a very close vector to b. Thus it is solving CVP, even worse than SVP in Lattice Problem.

But with s, it is easy for Alice to verify equation (1), making it a good trapdoor.

- Public-key Cryptosystem (PKC)[1]
  - Setup (Alice)
    - Let q be a prime. In a ring Rq, 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 $w = \left\lfloor \frac{q}{2} \right\rfloor$
    - 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} as the public key, as well as w = $\left\lfloor \frac{q}{2} \right\rfloor$
    - 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 + wm;
      - 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} as the public key, as well as $w = \left\lfloor \frac{q}{2} \right\rfloor$
    - 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:
      - $c0 = b \cdot r0 + r2 + wm;$
      - $c1 = a \cdot r0 + r1$
  - Decryption (Alice computes):
    - $c0 - s \cdot c1 = b \cdot r0 + r2 + wm - s \cdot a \cdot r0 - s \cdot r1$ (2)
      $= wm + e \cdot r0 + r2 - s \cdot r1 = wm +$ "small"
    - $m = \lfloor (c0 - s \cdot c1)/w \rceil$

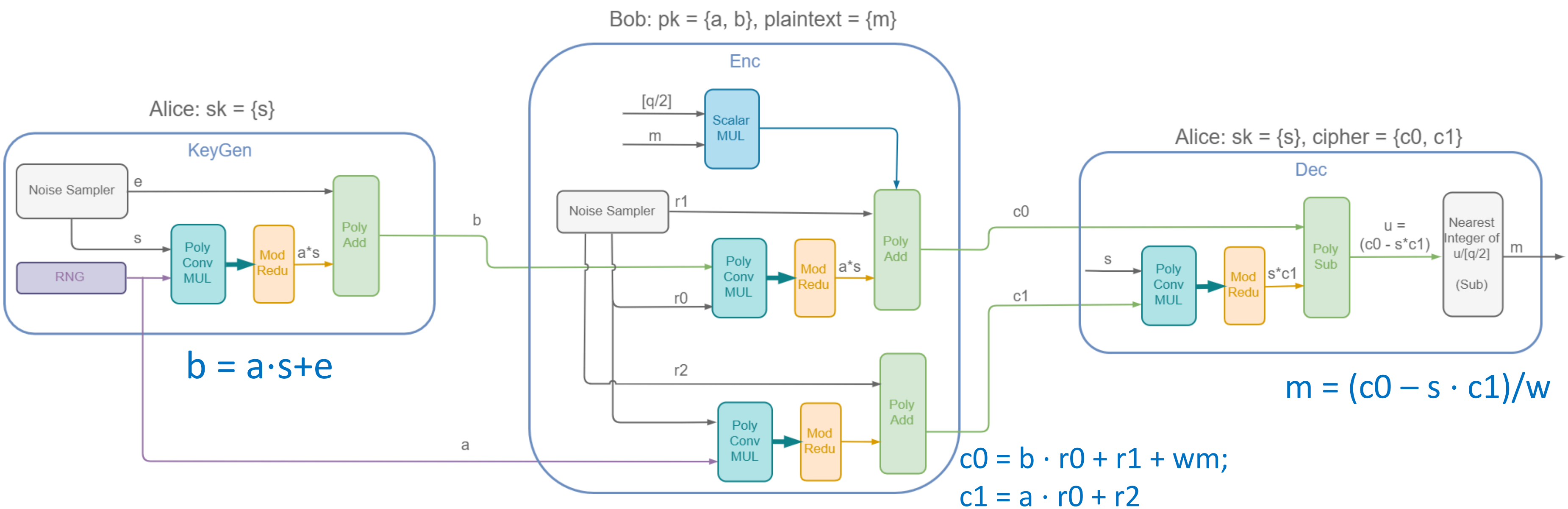
Taking the nearest integer

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

Ring-Learning with Error (Ring-LWE)

- Architecture
  - Key Generation
  - Encryption
  - Decryption



$b = a \cdot s + e$

$m = (c0 - s \cdot c1)/w$

$c0 = b \cdot r0 + r1 + wm;$
$c1 = a \cdot r0 + r2$

Ring-Learning with Error (Ring-LWE)

- Basic Operations

  (Every operation is modular)

  - 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

Ring-Learning with Error (Ring-LWE)

- Basic Operations

  (Every operation is modular)

  - 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

Ring-Learning with Error (Ring-LWE)

- 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 $\phi^2 \equiv \omega \mod p$. 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 $c(x) = a(x) \cdot b(x) \mod \langle x^n + 1 \rangle$.

```
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_ω^n(ā)
B̄ ← FFT_ω^n(b̄)
for i = 0 to n-1 do
    C̄_i ← Ā_i B̄_i  mod p
end for
c̄ ← IFFT_ω^n(C̄)
for i = 0 to n-1 do
    c_i ← c̄_i φ^{-i}  mod p
end for
return c
```

Negative Wrapped Convolution (NWC) →

Fast Number Theoretic Transform (NTT) →

Component-wise multiplication →

Inverse NTT →

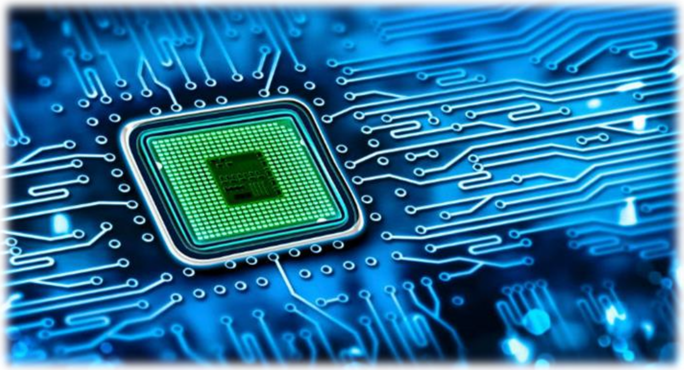
Inverse NWC →

Ring-Learning with Error (Ring-LWE)

- Modular Polynomial Multiplication
  - Hardware cost and latency

Length	LUT	Slice	DSP	BRAM	Cycles
512	3750	1348	4	4	3622
1024	6689	2112	4	8	7976

[1]



[1] Chen, Donald Donglong, et al. "High-Speed Polynomial Multiplication Architecture for Ring-LWE and SHE Cryptosystems." *IEEE Trans. on Circuits and Systems* 62.1 (2015): 157-166.

Presentation Flow

- Motivation: why quantum-proof?
- State of the Art
- What we are proposing
  - Advance Reduction Multiplier (ARM)
- Applications
  - Oblivious Transfer – privacy-preserving ML
  - Zero-Knowledge Proof – a new generation of Blockchain

Ring-Learning with Error (Ring-LWE)

- Modular Polynomial Multiplication with Advance Reduction
  - Computing the general representation of the product in advance
  - Computing the primitive polynomial reduction in advance

Algorithm : Polynomial Multiplication Using Advance Reduction Multiplier (ARM)

Let $a = \{a_0, \cdots, a_{n-1}\}, b = \{b_0, \cdots, b_{n-1}\} \in \mathbb{Z}_q[x]/\langle f(x)\rangle$ (where $f(x) = x^n + 1$) be two n-bit vectors, and $P(X)$ the primitive polynomial of the ring.

Let $d = \{d_0, \cdots, d_{n-1}\}, e = \{e_0, \cdots, e_{n-1}\}$ where d_i, e_i are just variable names.

Precompute:

Compute the general representation of the product (2n symbols)

$\hat{c} \leftarrow d \circledast e$ ($\circledast$ for convolution) such that

$\hat{c} = \hat{c}_0 + \hat{c}_1 x^1 + \cdots \hat{c}_{n-1} x^{n-1} + \hat{c}_n x^n + \cdots \hat{c}_{2n-2} x^{2n-2}$

\# Approach 1: by using P(x) for reduction

Reduce the product with primitive polynomial (n symbols)

for i=n to 2n-2 do

$x^i = l_0 + l_1 x^1 + \cdots + l_{n-1} x^{n-1}$

By substituting $\{x^n, \cdots x^{2n-2}\}$ to $\hat{c}$

$c = c_0 + c_1 x^1 + \cdots c_{n-1} x^{n-1}$

\# Approach 2: by using f(x) for reduction

$c \leftarrow \hat{c} / f(x) = c_0 + c_1 x^1 + \cdots c_{n-1} x^{n-1}$

Acquire the general representation of each symbol

Denote $c_i = g_i(d, e)$, where g_i is a general representation of c_i by d, e.

Substitute the multiplicands to get the product

for i=0 to n-1 do

$c_i \leftarrow g_i(a, b)$

end for

return c

Ring-Learning with Error (Ring-LWE)

- Modular Polynomial Multiplication with Advance Reduction
  - A Toy Example
    - $q = 17$, $n = 3$, $f(x) = x^3+1$

1) Pre-compute the general representation of $c = a \circledast b$

$$a = a_0+a_1x+a_2x^2,\ b = b_0+b_1x+b_2x^2,$$
$$c = a \circledast b = a_0b_0+(a_0b_1+a_1b_0)x+(a_0b_2+a_2b_0+a_1b_1)x^2 + (a_1b_2+a_2b_1)x^3+a_2b_2x^4$$

2) Pre-compute with the Primitive Polynomial the reduction of x^i, $i \geq n$

$$P(x) = x^3+x+3$$
$$x^3= 16x+14,\ x^3= 16x^2+14x$$

3) Pre-acquire the general representation of c

$$c = a \circledast b = c_0+c_1x+c_2x^2$$
$$\begin{cases} c_0=a_0b_0+14a_1b_2+14a_2b_1 \\ c_1=a_0b_1+a_1b_0+16a_1b_2+16a_2b_1+14a_2b_2 \\ c_2=a_0b_2+a_1b_1+a_2b_0+16a_2b_2 \end{cases}$$

4) Real-time: substitute the actual values of $\{a_i, b_i\}$ to compute c_i
   - In one step
   - Computing of all c_i can be made fully parallel

Ring-Learning with Error (Ring-LWE)

- Latency Estimation

Multiplier with NTT + NWC

Length	LUT	Slice	DSP	BRAM	Cycles
512	3750	1348	4	4	3622
1024	6689	2112	4	8	7976

[1]

Multiplier with ARM

Length	LUT	Slice	DSP	BRAM	Cycles
512	~3000	~1500	4	4	~128
1024	~7000	~2000	4	8	~ 256

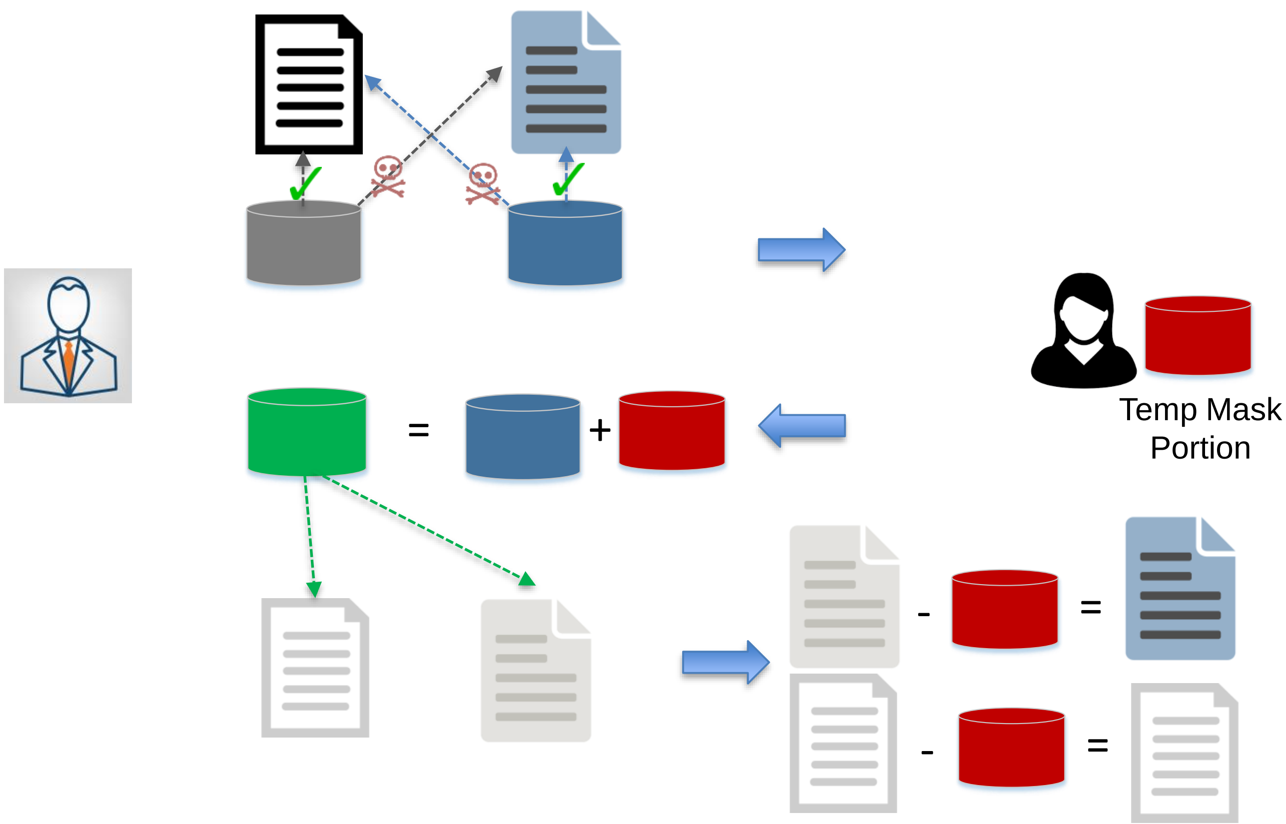
[1] Chen, Donald Donglong, et al. "High-Speed Polynomial Multiplication Architecture for Ring-LWE and SHE Cryptosystems." *IEEE Trans. on Circuits and Systems* 62.1 (2015): 157-166.

Presentation Flow

- Motivation: why quantum-proof?
- State of the Art: how most people are doing it?
- What we are proposing
  - Advance Reduction Multiplier (ARM)
- Applications
  - Oblivious Transfer – privacy-preserving ML
  - Zero-Knowledge Proof – a new generation of Blockchain

Applications

- Oblivious Transfer
  - Illustration: a reader wants to read a doc in a double-blind manner



Basic Concept

- The reader obfuscates her choice;
- The distributer uses her obfuscated choice to mask two docs, where only one can be recovered, and the other is permanently destroyed.

Applications

- Oblivious Transfer
  - MPC
  - DNA database access
  - Garbled circuit
  - Privacy-preserving ML

Alice		Bob
m0, m1, rand0, rand1 Key pair (pk, sk)		Choice: c K
rand0, rand1, pk	$\rightarrow$	
	$\leftarrow$	$V = rand_c + Enc_{pk}(K)$
$M0 = m0+Dec_{sk}(V – rand0)$ $M1 = m1+Dec_{sk}(V – rand1)$	$\rightarrow$	
		$m_c = M_c - K$

Note: (pk, sk) are a public-private key pair in the Ring-LWE PKC scheme.

Applications

- Zero-knowledge Proof
  - Privacy-preserving Blockchain
    - Other parties only know
      - A legal transaction has happened
    - But they know nothing about
      - The sender
      - The recipient
      - Asset class
      - Asset quantity

Alice	Bob
$b = a \cdot s + e$ $c = a \cdot r + m \cdot t + e'$ $t = q/2$	u
$\{a, b\}, m, c$ →	
	← u
$x = r + s \cdot u$ →	
	$\lfloor (c - a \cdot x + b \cdot u)/t \rceil = m$?
* e, e', r, u are sampled small noise	

Any Questions?