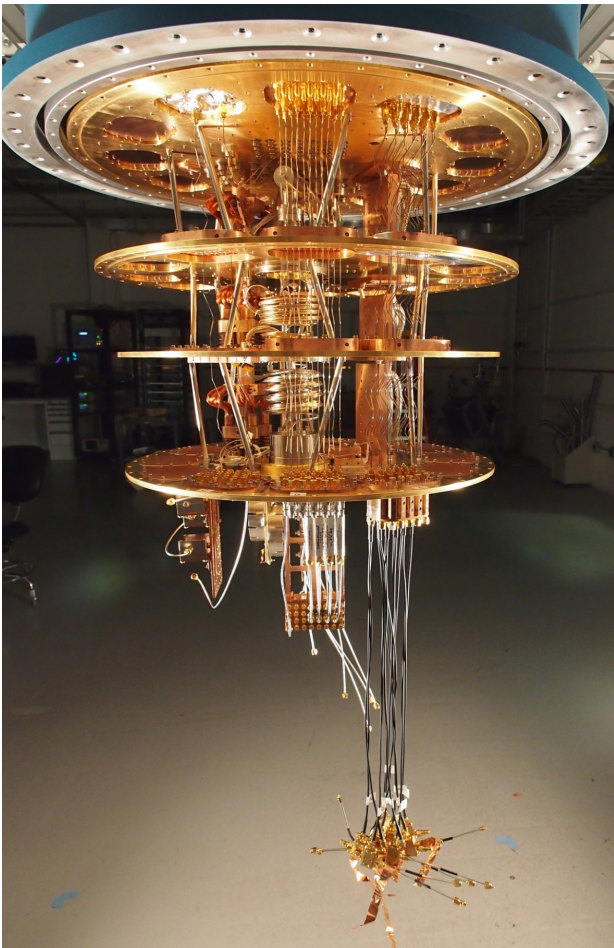


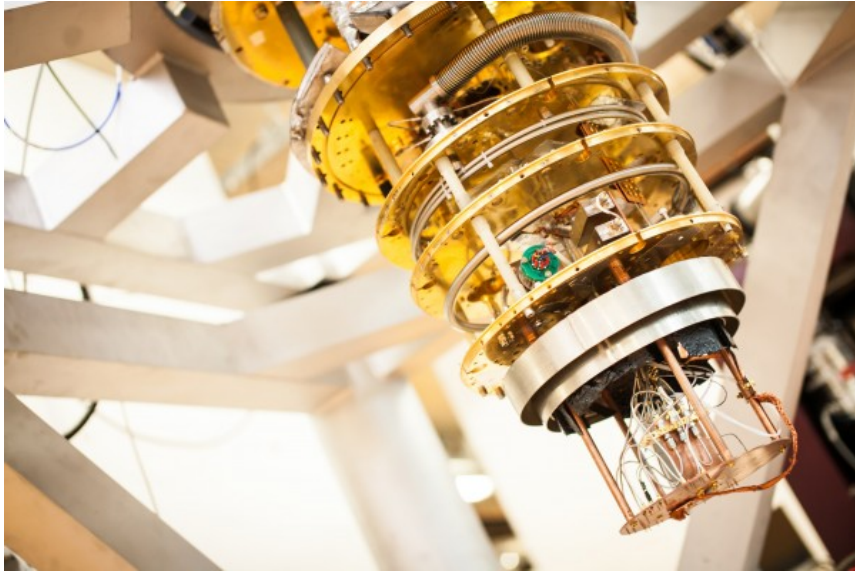
A High-Speed Constant-Time Hardware Implementation of NTRUEncrypt SVES



Farnoud Farahmand,
Malik Umar Sharif, Kevin Briggs
and Kris Gaj

ECE Department
George Mason University
USA

Quantum Computers



- One of ten breakthrough technologies of 2017
- Substantial investments by: Google, IBM, Intel, NTT, Microsoft, Alcatel-Lucent
- Quantum computers based on superconducting circuits operating in the temperature close to absolute 0 (~ 0.01 K)



- November 2017: IBM's 50-qubit chip
- January 2018: Intel's 49-qubit chip, "Tangle-Lake"
- March 2018: Google's 72-qubit chip "Bristlecone"

Quantum Computers & Cryptography

1994: **Shor's Algorithm**, breaks major public key cryptosystems based on

Factoring:

RSA

Discrete logarithm problem (DLP):

DSA, Diffie-Hellman

Elliptic Curve DLP:

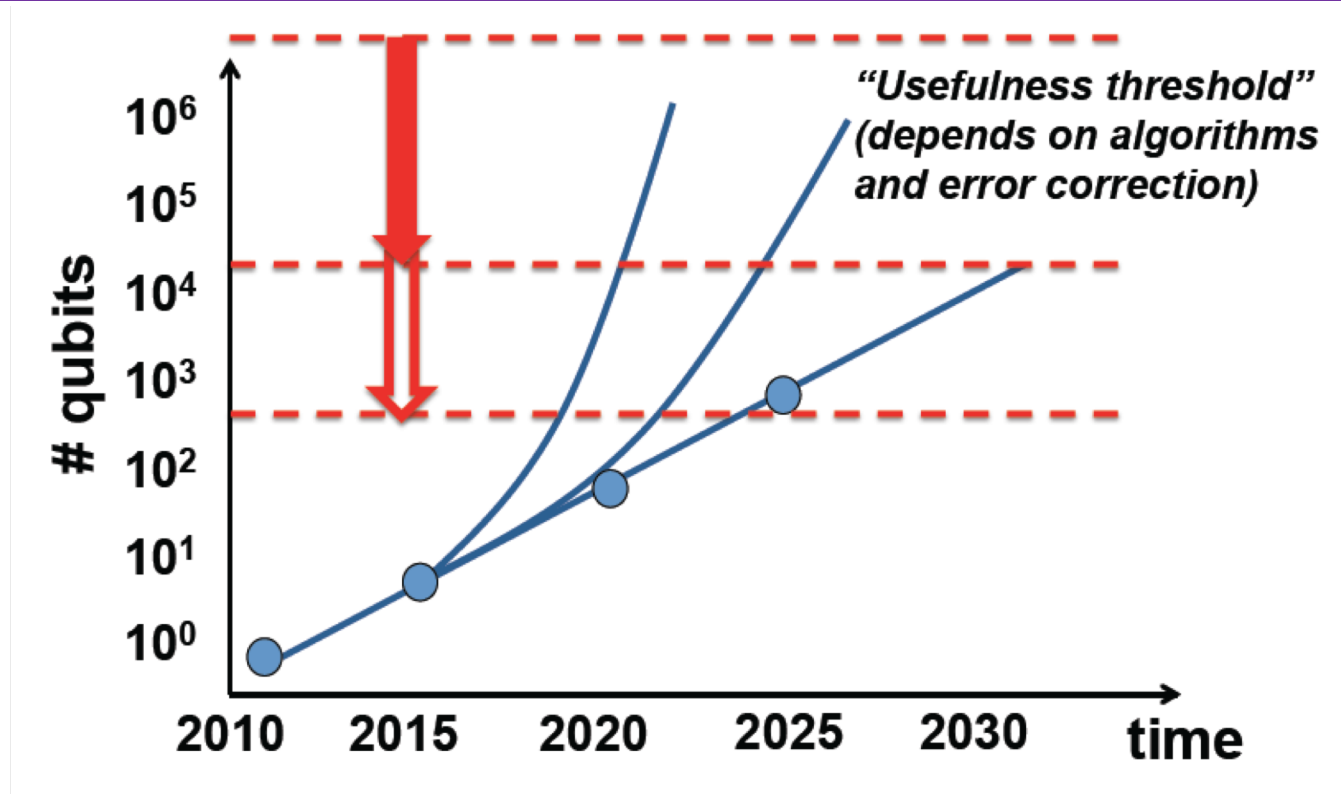
Elliptic Curve Cryptosystems

independently of the key size

assuming

a sufficiently powerful and reliable quantum computer available

How Real Is the Danger?



“There is a 1 in 7 chance that some fundamental public-key crypto will be broken by quantum by 2026, and a 1 in 2 chance of the same by 2031.”

Dr. Michele Mosca

Deputy Director of the Institute for Quantum Computing, University of Waterloo

April 2015

Post-Quantum Cryptography (PQC)

- Public-key cryptographic algorithms for which there are **no known attacks** using quantum computers
 - Capable of being implemented using any **traditional** methods, including **software and hardware**
 - Running efficiently on **any modern computing platforms**: PCs, tablets, smartphones, servers with FPGA accelerators, etc.
- Term introduced by Dan Bernstein in 2003
- Equivalent terms: quantum-proof, quantum-safe or quantum-resistant
- **Based entirely on traditional semiconductor VLSI technology!**

NTRUEncrypt SVES

NTRU – one of the earliest Post-Quantum Cryptosystems
- published in 1998

Standardized by IEEE in 2008 as
NTRUEncrypt Short Vector Encryption Scheme (SVES)

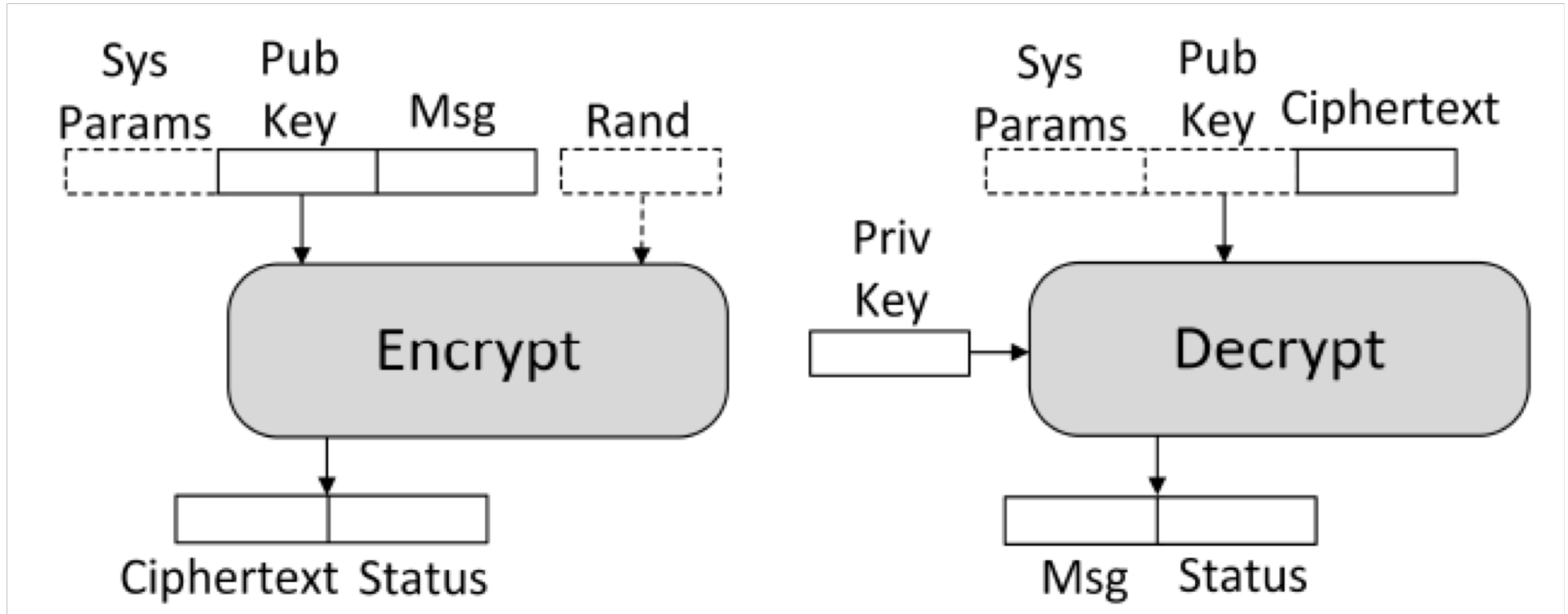
IEEE 1363.1 Standard Specification for Public Key
Cryptographic Techniques Based on Hard Problems over
Lattices

Subsequent standards by:

ANSI – 2010

Consortium for Efficient Embedded Security – 2015

NTRU as a Public Encryption Scheme



Targeted Parameter Sets

- 256-bit security: ees1499ep1:
 - $p=3$, $q=2048$, $N=1499$, $dr=df=79$
- 192-bit security: ees1087ep1:
 - $p=3$, $q=2048$, $N=1087$, $dr=df=63$

Main Operation of NTRU Encryption

$$e = r * h + m \quad \text{mod } x^N - 1$$

Public key:

h – polynomial of degree $N-1$ with N "big" coefficients in the range $[0..q-1]$ $q=2048$

Message:

m – polynomial of degree $N-1$ with N "small" coefficients in the set $\{-1, 0, 1\}$

Random value:

r – polynomial of degree $N-1$ with exactly d_r coefficients equal to 1, d_r coefficients equal to -1, and the remaining coefficients equal to 0.

Polynomial Multiplication mod x^N-1

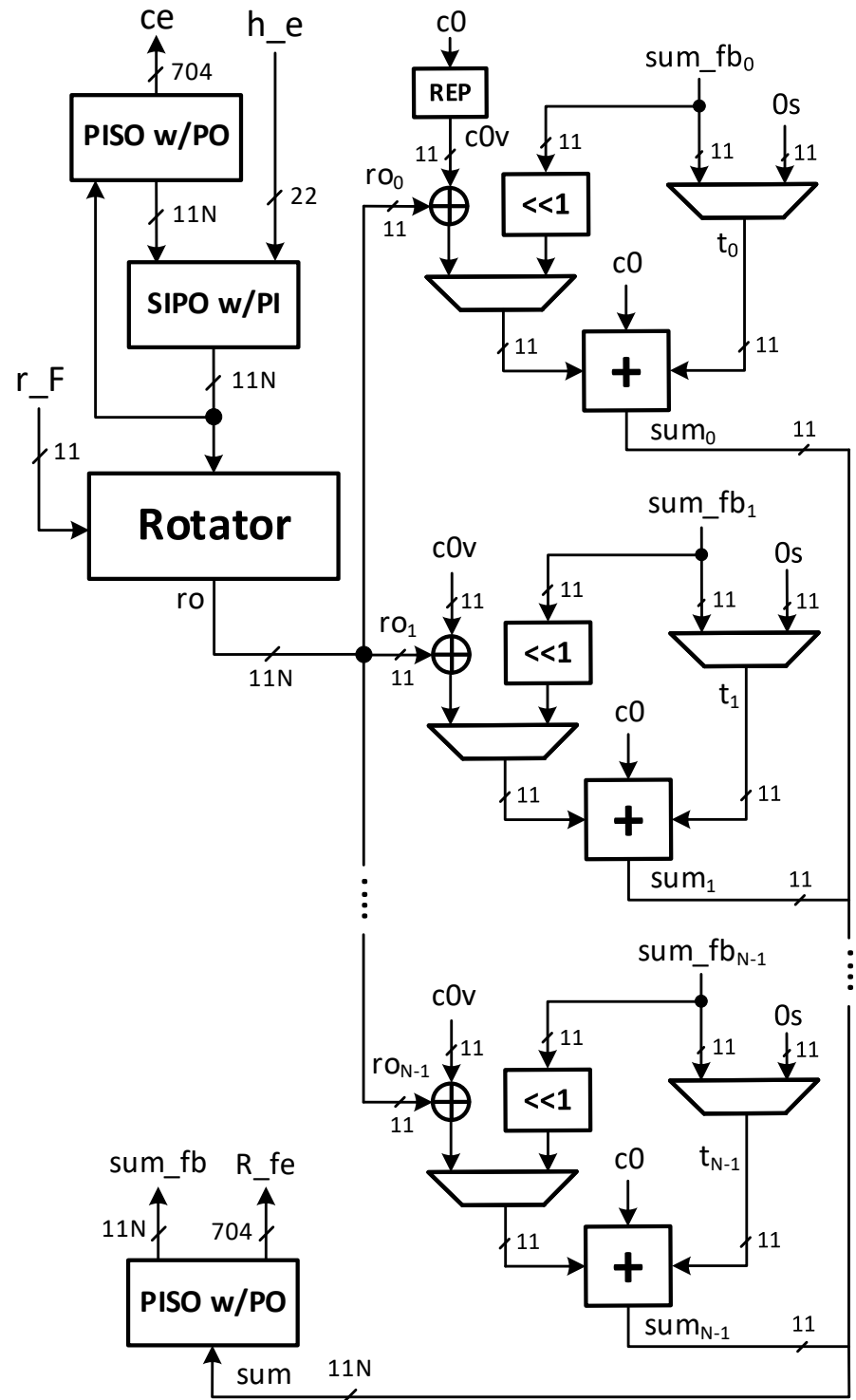
$$R = r * h \quad \text{mod } x^N-1$$

Each coefficient of r equal to 1 contributes to the results
 $h \lll \{\text{position of } 1\}$

Each coefficient of r equal to -1 contributes to the results
- ($h \lll \{\text{position_of_}-1\}$)

Rotation can be performed using barrel rotator
in a single clock cycle

Polynomial Multiplier



Random Polynomial r

Fully described by dr positions of 1 s and dr positions of -1 s (e.g., $N=1499$, $dr=79$)

Positions determined by the outputs from the Pseudorandom Function

(BPGM - Blinding Polynomial Generation Method)

based on the hash function SHA-2,

with the partially random seed, $sData$.

Pseudo Random Number Generator - BPGM

Given: sData – partially random seed,
Counter with consecutive values C1, C2, ..., Cn
Hash function h() [in our case SHA-2]

BPGM Calculates: n hash function outputs:

$$h(\text{sData} \parallel \text{C1}) = h(\text{sData}_0, \dots, \text{sData}_{t-1}, \text{sData}_t \parallel \text{C1})$$

$$h(\text{sData} \parallel \text{C2}) = h(\text{sData}_0, \dots, \text{sData}_{t-1}, \text{sData}_t \parallel \text{C2})$$

$$h(\text{sData} \parallel \text{C3}) = h(\text{sData}_0, \dots, \text{sData}_{t-1}, \text{sData}_t \parallel \text{C3})$$

.....

$$h(\text{sData} \parallel \text{Cn}) = h(\text{sData}_0, \dots, \text{sData}_{t-1}, \text{sData}_t \parallel \text{Cn})$$

Overlapping computations for t blocks of sData
performed only once; partial result stored

Pseudo Random Number Generator - BPGM

n 256-bit hash function outputs divided into chunks of $c=13$ bits

A c -bit random chunk with a value x is rejected if:

1) $x > \text{cthr} = 2^c - (2^c \bmod N)$

to assure that each value of

$x \bmod N$ (in the range from 0 to $N-1$)

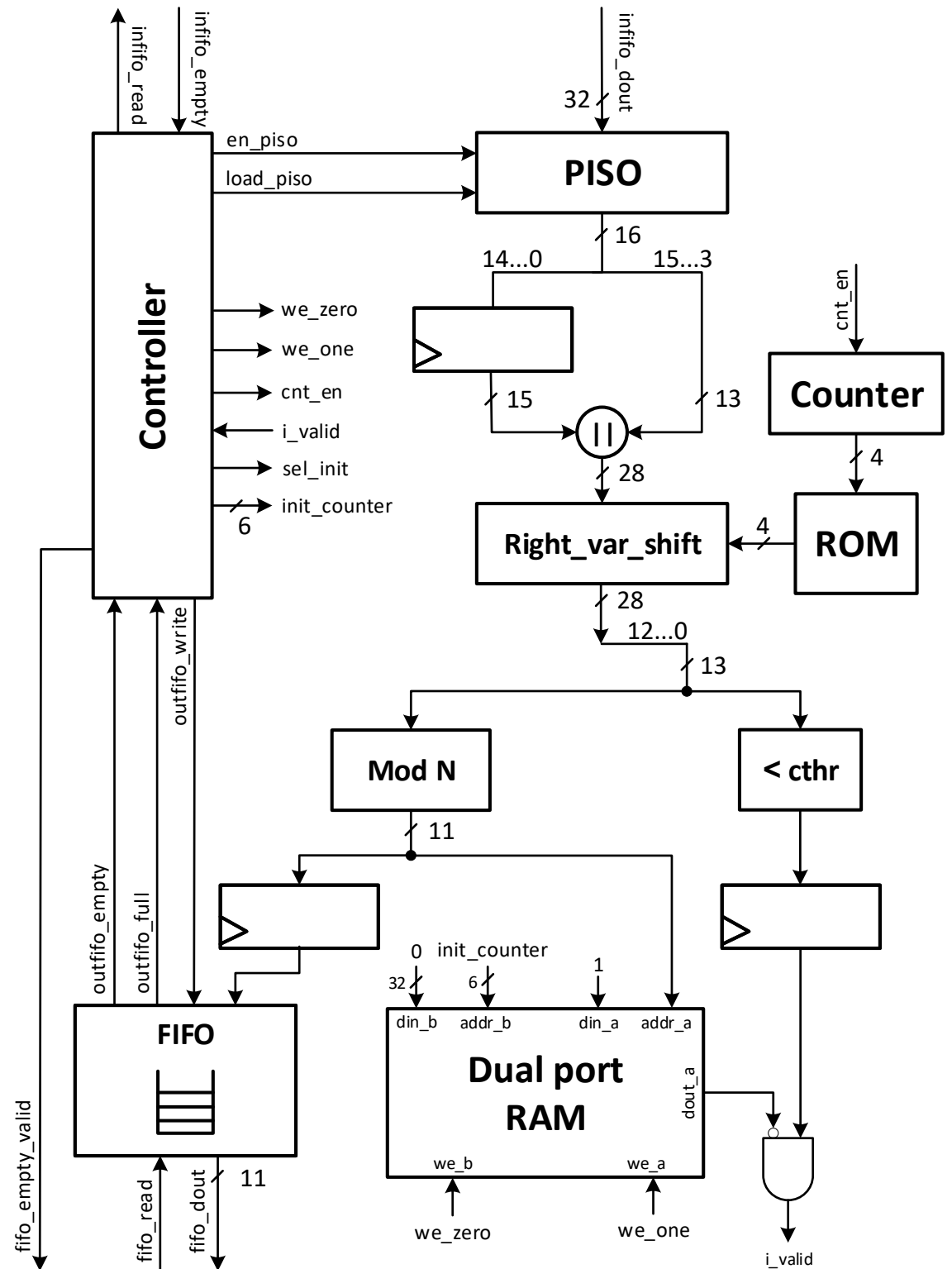
has the same probability

2) $i = x \bmod N$ was already used :

as we are looking for unique positions of 1 and -1s in the range $[0..N-1]$

Problem: Variable execution time

Constant-Time Implementation of BPGM



Solution for Constant-Time Implementation

Process the obtained positions of 1s and -1s only when a sufficient number of such positions accumulated in a FIFO with a very high probability

#SHA-256 outputs	#13-bit output chunks	Assumed minimum # of indices i generated	Probability of success
1	19	14	0.9952
2	39	30	0.9974
3	59	47	0.9955
4	78	62	0.9976
5	98	79	0.9958
6	118	94	0.9984
7	137	110	0.9971
8	157	126	0.9969
9	177	142	0.9964
10	196	158	0.9929
BPGM successful for ees1499ep1 (N=1499, 2df=158)			0.9794

Overhead

The overall probability of success = **0.9794**,

Thus, only in **2.06%** of cases, the entire encryption needs to be repeated with the different value of the random seed b .

In **98%** of cases, timing overhead of a constant-time implementation less than **10%**.

Implementation Assumptions

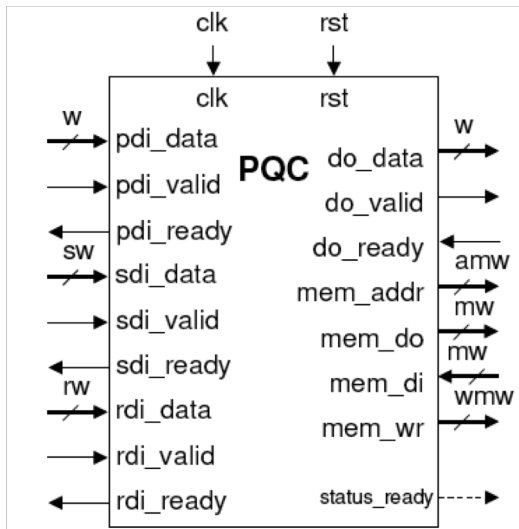
- Optimization for **Speed**
 - **Minimum Latency**
 - **Maximum Number of Operations per Second**
- **Application: high-end servers supporting a very large number of TLS, IPSec, and other protocol transactions**

PQC Hardware API proposed by GMU

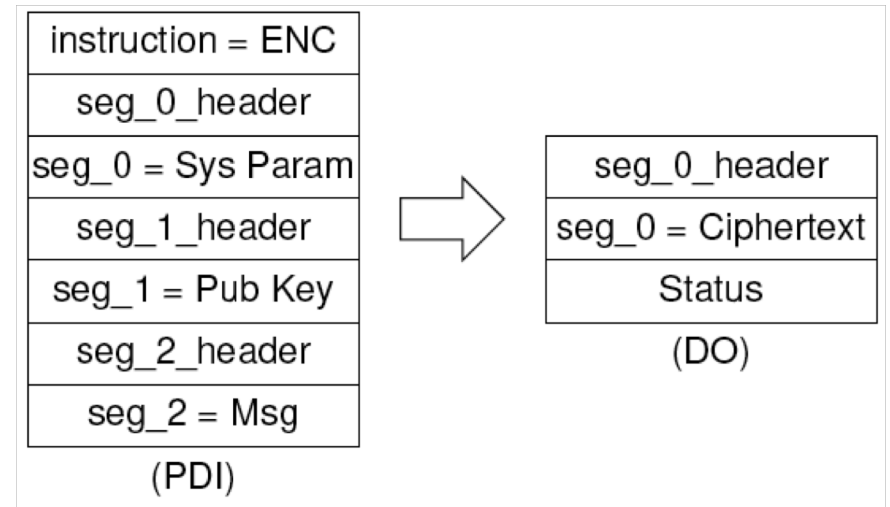
1. Minimum Compliance Criteria

- Encryption & decryption, Signature generation & verification
- Maximum message size
- Padding
- Permitted data port widths, etc.

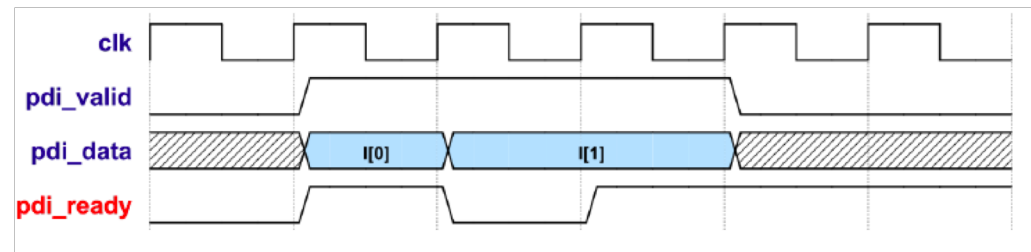
2. Interface



3. Communication Protocol



4. Timing Characteristics



v1: October 2016 (ENC & SIG), v2: April 2018 (+ KEM & lessons)

Implementation Platforms

Hardware Platform:

FPGA Family: Xilinx Virtex UltraScale
Device: xcvu440-flga2892-3-e
Technology: 20nm CMOS

Software Platform:

Intel Xeon CPU E5-2667 v3 @ 3.20GHz
128 GB RAM

Optimized Code in C by
Security Innovation, Inc. (co-authors of NTRU)

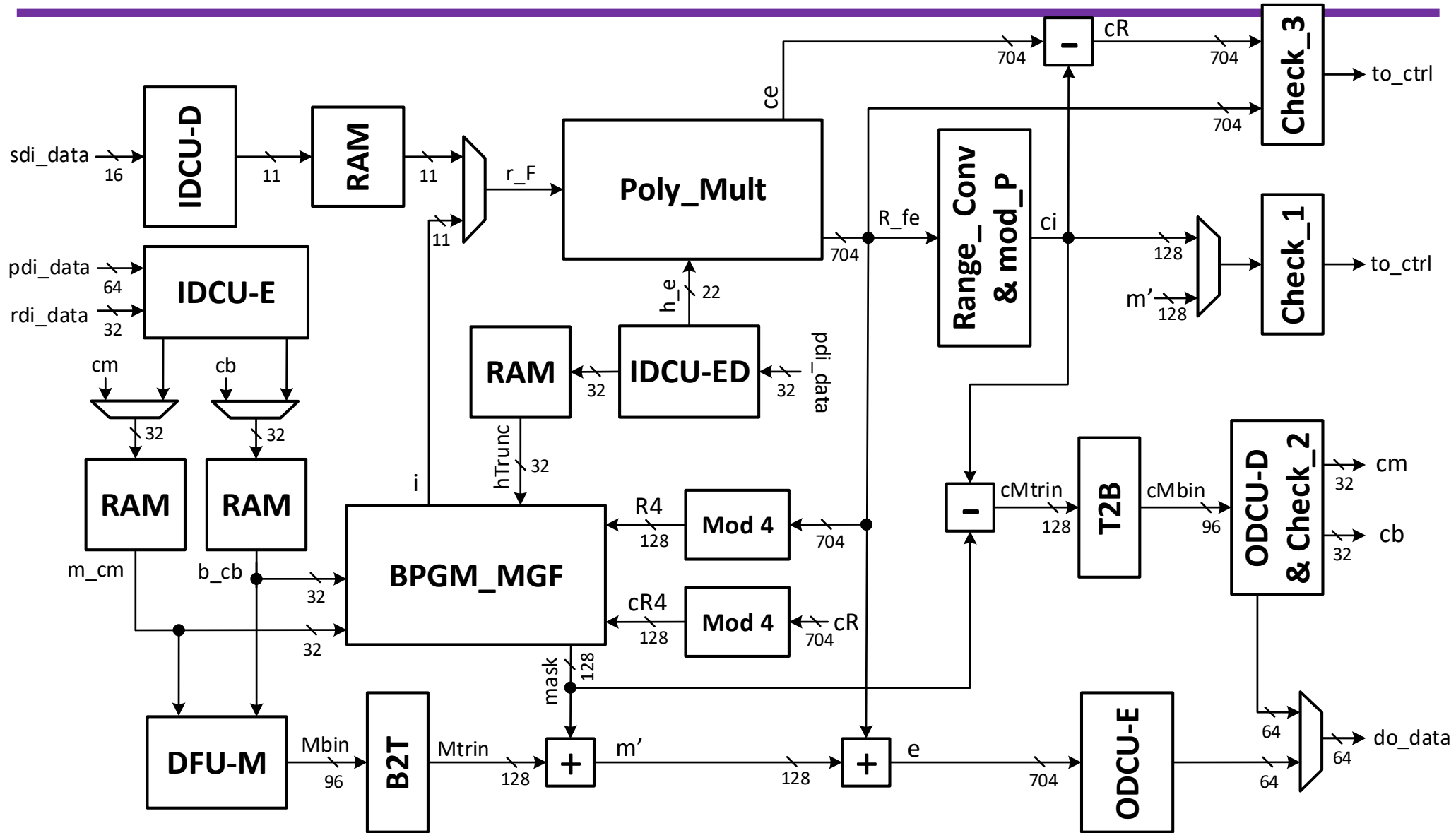
Speed-up vs. Software

Encryption Time in Hardware: 7.96 μ s

Encryption Time in Software: 674 μ s

Speed-up: x 85

Top-Level Block Diagram



Major Component Operations

Resource Utilization & Performance

Operation	LUTs: Slices	Clk Freq. [MHz]
Poly Mult	157,313 : 30,276	288.0
BPGM	2,611 : 553	354.0
MGF		
B2T	64 : 34	904.0
T2B	64 : 35	984.3
Poly Add	1338 : 272	316.3
Poly Subc	1221 : 258	331.2
Poly Sub	74 : 64	540.2

PolyMult contributes to **over 90% of area** and limits clock frequency

Comparison with Previously Reported Work

Challenging & Potentially Unfair due to:

1. Different variants of the algorithm
2. Different parameter sets
3. Different FPGA families
4. Different optimization targets

Comparison with the PQC McEliece Encryption

Classic McEliece

Published 40 years ago. No progress in cryptanalysis.

$m=13$, $t=119$, $N=6960$, optimization for speed,
the same security level of 256-bits
targeting Xilinx Virtex UltraScale

Designers: Yale University & Fraunhofer Institute for SIT, Germany

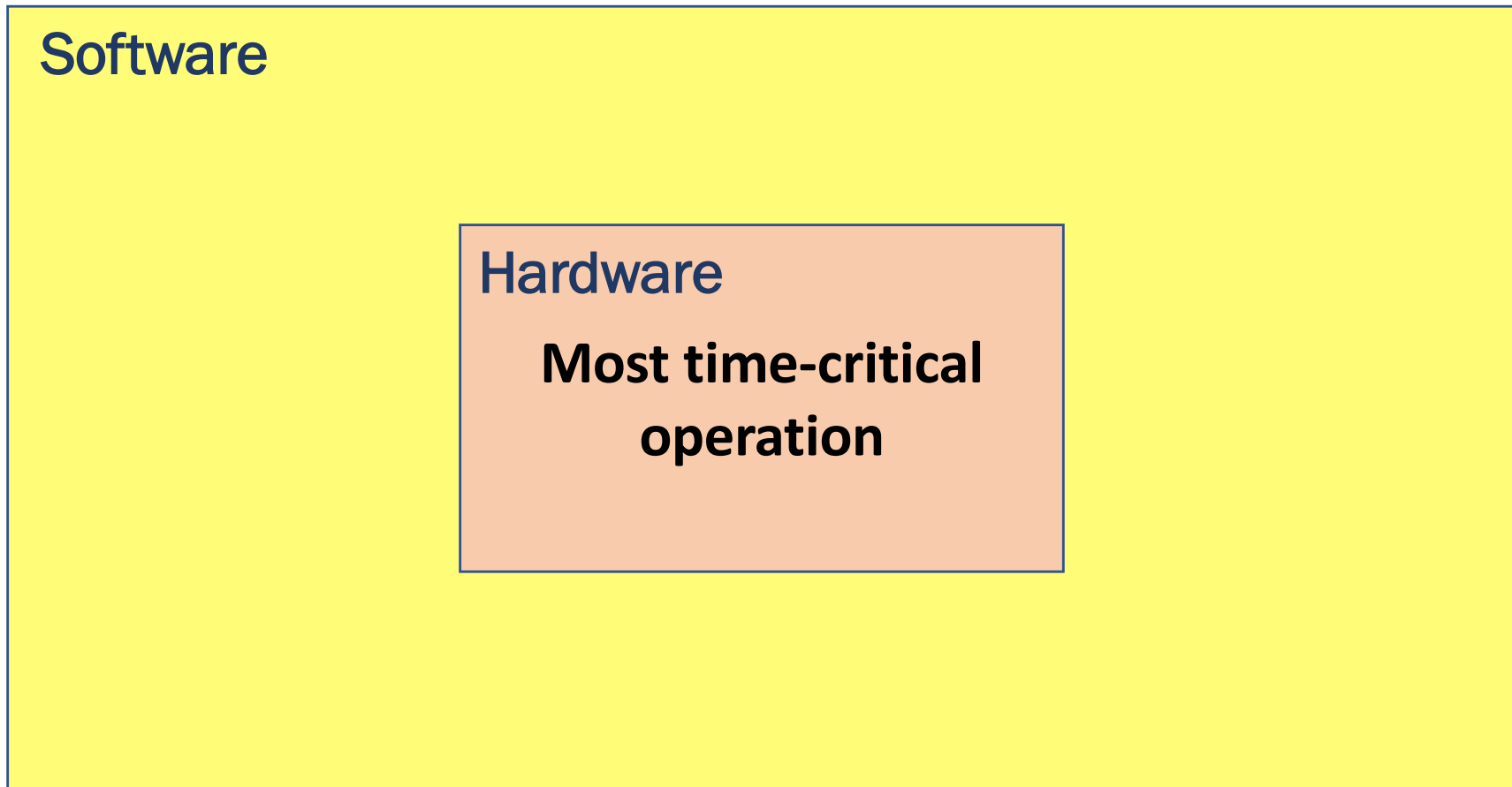
PQC Scheme	#Slices	#BRAMs	Freq. [MHz]	Enc [cycles]	Dec [cycles]	Enc [μ s]	Dec [μ s]
McEliece	8,236	35	426	5,413	17,055	12.70	40.04
NTRU	31,492	2	288	2,292	2,801	7.96	9.73
Ratio	3.82	0.06	0.68	0.42	0.16	0.63	0.24

NIST Standardization Process - Round 1

69 candidates accepted as complete, 5 since withdrawn
26 Countries, 260 co-authors

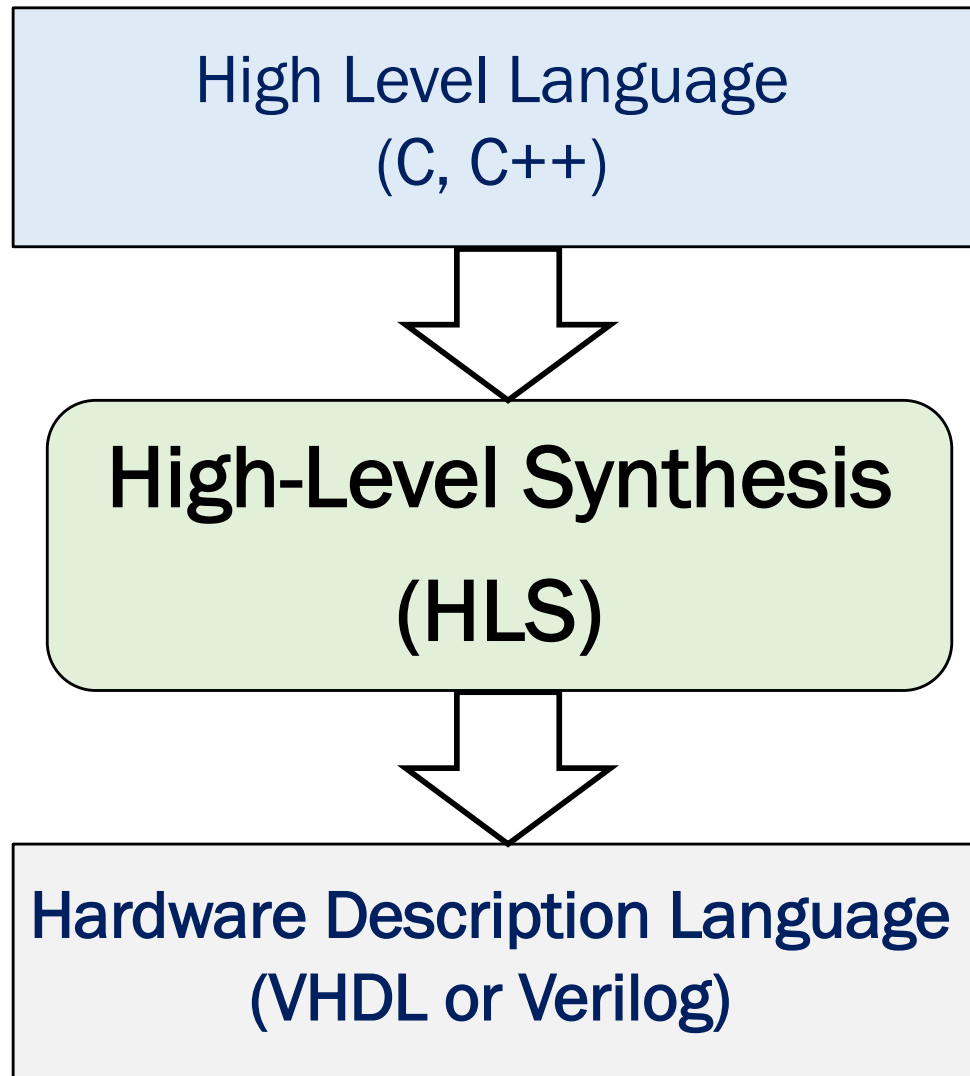
Family	Signature	Encryption/KEM	Overall
Lattice-based	5	22	27
Code-based	2	16	18
Multivariate	7	2	9
Hash-based	2		2
Isogeny-based		1	1
Other	3	4	7
Total	19	45	64

Approach 1: Software/Hardware Implementations



- Short development time
- Large improvement in execution time (up to ~150x)

Approach 2: High-Level Synthesis (HLS)



High-Level Synthesis – Initial Experiments

- Applied to Polynomial Multiplication Only
- Attempt to synthesize existing software implementation using Vivado HLS led to thousands of clock cycles per multiplication (vs. ~160 in RTL) in spite of introducing multiple Vivado HLS directives
- When HLS-ready C code written from scratch

HLS/RTL ratios were as follows:

#clock cycles: 1.00

frequency: 0.60

#LUTs: 2.15

#Slices: 2.30

More Investigation
Required

PQC Opportunities & Challenges

- The biggest **revolution in cryptography**, since the invention of public-key cryptography in 1970s
- Efficient **hardware implementations desperately needed** to prove the candidates suitability for high-performance applications and constrained environments.

Collaboration sought by submission teams!

- Likely **extensions to Instruction Set Architectures** of multiple major microprocessors
- **Start-up & new-product opportunities**
- **Once in the lifetime opportunity! Get involved!**

Q&A

Thank You!

Questions?



Comments?

Suggestions?

CERAG: <http://cryptography.gmu.edu>

ATHENa: <http://cryptography.gmu.edu/athena>