Post-Quantum Cryptography:

From the Point of View of Hardware Security

Debdeep Mukhopadhyay
Professor
Secured Embedded Architecture Laboratory (SEAL)

Department of Computer Science and Engineering
IIT Kharagpur
debdeep@cse.iitkgp.ac.in
debdeep.mukhopadhyay@gmail.com


SECURED
EMBEDDED
ARCHITECTURE
LABORATORY
VERTICAL DISCRETE ALGORITHM HORIZONTAL TIMING
SECURITY COMPUTATION
KHUDRA OPTIMIZATION
MICKEY ATTACK MICRO GALOIS TVLA PARALLEL
FIBONACCI PUF DEGREE ERROR TEMPLATE WATERMARKING
BLOCK PRIME HASH GROUP PREDICTOR SBOX POWER ECC RSA
COMMUNICATION FIELD SWARM ENTROPY SUPPORT
SYSTEM LFSR CIPHER FAULT GRAIN CACHE MULTICORE
COMPUTING THEORY
CORRECTION BRANCH CRYPTANALYSIS FUNCTION
STREAM
XOR
VECTOR COUNTERMEASURE
BYTE DECRYPTION MACHINE LOGARITHM
TROJAN PERMUTATION CODING RING
EMBEDDED AES CORRELATION SIDE-CHANNEL ALGEBRAIC
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

Family

Our Alumni PhD Students


Dr Chester Rebeiro, IITM


Dr Santosh Ghosh, Intel Labs, USA


Dr Bodhisatwa Mazumdar, IIT Indore


Dr Subidh Ali, IIT Bhilai


Dr Anju Johnson, University of Huddersfield


Dr Durga Prasad. Sahoo, Synopsys


Dr Sikhar Patranabis, IBM RL


Dr Sarani Bhattacharya, Imec Belgium, IIT KGP


Dr Sayandeep Saha, NTU SN, UCL, IIT B


Dr Rajat Sadhukhan, NYU IIT Roorkee


Dr Debapriya B Roy, TUM, IIT Kanpur


Dr Urbi Chatterjee, IIT Kanpur


Dr Manaar Alam, NYU, USA/AD


Dr Harishma Boyapally, NTU Singapore.


Dr Arnab Bag, Imec Belgium


Dr Durba Chatterjee, Radboud Univ, Netherlands.

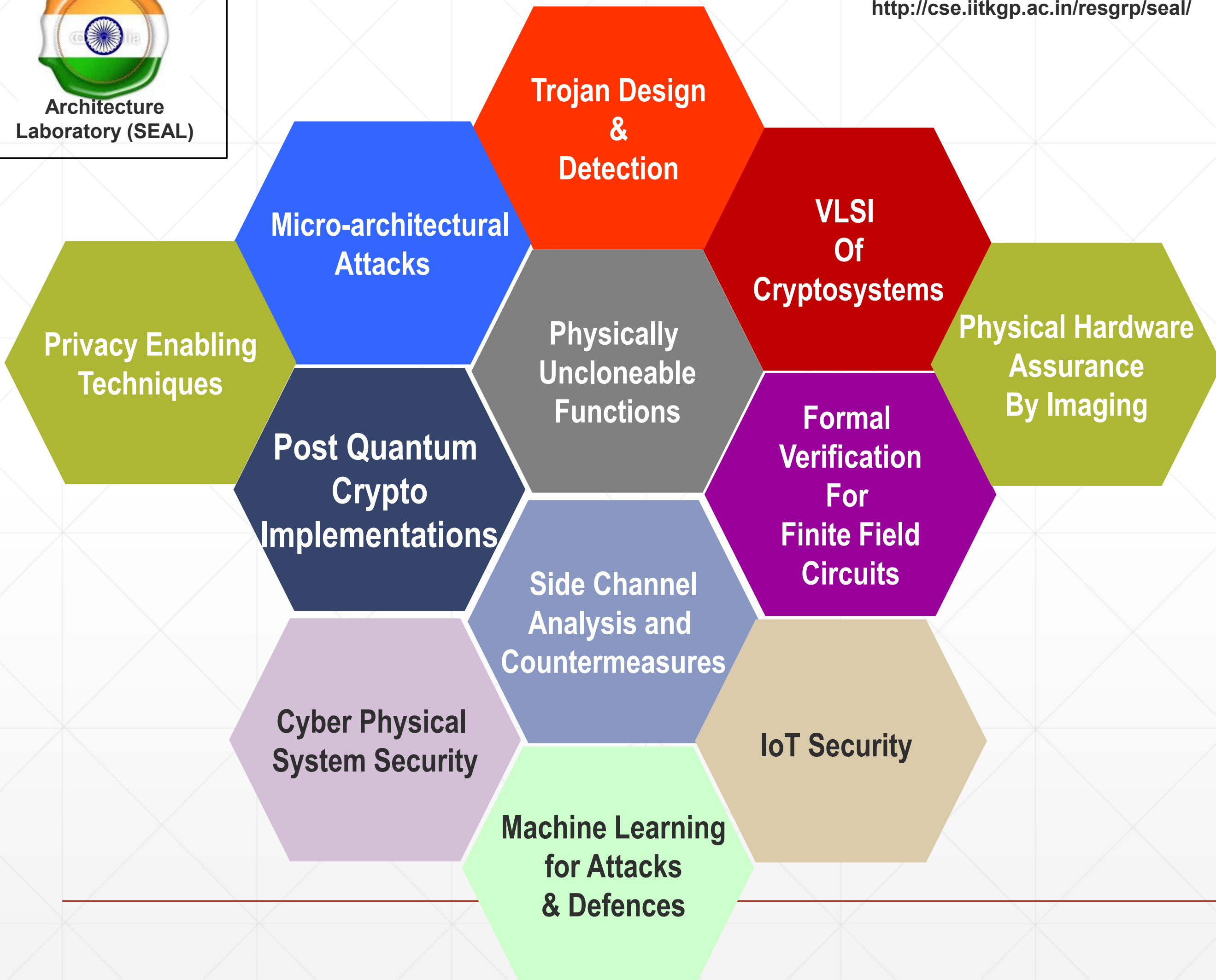

Dr Akashdeep Saha, NYU USA/AD.


Dr Anirban Chakraborty, MPI/Ruhr Univ, Germany.

Secured Embedded Architecture Laboratory (SEAL)

Research Focus of SEAL, IIT Kharagpur

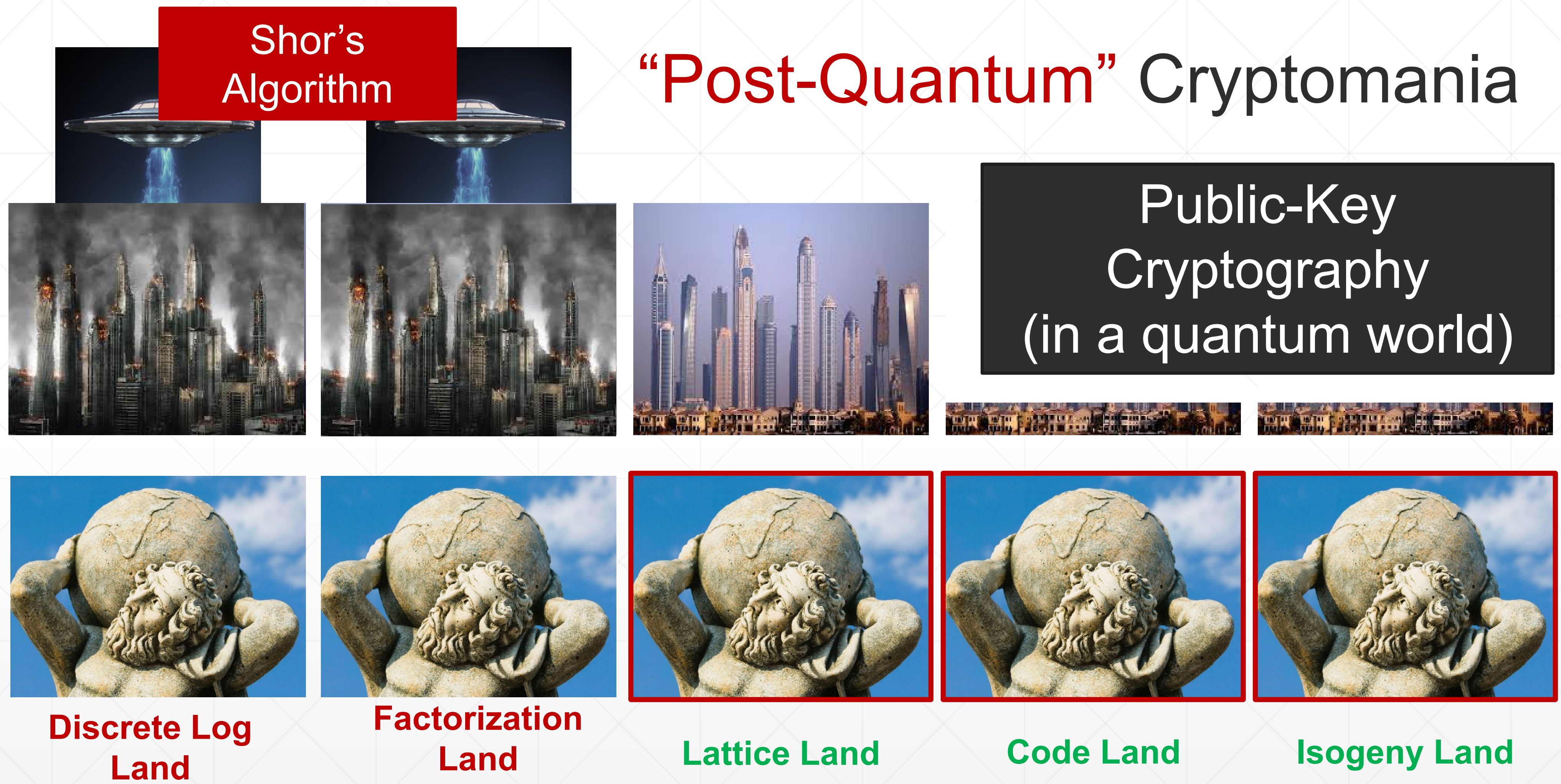
http://cse.iitkgp.ac.in/resgrp/seal/



- 16 PhD Graduates
- 4 Books (Mentioned in the global bestseller list for 2015)
- First state-of-the-art lab on Hardware Security in India, and among the few in Asia.
- Funding of Rs 50 Crores
- Invited Talks: Dagstuhl Seminars, Shonan Seminar, Intel, TI, Canon, IBM-Research, Bosch, NTT Labs, K U Leuven, Shanghai JiaTong Univ, Stanford, Columbia, NYU (USA, Shanghai, UAE), TU Darmstadt, TelecomParis, NTU Singapore, SUTD.
- Shanti Swarup Bhatnagar Award 2021 for Science & Technology, Swarnajayanti Fellowship (15-16), DSCI Award from Data Security Council of India (2018)

A Quick Look into SEAL, IIT Kharagpur

<u>Best Lab Demo at</u> IEEE International Conference on PHYSICAL ASSURANCE and INSPECTION of ELECTRONICS (PAINE)
https://paine-conference.org/paine-2023-winners/


"Post-Quantum" Cryptomania
Shor's Algorithm
Public-Key Cryptography (in a quantum world)
Discrete Log Land
Factorization Land
Lattice Land
Code Land
Isogeny Land

How is the (classical) cryptographic landscape shaped?

- **What we want from cryptography :** Informally, a user would like to *protect* confidential data from being recovered by an adversary.

How is the (classical) cryptographic landscape shaped?

- **What we want from cryptography :** Informally, a user would like to *protect* confidential data from being recovered by an adversary.

- **How to do that :** Find *hard* to solve **mathematical problems** (like computing the logarithm of modular exponentiation) and build a cryptosystem about it.

How is the (classical) cryptographic landscape shaped?

- **What we want from cryptography :** Informally, a user would like to *protect* confidential data from being recovered by an adversary.

- **How to do that :** Find *hard* to solve **mathematical problems** (like computing the logarithm of modular exponentiation) and build a cryptosystem about it.

- **A note on "assumption":** Note that such *hard* mathematical problems are *assumed/believed* to be intractable by *efficient* adversaries (say a poly-bounded adversary in time and memory)
    - **Caveat**: This is only an assumption, not a guarantee. Implying, if a hard problem is tractable by an adversary in future, *all* cryptosystems based upon it are vulnerable

How is the (classical) cryptographic landscape shaped?

- **What we want from cryptography :** Informally, a user would like to *protect* confidential data from being recovered by an adversary.

- **How to do that :** Find *hard* to solve **mathematical problems** (like computing the logarithm of modular exponentiation) and build a cryptosystem about it.

- **A note on "assumption":** Note that such *hard* mathematical problems are *assumed/believed* to be intractable by *efficient* adversaries (say a poly-bounded adversary in time and memory)
    - **Caveat**: This is only an assumption, not a guarantee. Implying, if a hard problem is tractable by an adversary in future, *all* cryptosystems based upon it are vulnerable

- Quantum Computing can solve classical hard problems efficiently!

Stepping into the Post-Quantum world

- **What we want from (post-quantum)cryptography :** Informally, a user would like to *protect* confidential data from being recovered by both classical and quantum adversary.

Stepping into the Post-Quantum world

- **What we want from (post-quantum)cryptography :** Informally, a user would like to *protect* confidential data from being recovered by both classical and quantum adversary.

- Formally, find *hard mathematical assumptions* that are secure under two models:

  - Random Oracle Model
    - Used extensively to model a classical adversary
    - Gives the adversary the capability of querying a black-box hash function

Stepping into the Post-Quantum world

- **What we want from (post-quantum)cryptography :** Informally, a user would like to *protect* confidential data from being recovered by both classical and quantum adversary.

- Formally, find *hard mathematical assumptions* that are secure under two models:

  - Random Oracle Model
    - Used extensively to model a classical adversary
    - Gives the adversary the capability of querying a black-box hash function

  - (Quantum-accessible) Random Oracle Model
    - Models a quantum adversary
    - Gives the adversary the capability to query in *superposition,* a special property in quantum physics in which a particle (like a photon) can co-exist in multiple states *at the same time* (thus allowing parallel computation capability)

Roadmap of developing PQ Cryptosystems

- An overview on developing a PQ Cryptosystem
  - **Step 1**: Define new *hard* problems secure in ROM and QROM security models
  - **Step 2**: Build cryptosystems atop it, and *reduce* their security to that of the hard problem
    - **Sample reduction statement:** "If <X> hard problem is secure against a poly-bounded adversary in ROM and QROM security model, then my cryptosystem <C> is also secure against a poly-bounded classical and quantum adversary"

Roadmap of developing PQ Cryptosystems

- An overview on developing a PQ Cryptosystem
  - **Step 1**: Define new *hard* problems secure in ROM and QROM security models
  - **Step 2**: Build cryptosystems atop it, and *reduce* their security to that of the hard problem
    - **Sample reduction statement:** "If <X> hard problem is secure against a poly-bounded adversary in ROM and QROM security model, then my cryptosystem <C> is also secure against a poly-bounded classical and quantum adversary"
  - **Step 3**: Set parameter levels for the cryptosystem, allowing for implementation optimizations at the software level

Roadmap of developing PQ Cryptosystems

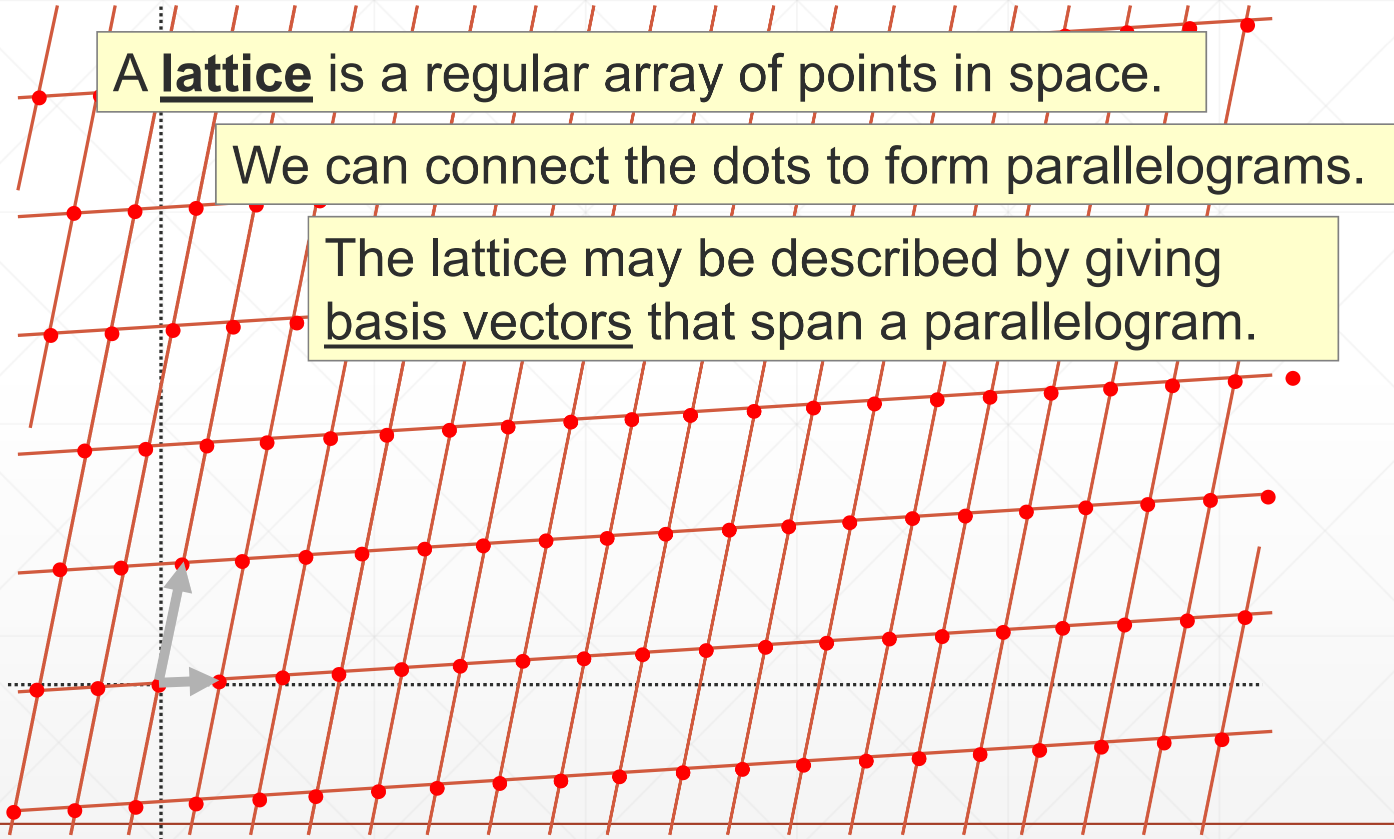
- An overview on developing a PQ Cryptosystem
  - **Step 1**: Define new *hard* problems secure in ROM and QROM security models
  - **Step 2**: Build cryptosystems atop it, and *reduce* their security to that of the hard problem
    - **Sample reduction statement:** "If <X> hard problem is secure against a poly-bounded adversary in ROM and QROM security model, then my cryptosystem <C> is also secure against a poly-bounded classical and quantum adversary"
  - **Step 3**: Set parameter levels for the cryptosystem, allowing for implementation optimizations at the software level
  - **Step 4:** Look for hardware/software co-design (or complete hardware) acceleration
  - **Step 5**: Look and secure the cryptosystem against side-channels

Step 1: Define new hard problems secure in ROM and QROM security models

PQ resistant hard problems

- Several types:
  - Code-based : based on linear codes
  - Multivariate-based : based on multivariate polynomials
  - Lattice-based : based on lattices
- All these problems, like their classical counterparts, are *assumptions.* That is, they are believed, so far, to be intractable against poly-bounded adversaries in time and memory.

What is a Lattice?



A **lattice** is a regular array of points in space.

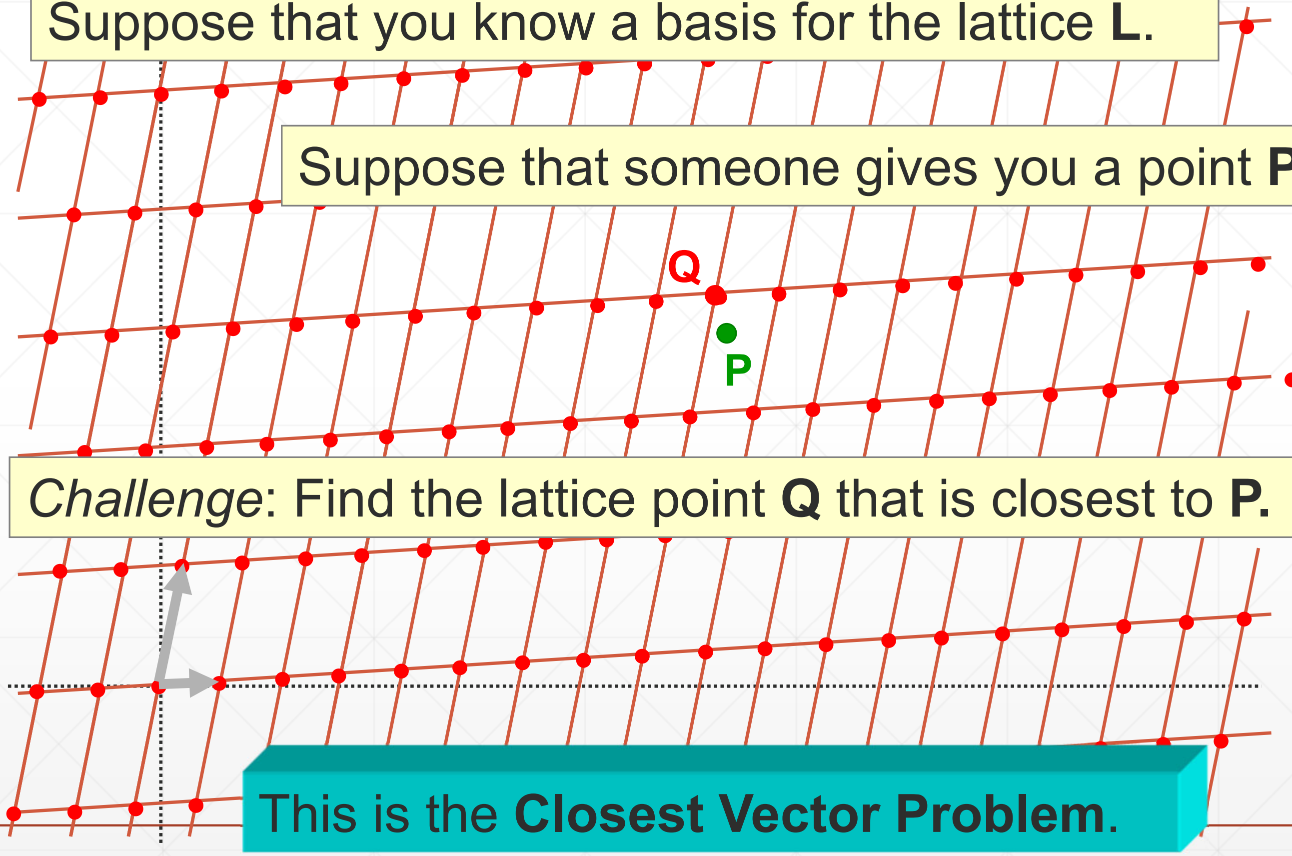
We can connect the dots to form parallelograms.

The lattice may be described by giving basis vectors that span a parallelogram.

What is the Closest Vector Problem?

Suppose that you know a basis for the lattice **L**.

Suppose that someone gives you a point **P**.



Challenge: Find the lattice point **Q** that is closest to **P.**

This is the **Closest Vector Problem**.

Noise and Hardness

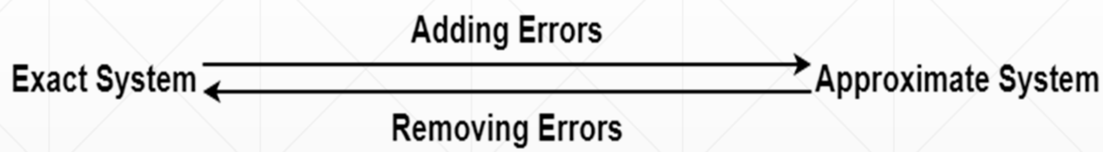
- Noise has been found to convert “easy” problems to more hard instances.
- Let’s start with a simple instance of greatest-common-divisor (GCD).
- Let’s say that one chooses a secret integer s, then samples several random integers q_i’s
- Define multiples of s, by $p_i = sq_i, 1 \leq i \leq l$
- It is easy to compute $s, s = \gcd(q_1, \cdots, q_l)$.
- But what if we are given “approximate multiples” of s instead of exact multiples, that is, if one adds small integers r_i to the p_i’s we have:

$$b_i = sq_i + r_i, 1 \leq i \leq l$$

- How to obtain s? This problem, called as the **Approximate Common Divisor Problem** is hard for properly chosen parameters.

Learning With Errors[1]

$$
\begin{aligned}
a_{11} * s_{11} + a_{12} * s_{12} + \cdots + a_{1n} * s_{1n} &= b_1 \pm e_1 \dashrightarrow a_{11} * s_{11} + a_{12} * s_{12} + \cdots + a_{1n} * s_{1n} \approx b_1 \\
a_{21} * s_{21} + a_{22} * s_{22} + \cdots + a_{2n} * s_{2n} &= b_2 \pm e_2 \dashrightarrow a_{21} * s_{21} + a_{22} * s_{22} + \cdots + a_{2n} * s_{2n} \approx b_2 \\
&\vdots \\
a_{m1} * s_{m1} + a_{m2} * s_{m2} + \cdots + a_{mn} * s_{mn} &= b_m \pm e_m \dashrightarrow a_{m1} * s_{m1} + a_{m2} * s_{m2} + \cdots + a_{mn} * s_{mn} \approx b_m
\end{aligned}
$$

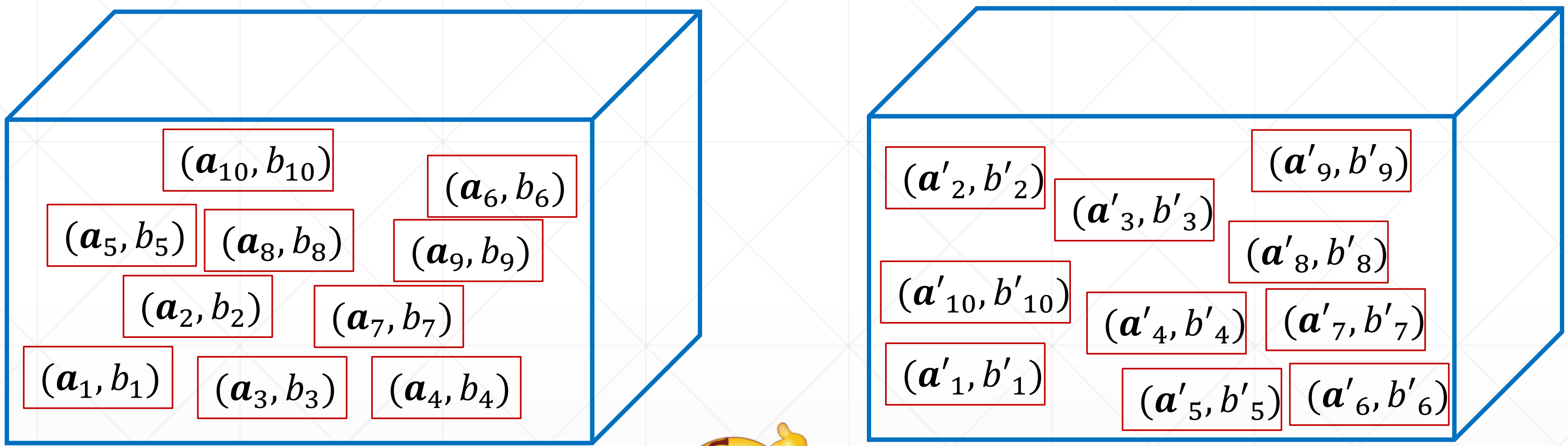


1. O. Regev, “On lattices, learning with errors, random linear codes, and cryptography”, 2005.

An Example

- Suppose that $s = (3,7)$, and $e_1 = e_2 = -1$
- $5s_1 + 3s_2 \approx 35$
- $4s_1 + 2s_2 \approx 27$
- Performing standard row-reduce, we obtain $s = (\frac{11}{2}, \frac{5}{2})$
  - *Rounding this works to* $s = (6,3)$*, which is far off from the actual result.*

The Learning With Errors (LWE) Problem



$\boldsymbol{s} \in \mathbb{Z}_q^n, \boldsymbol{a}_i \stackrel{\$}{\leftarrow} \mathbb{Z}_q^n, e \leftarrow \mathcal{G}, b_i = \boldsymbol{a}_i \cdot \boldsymbol{s} + e$

$\boldsymbol{a}'_i \stackrel{\$}{\leftarrow} \mathbb{Z}_q^n, b'_i \stackrel{\$}{\leftarrow} \mathbb{Z}_q$

Problem: Distinguish the box with LWE samples from the box with uniform random samples efficiently (in polynomial time).

Find s by observing the inputs and outputs – Search-LWE. While the problem of distinguishing is called Decisional-LWE problem.

PQ resistant hard problems : What to choose?

- The choice of the hard problem depends upon the requirements of the cryptosystem. Examples:
  - **Hardness level**: Whether worst-case or average-case hardness is needed
    - **Worst-case** : There exists *at least one* instance of the problem that is difficult to solve. Cryptosystems depending on **worst-case hardness** should use such instances only
    - **Average-case:** The problem is hard even on random samples of problem instances. Cryptosystems depending on **average-case hardness** can be more lenient on their problem samples.
    - Lattice-based problems have provably worst-case to average-case reductions.

PQ resistant hard problems : What to choose?

- The choice of the hard problem depends upon the requirements of the cryptosystem. Examples:
  - **Hardness level**: Whether worst-case or average-case hardness is needed
  - **Optimization opportunities** : Whether the structure of the problem has opportunities for optimizations
    - **Lattice-based schemes can exploit the structure of algebraic Rings to have reduced storage and faster runtimes through NTT (Number Theoretic Transform).**
    - Code-based schemes work on Binary Fields only.
    - Multivariate-based schemes can benefit from optimizations in related literature on improving polynomial evaluations.

PQ resistant hard problems : What to choose?

- The choice of the hard problem depends upon the requirements of the cryptosystem. Examples:
    - **Hardness level**: Whether worst-case or average-case hardness is needed
    - **Optimization opportunities** : Whether the structure of the problem has opportunities for optimizations
    - **Parameters:** Size of key, ciphertext etc. in the cryptosystem built upon the problem.

Step 2: Build cryptosystems atop the hard problems

NIST Standardization (and other research)

- Process of standardizing cryptosystems built upon these problems

- Two types:
  - **Key Encapsulation Mechanisms** : A PQ cryptosystem to establish (usually symmetric) session keys between two parties

  - **Digital Signature schemes** : PQ cryptosystems to establish authenticity of messages by signing them with the identity of initial message holder.

- Other cryptosystems (outside the scope of standardization)
  - **Privacy Enabling Technologies :** post-quantum Fully Homomorphic Encryption, Multi-Party Computation, Searchable Symmetric Encryption etc.
  - **Encryption schemes :** generic encryption schemes, identity based schemes, attribute-based encryption schemes
  - Many others....

NIST Standardization (and other research)

- Process of standardizing cryptosystems built upon these problems

- Two types:
  - **Key Encapsulation Mechanisms** : A PQ cryptosystem to establish (usually symmetric) session keys between two parties

    - BIKE : Code-based KEM
    - Classic McEliece : Code-based KEM
    - HQC : Code-based KEM
    - SIKE : Isogeny based
    - Kyber : Lattice based (**chosen for standardization)**

  - **Digital Signature schemes** : PQ cryptosystems to establish authenticity of messages by signing them with the identity of initial message holder.

NIST Standardization (and other research)

- Process of standardizing cryptosystems built upon these problems

- Two types:
  - **Key Encapsulation Mechanisms** : A PQ cryptosystem to establish (usually symmetric) session keys between two parties

  - **Digital Signature schemes** : PQ cryptosystems to establish authenticity of messages by signing them with the identity of initial message holder.

    - CRYSTALS-Dilithium : Lattice based (**chosen for standardization)**
    - Falcon : Lattice based (**chosen for standardization)**
    - SPHINCS+ : Hash based (**chosen for standardization)**

Step 3: Choose parameter levels

Parameter Levels

- An extremely important choice

Parameter Levels

- An extremely important choice
- Tradeoff between security and efficiency
  - Small / Conservative parameters
    - Efficient
    - (Almost always) insecure against poly-bounded adversary in time and memory
  - Large parameters
    - Inefficient
    - Secure against poly-bounded adversary in time and memory

Parameter Levels

- An extremely important choice
- Tradeoff between security and efficiency
  - Small / Conservative parameters
    - Efficient
    - (Almost always) insecure against poly-bounded adversary in time and memory
  - Large parameters
    - Inefficient
    - Secure against poly-bounded adversary in time and memory
  - <u>Strategy 1</u>: Choose large params and throw ever-improving computing power and parallelization at it

Parameter Levels

- An extremely important choice

- Tradeoff between security and efficiency
  - Small / Conservative parameters
    - Efficient
    - (Almost always) insecure against poly-bounded adversary in time and memory
  - Large parameters
    - Inefficient
    - Secure against poly-bounded adversary in time and memory
  - Strategy 1: Choose large params and throw ever-improving computing power and parallelization at it
    - What about resource-constrained devices?

Parameter Levels

- An extremely important choice
- Tradeoff between security and efficiency
  - Small / Conservative parameters
    - Efficient
    - (Almost always) insecure against poly-bounded adversary in time and memory
  - Large parameters
    - Inefficient
    - Secure against poly-bounded adversary in time and memory
  - Strategy 2: Carefully compute (wherever possible) the bit security of the cryptosystem and offer multiple levels of security. Users are free to choose
    - Allows users to tradeoff efficiency/security based on available compute power

Step 4: Hardware Support for Post-Quantum Protocols

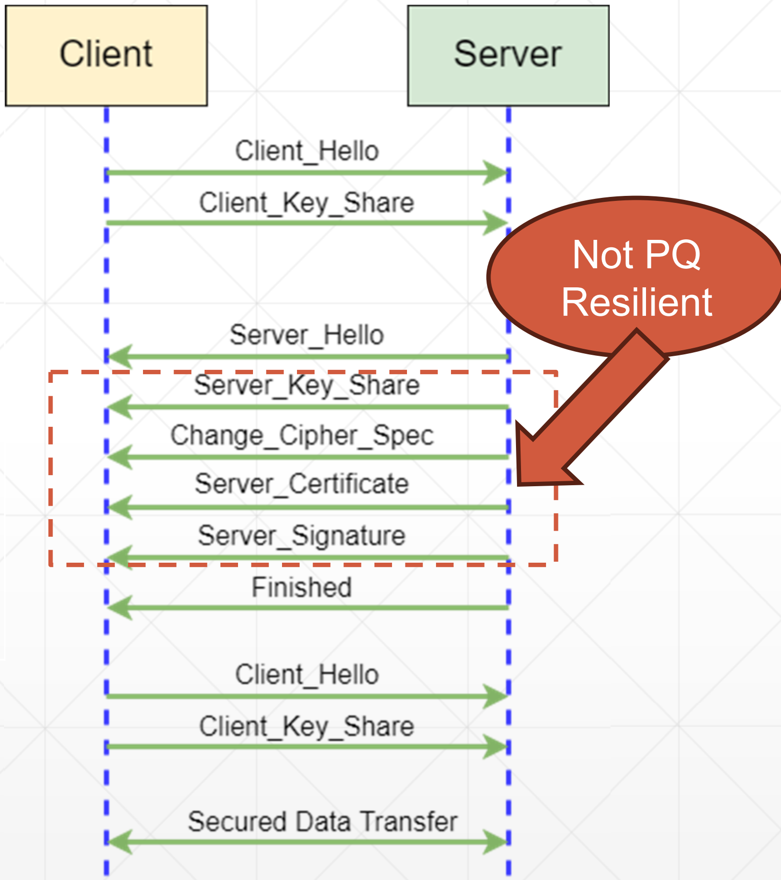
TLS basic Architecture

- TLS is the current standard protocol for establishing secure communication on the Internet.

- TLS consists of three basic steps: Connection establishment, **TLS handshake** and the encryption of application data using symmetric cryptography

- In the figure, we have shown a overview of TLS 1.3.

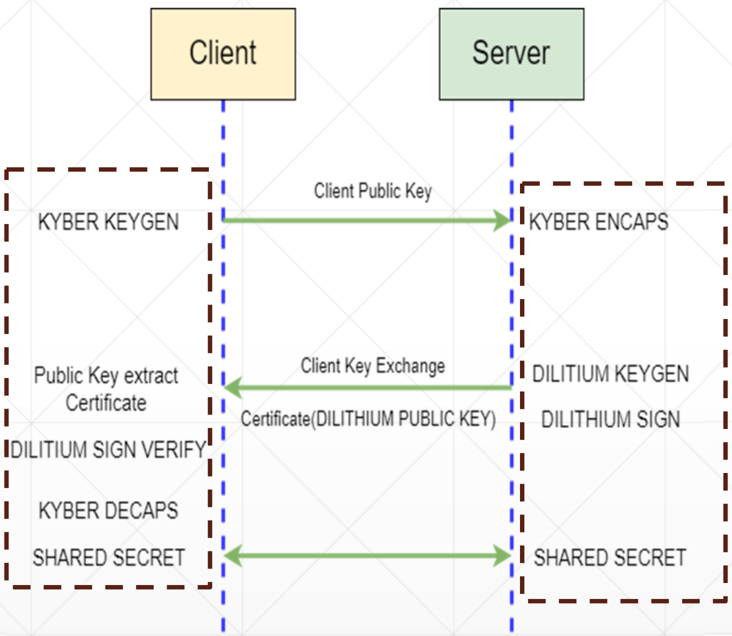
    - In the first step, the client contacts the server with the Client_Hello message consisting of specific parameter
    - To reduce network traffic, the client also sends its key material (Client_Key_Share) for the key establishment
    - In the second step, the server replies with the Server_Hello that is similar to the Client_Hello
    - The server reply is signed by its private key.
    - In the last step, the client also transmits a confirmation for encryption of subsequent messages (Change_Cipher_Spec) and its readiness to communicate securely (Finished).



Overview of TLS Handshake

PQ-TLS: Making TLS PQ Secure

- TLS is the current standard protocol for establishing secure communication on the Internet.
- TLS consists of three basic steps: Connection establishment, TLS handshake and the encryption of application data using symmetric cryptography
- In the figure, we have introduced our version of TLS 1.3



Choosing the PQC algorithms: Kyber and Dilithium

- **In order to make the public key infrastructure quantum-safe, the pre-quantum schemes in protocols such as TLS are needed to be replaced**
- We choose **Kyber** for the key encapsulation mechanism (KEM) and **Dilithium** for the digital signature generation which are the most important components of TLS
- Both Dilithium and Kyber has a similar mathematical background and has a similar structure of NTT multiplier and Keccak core.
- **KGP-PQC-TLS:** An **agile** Post-Quantum TLS accelerator which encompasses all the security levels of Kyber and Dilithium
    - From an application perspective, a unified design has helped us in implementing post-quantum version of TLS-1.3 protocol.

Siddhartha Chowdhury, A Minimalistic Perspective on Hardware Designs for Modern-day Public-Key Cryptosystems (MS Thesis)

Hardware Acceleration : A Case Study of PQ-TLS

- Case Study of an hardware accelerated implementation of a Post Quantum TLS accelerator for resource constrained devices. Developed by **Secured Embedded Architecture Laboratory, IIT Kharagpur.**

Hardware Acceleration : A Case Study of PQ-TLS

- Case Study of an hardware accelerated implementation of a Post Quantum TLS accelerator for resource constrained devices. Developed by **Secured Embedded Architecture Laboratory, IIT Kharagpur.**

- General considerations of designing hardware acceleration units for PQ cryptosystem
  - Must not affect the param choices (and transitively the security!)

Hardware Acceleration : A Case Study of PQ-TLS

- Case Study of an hardware accelerated implementation of a Post Quantum TLS accelerator for resource constrained devices. Developed by **Secured Embedded Architecture Laboratory, IIT Kharagpur.**

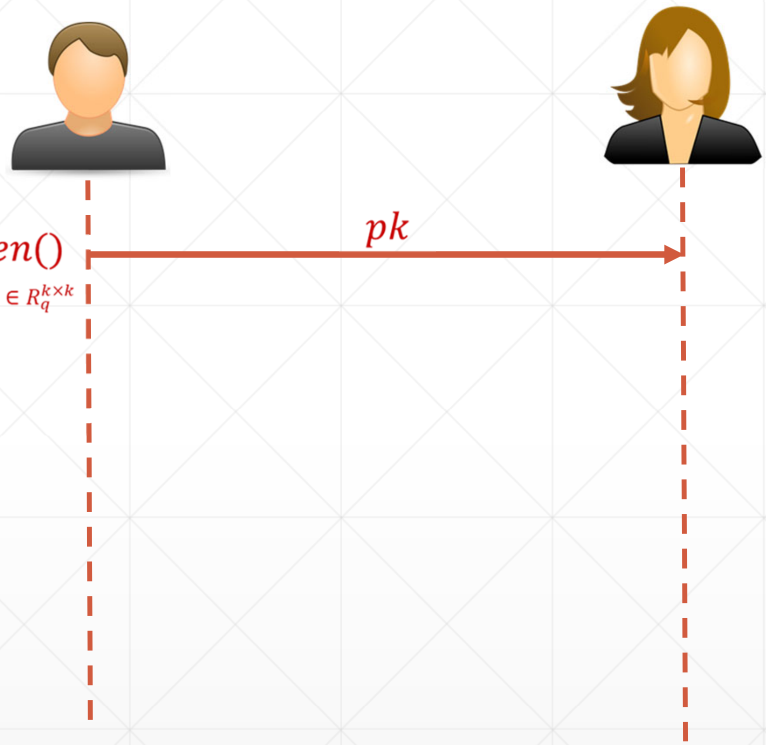
- General considerations of designing hardware acceleration units for PQ cryptosystem
  - Must not affect the param choices (and transitively the security!)
  - Should not optimize any security critical operations
    - For instance, *repetitive computations* might seem an avenue for optimization, but will affect CCA-2 security (security against chosen ciphertext attacks)

Hardware Acceleration : A Case Study of PQ-TLS

- Case Study of an hardware accelerated implementation of a Post Quantum TLS accelerator for resource constrained devices. Developed by **Secured Embedded Architecture Laboratory, IIT Kharagpur.**

- General considerations of designing hardware acceleration units for PQ cryptosystem
  - Must not affect the param choices (and transitively the security!)
  - Should not optimize any security critical operations
    - For instance, *repetitive computations* might seem an avenue for optimization, but will affect CCA-2 security (security against chosen ciphertext attacks)
  - **Rule of thumb**: Do not optimize the algorithm. Only optimize the implementation
    - Example: Build a hardware core that does matrix-vector multiplications (**for the same param set)** faster than software

Kyber in a nutshell



$(pk, sk) = Keygen()$

Generate a matrix A, where $A \in R_q^{k \times k}$

Generate $s \in R_q^k$ and $e \in R_q^k$

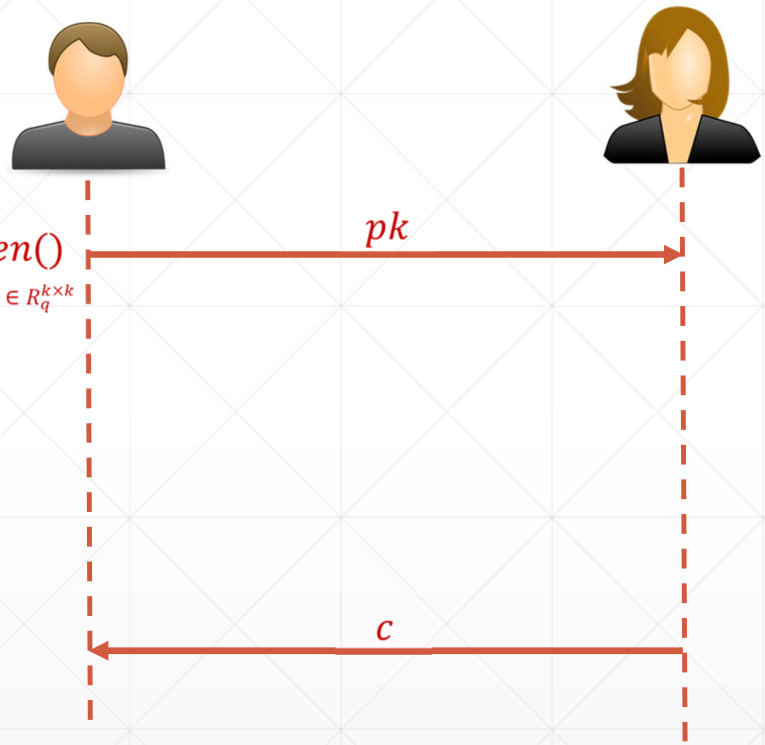
$$t = A \cdot s + e$$

$$\boldsymbol{pk = Encode(t)}$$

$$sk = Encode(s)$$

XOF: SHAKE-128; H: SHA3-256; G: SHA3-512; KDF: SHAKE-256.

Kyber in a nutshell



$(pk, sk) = Keygen()$

Generate a matrix A, where $A \in R_q^{k \times k}$

Generate $s \in R_q^k$ and $e \in R_q^k$

$t = A \cdot s + e$

$\boldsymbol{pk = Encode(t)}$

$sk = Encode(s)$

$(c, K) = Encaps(pk)$

$(\hat{K}, r) = G(m \parallel H(pk)), where\ r \in R_q^k$

$m \in B^{32}, m = H(m)$

$Enc(r, m, pk) : Input \rightarrow (r, m, pk)\ , Output \rightarrow c$

Generate a matrix A, where $A \in R_q^{k \times k}$

$t = Decode(pk)$

$e_1 \in R_q^k$ and $e_1 \in R_q$

$u = A^T \cdot r + e_1$

$v = t^T \cdot r + e_2 + Decompress_q(m, 1)$

$c_1 = Encode(Compress_q(u))$

$c_2 = Encode(Compress_q(v))$

$\boldsymbol{c = (c_1 \parallel c_2)}$

$K = KDF(\hat{K} \parallel H(c))$

XOF: SHAKE-128; H: SHA3-256; G: SHA3-512; KDF: SHAKE-256.

Kyber in a nutshell

$$(pk, sk) = Keygen()$$

Generate a matrix A, where $A \in R_q^{k \times k}$
Generate $s \in R_q^k$ and $e \in R_q^k$

$$t = A \cdot s + e$$
$$\boldsymbol{pk = Encode(t)}$$
$$sk = Encode(s)$$

pk →

$$(c, K) = Encaps(pk)$$

$$(\hat{K}, r) = G(m \parallel H(pk)), where\ r \in R_q^k$$
$$m \in B^{32}, m = H(m)$$

$Enc(r, m, pk) : Input \rightarrow (r, m, pk)\ ,\ Output \rightarrow c$
Generate a matrix A, where $A \in R_q^{k \times k}$

$$t = Decode(pk)$$
$e_1 \in R_q^k$ and $e_1 \in R_q$
$$u = A^T \cdot r + e_1$$
$$v = t^T \cdot r + e_2 + Decompress_q(m, 1)$$
$$c_1 = Encode(Compress_q(u))$$
$$c_2 = Encode(Compress_q(v))$$
$$\boldsymbol{c = (c_1 \parallel c_2)}$$

$$K = KDF(\hat{K} \parallel H(c))$$

← c

$$K = Decaps(sk, c)$$

$$u, v = Decode(Decompress_q(c))$$
$$\hat{s} = Decode(s)$$
$$h = v - \hat{s}^T \cdot u$$
$$m' = Encode(Compress_q(h, 1))$$
$$(K', r') = G(m')$$
$$c' = Enc(pk, m', r')$$
$$if\,(c == c') \rightarrow \boldsymbol{K = KDF(K' \parallel c')}$$
$$else \rightarrow \boldsymbol{K = random}$$

H: SHA3-256; G: SHA3-512; KDF: SHAKE-256.

Parameters

	n	k	q	η_1	η_2	(d_u, d_v)	δ
Kyber512	256	2	3329	3	2	$(10, 4)$	2^{-139}
Kyber768	256	3	3329	2	2	$(10, 4)$	2^{-164}
Kyber1024	256	4	3329	2	2	$(11, 5)$	2^{-174}

Dilithium in a nutshell

$$(pk, sk) = Keygen()$$

Generate a matrix A, where $A \in R_q^{k \times l}$

Generate $s_1 \in R_q^l$ and $s_2 \in R_q^k$

$$t = A \cdot s_1 + s_2$$

$$(t_1, t_0) = Power2Round_q(t)$$

$$\boldsymbol{pk = Encode(t_1)}$$

$$sk = Encode(t_0, s_1, s_2)$$

Dilithium in a nutshell

pk, M, σ

$(pk, sk) = Keygen()$

Generate a matrix A, where $A \in R_q^{k \times l}$

Generate $s_1 \in R_q^l$ and $s_2 \in R_q^k$

$t = A \cdot s_1 + s_2$

$(t_1, t_0) = Power2Round_q(t)$

$\boldsymbol{pk = Encode(t_1)}$

$sk = Encode(t_0, s_1, s_2)$

$\sigma = Sign(sk, M)$

Generate a matrix A, where $A \in R_q^{k \times l}$

Generate y $\in R_q^l$

$w = A \cdot y$, $w_1 = HighBits_q(w)$

$c = H(w_1)$, $z = y + c \cdot s_1$

$h = MakeHint_q()$

Return $\sigma(c, z, h)$

Dilithium in a nutshell

$$(pk, sk) = Keygen()$$

Generate a matrix A, where $A \in R_q^{k\times l}$
Generate $s_1 \in R_q^l$ and $s_2 \in R_q^k$

$$t = A \cdot s_1 + s_2$$
$$(t_1, t_0) = Power2Round_q(t)$$
$$\boldsymbol{pk = Encode(t_1)}$$
$$sk = Encode(t_0, s_1, s_2)$$

$$pk, M, \sigma$$

$$\sigma = Sign(sk, M)$$

Generate a matrix A, where $A \in R_q^{k\times l}$
Generate $y \in R_q^l$
$w = A \cdot y$, $w_1 = HighBits_q(w)$
$c = H(w_1)$, $z = y + c \cdot s_1$

$$h = MakeHint_q()$$

Return $\sigma(c, z, h)$

$$Verify(pk, M, \sigma) = valid/invalid$$

Generate a matrix A, where $A \in R_q^{k\times l}$

$$w_1 = UseHint_q(h)$$

Verify the correctness of the signature based on the value of z, c and h

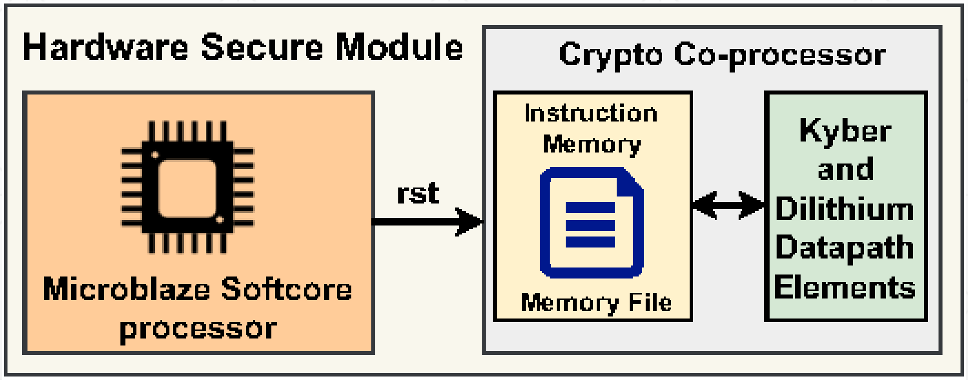
Parameters

NIST Security Level	2	3	5
Parameters			
q [modulus]	8380417	8380417	8380417
d [dropped bits from $\mathbf{t}$]	13	13	13
τ [# of ± 1's in c]	39	49	60
challenge entropy [$\log \binom{256}{\tau} + \tau$]	192	225	257
γ_1 [$\mathbf{y}$ coefficient range]	2^{17}	2^{19}	2^{19}
γ_2 [low-order rounding range]	$(q-1)/88$	$(q-1)/32$	$(q-1)/32$
(k, ℓ) [dimensions of $\mathbf{A}$]	$(4, 4)$	$(6, 5)$	$(8, 7)$
η [secret key range]	2	4	2
β [$\tau \cdot \eta$]	78	196	120
ω [max. # of 1's in the hint $\mathbf{h}$]	80	55	75
Repetitions (from Eq. (5))	4.25	5.1	3.85

Overall Architecture of our Proposed KGP-PQC-TLS:

- We have chosen a lightweight Xilinx board NEXYS 4 DDR which houses an Artix-7 FPGA and a soft-core microprocessor

- The Microblaze processor triggers the respective key generation, key encapsulation/decapsulation, signature generation and verification operations whenever required



Overall Architecture of our Proposed Design

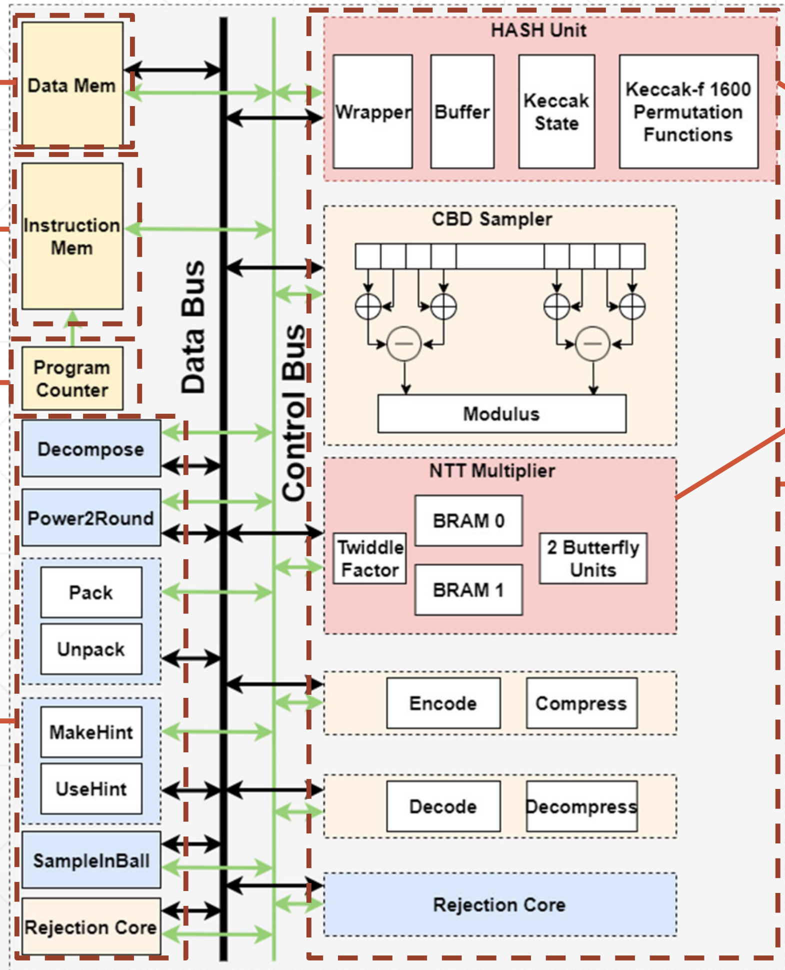
Hardware Acceleration : A Case Study of Post Quantum TLS

Data Memory

Instruction Memory

Program Counter

Datapath Elements



Overview of the Hardware Architecture of the Design

Common Elements for the Unified Design

Datapath Elements

The multiple outputs produced by these modules are fed to a multiplexer unit connected to each individual module, the appropriate value is extracted based on the value of the select line

Number Theoretic Transforms (NTT)- The Heart of Lattice based PQC Designs

$$
\begin{array}{r}
1 + 2x + 3x^2 + 4x^3 \\
5 + 6x + 7x^2 + 8x^3 \quad \times \\
\hline
8x^3 + 16x^4 + 24x^5 + 32x^6 \\
7x^2 + 14x^3 + 21x^4 + 28x^5 \\
6x + 12x^2 + 18x^3 + 24x^4 \\
5 + 10x + 15x^2 + 20x^3 \\
\hline
5 + 16x + 34x^2 + 60x^3 + 61x^4 + 52x^5 + 32x^6 \quad +
\end{array}
$$

$$
\begin{array}{r}
32x^2 + 52x + 61 \\
x^4 + 1 \;\big)\; 32x^6 + 52x^5 + 61x^4 + 60x^3 + 34x^2 + 16x + 5 \\
32x^6 + 0x^5 + 0x^4 + 0x^3 + 32x^2 \\
\hline
52x^5 + 61x^4 + 60x^3 + 2x^2 + 16x + 5 \quad - \\
52x^5 + 0x^4 + 0x^3 + 0x^2 + 52x \\
\hline
61x^4 + 60x^3 + 2x^2 - 36x + 5 \quad - \\
61x^4 + 0x^3 + 0x^2 + 0x + 61 \\
\hline
60x^3 + 2x^2 - 36x - 56 \quad -
\end{array}
$$

Polynomial multiplication can be seen as a convolution of two vectors.
An alternate way of expressing the polynomials (rather than coefficients) is to evaluate the function (in this case 4 points).
Then, we can point to point multiply the results! - O(n) steps
The transformation should be however efficient – O(nlogn) steps!

NTT of Kyber

- **Kyber is based on NTT-friendly prime $q = 3329$**
- The prime is of the form $q - 1 = 2^8 \cdot 13$ and base field $\mathbb{Z}_q/(X^n + 1)$, where $n = 256$ has only 256-th root of unity but not 512-th root of unity
- Let ζ be the first 256-th primitive root of unity

$$X^{256} + 1 \longrightarrow \left(X^2 - \zeta^{2br(0)+1}\right)\left(X^2 - \zeta^{2br(1)+1}\right)\left(X^2 - \zeta^{2br(2)+1}\right) \ldots \left(X^2 - \zeta^{2br(127)+1}\right)$$

, $br \rightarrow$ bit reversal

- The NTT of $f \in R_q$ is given as: $\left(f \; mod \; X^2 - \zeta^{2br(0)+1}, \ldots, f \; mod \; X^2 - \zeta^{2br(127)+1}\right)$

$$\hat{f}_{2i} = f_0\zeta^{(2br(i)+1)\cdot 0} + f_2\zeta^{(2br(i)+1)\cdot 1} + f_4\zeta^{(2br(i)+1)\cdot 2} + \cdots + f_{254}\zeta^{(2br(i)+1)\cdot 127}$$

$$\hat{f}_{2i+1} = f_1\zeta^{(2br(i)+1)\cdot 0} + f_3\zeta^{(2br(i)+1)\cdot 1} + f_5\zeta^{(2br(i)+1)\cdot 2} + \cdots + f_{255}\zeta^{(2br(i)+1)\cdot 127}$$

$$NTT(f) = \hat{f} = (\hat{f}_0 + \hat{f}_1 X, \hat{f}_2 + \hat{f}_3 \mathrm{X}, \ldots, \hat{f}_{254} + \hat{f}_{255} X)$$

NTT of Dilithium

- **Dilithium is based on NTT-friendly prime $q = 8380417$**
- The prime is of the form $q - 1 = 2^{13} \cdot 1023$ and base field $\mathbb{Z}_q/(X^n + 1)$, where $n = 256$ has both 256-th and 512-th root of unity
- Let r be the first 512-th primitive root of unity

$$X^{256} + 1 \longrightarrow (X - r)(X + r)(X - r^{129})(X + r^{129}) \ldots (X - r^{127})(X + r^{127})(X - r^{255})(X + r^{255})$$

- The NTT of a ∈ R_q is given as: $(a\ mod\ X - r, a\ mod\ X + r, \ldots, a\ mod\ X - r^{255}, a\ mod\ X + r^{255})$

$NTT(a) = \hat{a} = (a(r_0), a(-r_0), \ldots, a(r_{127}), a(-r_{127}))$, where $r_i = t^{brv(128+i)}$, $brv \rightarrow$ bit reversal

Architecture of Unified NTT

- In case of Kyber the irreducible polynomial $X^{256}+1$ is split into 128 degree 2 polynomials

$$X^{256}+1 \longrightarrow \left(X^2-\zeta^{2br(0)+1}\right)\left(X^2-\zeta^{2br(1)+1}\right)\left(X^2-\zeta^{2br(2)+1}\right)\ldots\left(X^2-\zeta^{2br(127)+1}\right)$$

, $br \rightarrow$ bit reversal

- In case of Dilithium the irreducible polynomial $X^{256}+1$ is split into 256 degree 1 polynomials

$$X^{256}+1 \longrightarrow (X-r)(X+r)(X-r^{129})(X+r^{129})\ldots(X-r^{127})(X+r^{127})(X-r^{255})(X+r^{255})$$

- Consequently, in case of Kyber the NTT of a polynomial has 128 degree 1 polynomials

$$NTT(f)=\hat{f}=(\hat{f}_0+\hat{f}_1X, \hat{f}_2+\hat{f}_3X, \ldots, \hat{f}_{254}+\hat{f}_{255}X)$$

- Consequently, in case of Dilithium the NTT of a polynomial has 256 degree 0 polynomials

$NTT(a)=\hat{a}=(a(r_0), a(-r_0), \ldots, a(r_{127}), a(-r_{127}))$, where $r_i=t^{brv(128+i)}$, $brv \rightarrow$ bit reversal

- **So, in order to combine them we have stopped splitting the Dilithium polynomial after obtaining 128 degree 2 polynomials.**

Architecture of Unified NTT

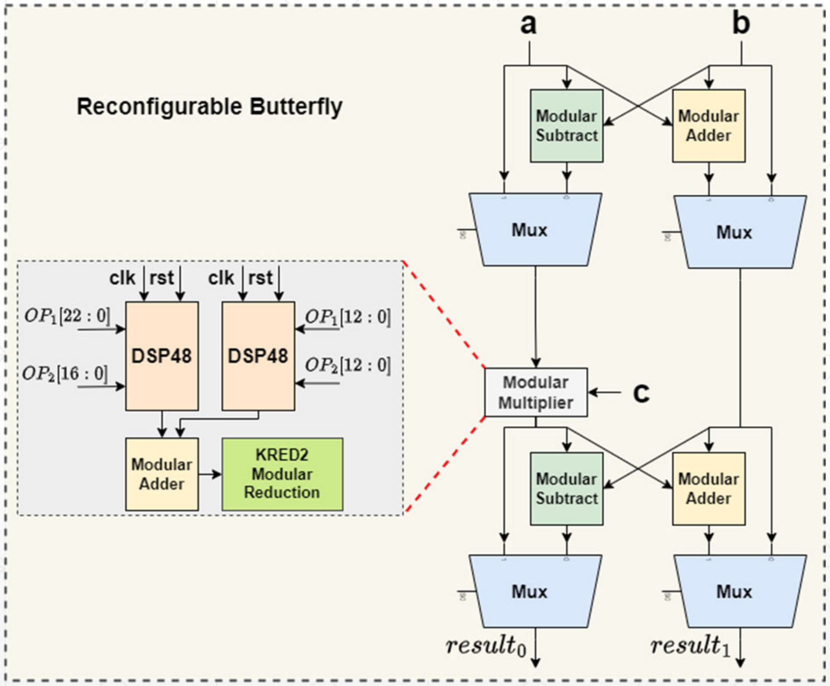
- Using the NTT and Inverse NTT, we can compute the product $f \cdot g$ of two elements $f, g \in R_q$
- The formulation for the calculation is $NTT^{-1}\left(NTT(f) \odot NTT(g)\right) = \hat{f} \odot \hat{g} = \hat{h}$
- So, the basecase multiplication consist of 128 products of degree 1 polynomials,
- While executing pointwise multiplication we followed the Karatsuba multiplication technique:

$$h_{2i} + h_{2i+1}X = \left(f_{2i} + f_{2i_1}X\right) \cdot (g_{2i} + g_{2i+1}X) \; mod \left(X^2 - \zeta^{2br(i)+1}\right)$$
$$h_{2i} = f_{2i}\, g_{2i} + f_{2i+1}\, g_{2i+1} \cdot \zeta^{2br(i)+1}$$
$$h_{2i+1} = (f_{2i} + f_{2i+1})(g_{2i} + g_{2i+1}) - (f_{2i}\, g_{2i} + f_{2i+1}\, g_{2i+1})$$

- So effectively the complexity of the polynomial multiplication is reduced from $O(n^2)$ to $O(n \log n)$

NTT Multiplier

We have implemented both the Cooley-Tukey (for forward NTT) and the Gentleman-Sande (for Inverse NTT) algorithms to implement the NTT multiplier.



Butterfly Architecture of our NTT Core

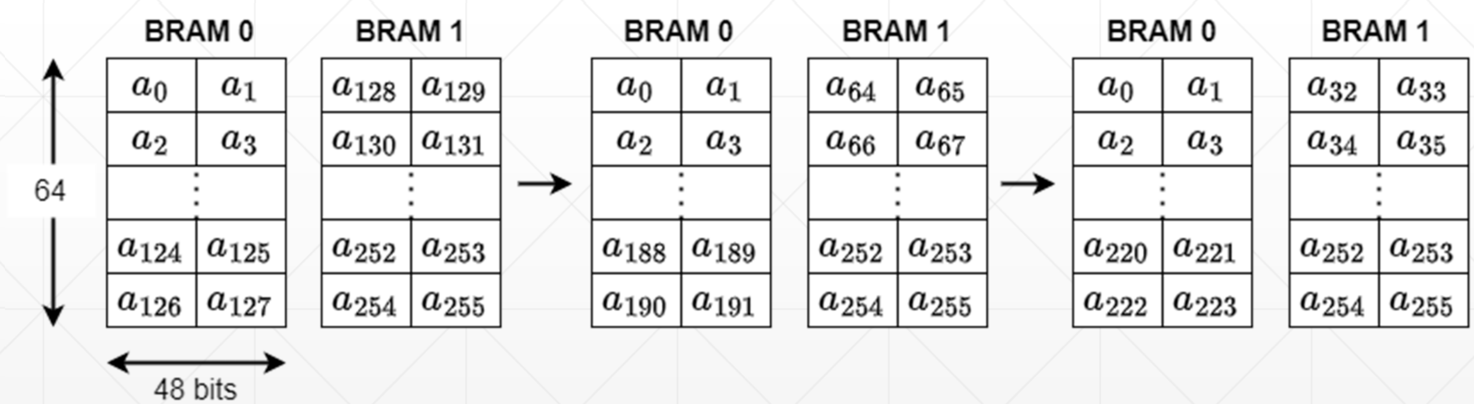
- **Figure shows the basic structure of a butterfly unit which is capable of processing two coefficients at a time**.
- The NTT multiplier houses 2 such butterfly computation units that is capable of processing four polynomial coefficients after each iteration
- The butterfly unit in the figure can operate in 3 separate modes: **Forward NTT, Inverse NTT, Point-wise multiplication**
- Depending on the operating mode, the input c can be switched between **twiddle factors** or a **polynomial coefficient**.

Forward NTT	Inverse NTT
$t = \zeta . a$	$a = a + b$
$a = a - t$	$b = a - b$
$b = b + t$	$b = \zeta . b$

ζ is the twiddle factor

NTT Multiplier

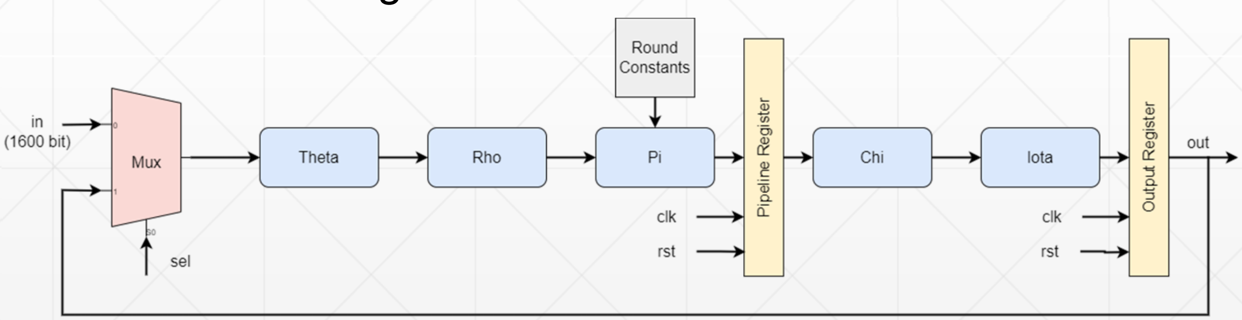
- The NTT multiplier block has two separate BRAM units capable of holding two coefficients in a single memory cell separated by an index of s, where $s \in \{128,64,32,16,8,4,2\}$
- The figure below shows an example of the content of the BRAM units



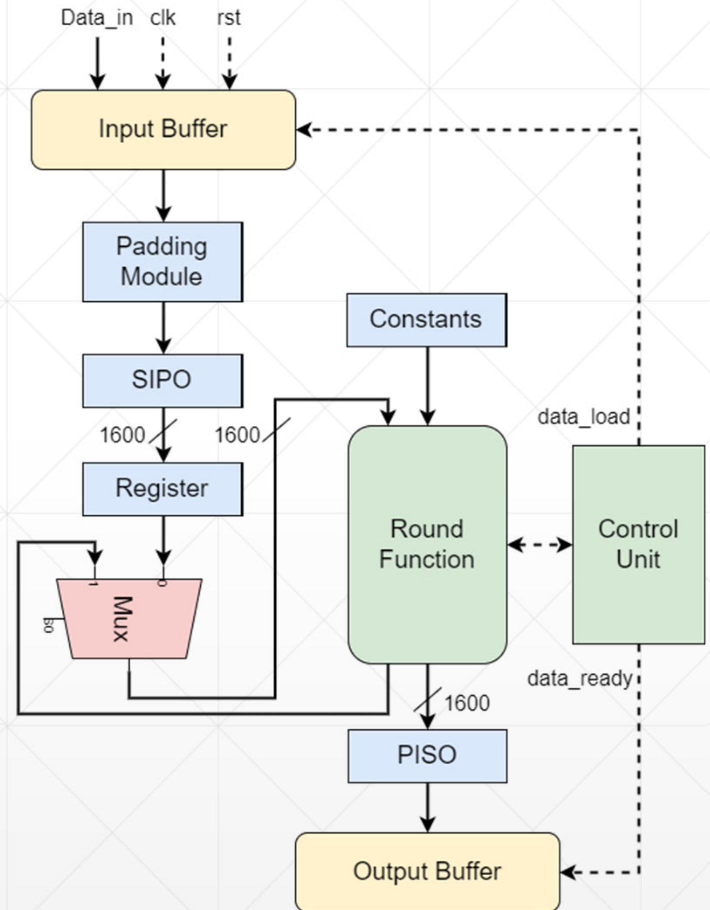
NTT multiplier scheduling in the NTT RAM

KECCAK Module for Kyber and Dilithium

- Kyber requires four modes of the Keccak core - namely SHA3-256, SHA3-512, SHAKE-128, and SHAKE-256.
    - **Whereas, Dilithium requires two modes - SHAKE-128 and SHAKE-256.**
- These modes are implemented using the Keccak sponge structure internally equipped with separate wrappers and individual buffers that are multiplexed based on the micro-coded control signals.

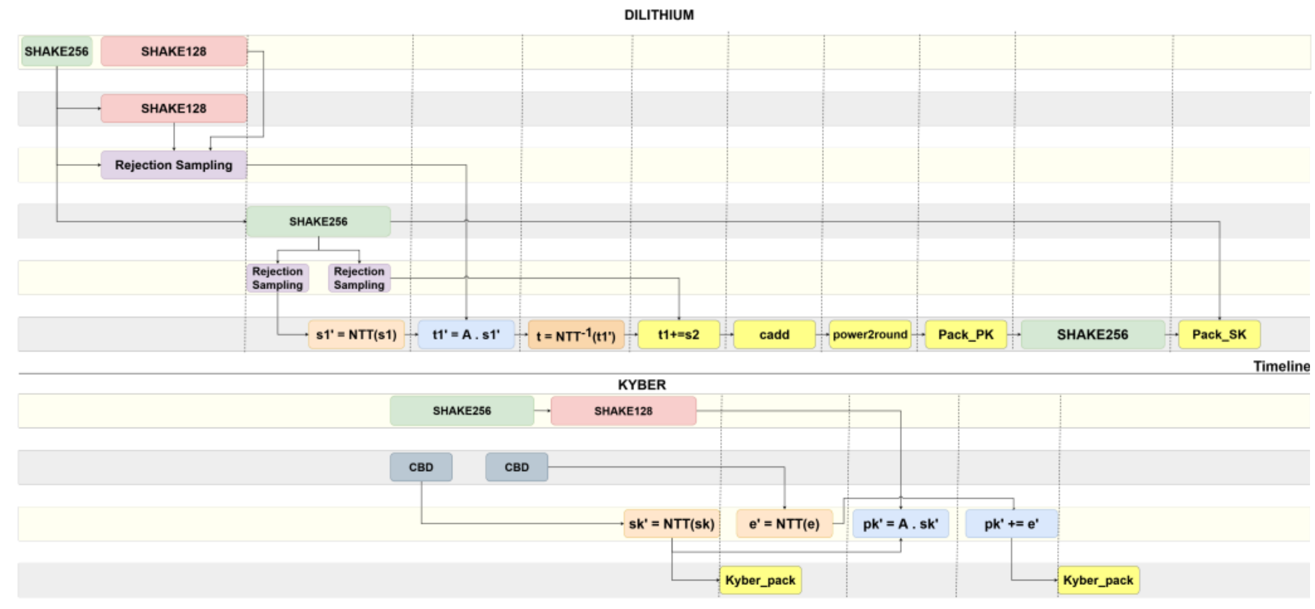


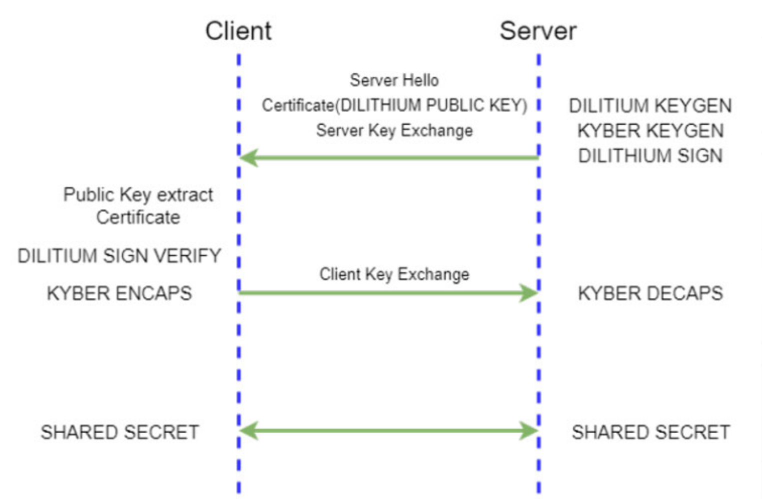
Proposed architecture of transformation round



Proposed architecture of the KECCAK hash function

Parallel Scheduling of the Key generation of Kyber and Dilithium





Implementation Details

- The table below shows the resource utilization of various components of Kyber and Dilithium when implemented on the FPGA

Algorithm	Components	LUTs	DSPs	BRAMs
Dilithium	Decompose	504	-	-
	MakeHint	80	-	-
	UseHint	708	-	-
	Powe2Round	75	-	-
	Pack	658	-	-
	Unpack	325	-	-
	Encode	354	-	-
	Decode	160	-	-
	SampleInBall	485	-	-
	Verify	16	-	-
	Rejection Core	718	-	-
Kyber	Compress/Decompress	258	-	-
	Encode	581	-	-
	Decode	268	-	-
	COPY	30	-	-
	CMOV	35	-	-
	Rejection Core	185	-	-
	Verification Core	98	-	-
Common	KECCAK	10879	-	-
	Data Memory	-	-	19
	NTT Multiplier	2899	4	1
	Controller	2618	-	5
Total		22125	4	25

Implementation Details

- The table below shows the comparison of our proposed design with the state of the art Dilithium and Kyber hardware designs

Works	Algorithm	LUTs	Sign	Verify	Encaps	Decaps	AT Product
			μs	μs	μs	μs	
[8]+[9]	Dilithium II+ Kyber-512	37935	178	121	10	20	12.4
	Dilithium III+ Kyber-768	42683	310	221	15	25	24.3
	Dilithium V+ Kyber-1024	58000	503	377	20	30	53.9
Our Design	**Dilithium II+ Kyber-512**	**22125**	**200**	**113**	**21.27**	**26.32**	**7.9**
	Dilithium III+ Kyber-768	**22125**	**350**	**181**	**27.11**	**31.85**	**13.0**
	Dilithium V+ Kyber-1024	**22125**	**500**	**270**	**33.11**	**39.89**	**18.6**

Resource utilization and timing details of our proposed design

Ref.8 : Georg Land et al. "A hard crystal - implementing dilithium on reconfigurable hardware." IACR eprint, 2021.

Ref.9: Mojtaba Bisheh-Niasar et al. "High-speed NTT-based polynomial multiplication accelerator for crystals-kyber post-quantum cryptography." IACR eprint, 2021.

Step 5: Side-channels

Side-channels : Importance for PQC standardization

- The hard problems do not factor in physical attacks. But such attack vectors can still leak secret cryptographic material in presence of side channels.

Side-channels : Importance for PQC standardization

- The hard problems do not factor in physical attacks. But such attack vectors can still leak secret cryptographic material in presence of side channels.

- NIST also considers evaluation of PQ cryptosystems against such attack vectors.
  - Quote NIST: *"NIST seeks any distinguishing information in the realm of side-channel analyses that especially indicate a reason for NIST to prefer one of the finalists over the others."*

Side-channels : Importance for PQC standardization

- The hard problems do not factor in physical attacks.
  - <u>But such attack vectors can still leak secret cryptographic material in presence of side channels.</u>

- **NIST also considers evaluation of PQ cryptosystems against such attack vectors.**
  - **Quote NIST: *"NIST seeks any distinguishing information in the realm of side-channel analyses that especially indicate a reason for NIST to prefer one of the finalists over the others."***

- Two kinds of adversaries:
  - **Passive adversary** : Passively observes leakage and tries to reconstruct secret cryptographic material. Example: power side-channel
  - **Active adversary** : Actively injects faults in computation, and uses differential computation paths to reconstruct secret cryptographic material.

Side-channels : Case study for PQ Lattice KEMs

- Presenting a case-study on PQ Lattice KEMs. Similar issues plague other cryptosystems too.
- Rely on distinguishable effective/ineffective faults to draw inferences

Algorithm 3 LPR.KEM.Decaps.

Input: (sk, c)

m = LPR.PKE.Dec(sk, c) — Fault here!

$(K'_H, r') = \mathcal{G}(\text{pkh}, m)$

$c_* = \text{LPR.PKE.Enc}(pk, m, r')$

if $c_* = c$ **then**

   $K = \mathcal{KDF}(K'_H, \mathcal{H}(c))$ — Ineffective fault

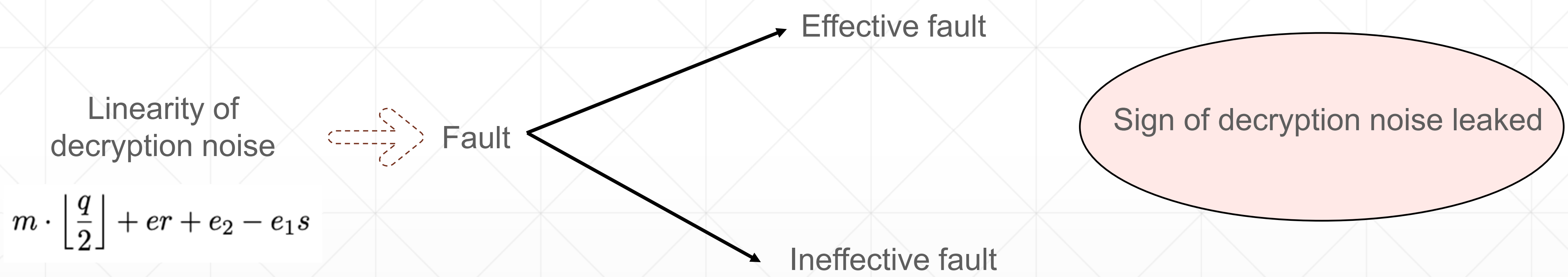
else

   $K = \mathcal{KDF}(z, \mathcal{H}(c))$ // Use randomness z to output an incorrect key — Effective fault

return (c, K)

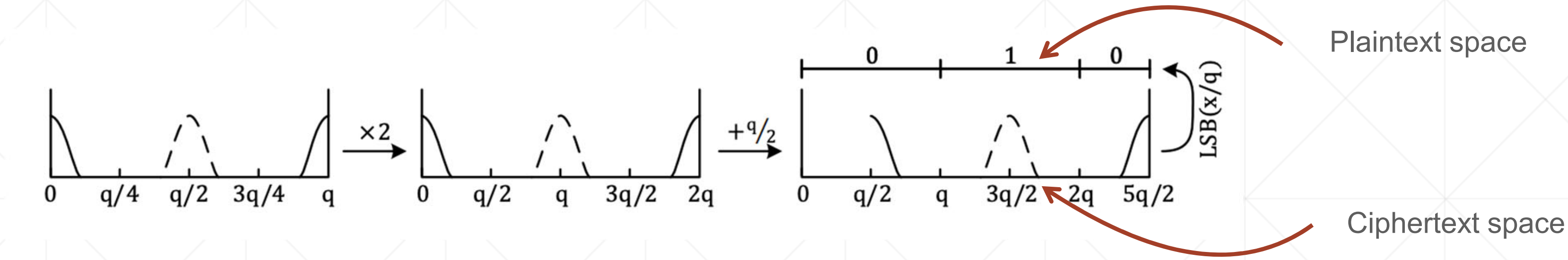
Side-channels : Case study for PQ Lattice KEMs

- Presenting a case-study on PQ Lattice KEMs. Similar issues plague other cryptosystems too.
- Rely on distinguishable effective/ineffective faults to draw inferences

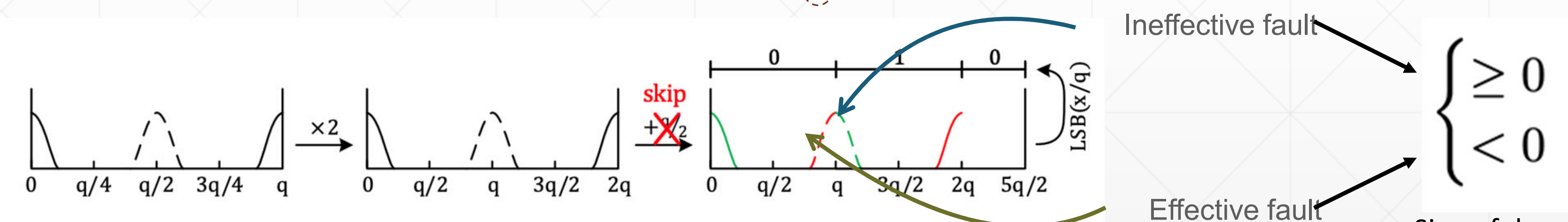


Side-channels : Case study for PQ Lattice KEMs

➢ Attack on Kyber



Decoding in Kyber



System of in-equations then solved by belief propagation

Sign of decryption noise inferred

Decoding in Kyber after instruction skip fault

Side-channels : Case study for PQ Lattice KEMs

- Countermeasures

Algorithm 3 LPR.KEM.Decaps.

Input: (sk, c)
m = LPR.PKE.Dec(sk, c, seed)
$(K'_H, r') = \mathcal{G}(pkh, m)$
$c_* =$ LPR.PKE.Enc(pk, m, r')
if $c_* = c$ **then**
    $K = \mathcal{KDF}(K'_H, \mathcal{H}(c))$
else
    $K = \mathcal{KDF}(z, \mathcal{H}(c))$ // Use randomness z to output an incorrect key
return (c, K)

Shuffle the order of coefficients being processed

Algorithm 3 LPR.KEM.Decaps.

Input: (sk, c)
for i = 1 to *k* **do** // *k* repetitions
    m_i = LPR.PKE.Dec(sk, c)
if Check($m_1, m_2, ..., m_k$) = 0 **then**
    abort // If a fault is detected
$(K'_H, r') = \mathcal{G}(pkh, m_1)$
$c_* =$ LPR.PKE.Enc(pk, m_1, r')
if $c_* = c$ **then**
    $K = \mathcal{KDF}(K'_H, \mathcal{H}(c))$
else
    $K = \mathcal{KDF}(z, \mathcal{H}(c))$ // Use randomness z to output an incorrect key
return (c, K)

Repeat the computation multiple times (k repetitions protect against k-1 faults

Future

The future is Post-Quantum

- Expect worldwide organizations to adapt NIST standardized cryptosystems

The future is Post-Quantum

- Expect worldwide organizations to adapt NIST standardized cryptosystems

- "Aren't classical cryptosystems secure until Quantum Computers become practical?"
  - No!
  - **"Harvest-Now-Decrypt-Later" attacks**: stores several ciphertexts to be decrypted 10-15 years into the future. A real threat to national security!
  - Establishes the need to transition to post-quantum cryptosystems ASAP

The future is Post-Quantum

- Expect worldwide organizations to adapt NIST standardized cryptosystems

- "Aren't classical cryptosystems secure until Quantum Computers become practical?"
  - No!
  - **"Harvest-Now-Decrypt-Later" attacks**: stores several ciphertexts to be decrypted 10-15 years into the future. A real threat to national security!
  - Establishes the need to transition to post-quantum cryptosystems ASAP

- "What are the ideal transition steps?"
  - Let the NIST standardization process end (the fourth round ends about 2025)

The future is Post-Quantum

- Expect worldwide organizations to adapt NIST standardized cryptosystems

- "Aren't classical cryptosystems secure until Quantum Computers become practical?"
  - No!
  - **"Harvest-Now-Decrypt-Later" attacks**: stores several ciphertexts to be decrypted 10-15 years into the future. A real threat to national security!
  - Establishes the need to transition to post-quantum cryptosystems ASAP

- "What are the ideal transition steps?"
  - Let the NIST standardization process end (the fourth round ends about 2025)
  - All cryptosystems are open-source by design, and such implementations are well-audited.

The future is Post-Quantum

- Expect worldwide organizations to adapt NIST standardized cryptosystems

- "Aren't classical cryptosystems secure until Quantum Computers become practical?"
  - No!
  - **"Harvest-Now-Decrypt-Later" attacks**: stores several ciphertexts to be decrypted 10-15 years into the future. A real threat to national security!
  - Establishes the need to transition to post-quantum cryptosystems ASAP

- "What are the ideal transition steps?"
  - Let the NIST standardization process end (the fourth round ends about 2025)
  - All cryptosystems are open-source by design, and such implementations are well-audited.
  - At the least, use out-of-box PQ implementation with recommended parameter settings.

The future is Post-Quantum

- Expect worldwide organizations to adapt NIST standardized cryptosystems

- "Aren't classical cryptosystems secure until Quantum Computers become practical?"
  - No!
  - **"Harvest-Now-Decrypt-Later" attacks**: stores several ciphertexts to be decrypted 10-15 years into the future. A real threat to national security!
  - Establishes the need to transition to post-quantum cryptosystems ASAP

- "What are the ideal transition steps?"
  - Let the NIST standardization process end (the fourth round ends about 2025)
  - All cryptosystems are open-source by design, and such implementations are well-audited.
  - At the least, use standard PQ implementation with recommended parameter settings.
  - Any edits to standar PQ implementation requires *reproving* security in the relevant hard-problem assumption through standard cryptographic reduction techniques.

Thank You

Binomial Sampler

- A binomial sampler is used as substitution for the Gaussian sampler
- The binomial distribution that is parametrized by $k = \sigma^2$ is sufficiently close to a discrete Gaussian distribution with standard deviation σ and does not significantly decrease the security level.
- Algorithm:
  - Uniformly sampling two k-bit vectors and computing their respective Hamming weights.
  - Subtracting the Hamming weights of both bit vectors.
- As k scales quadratically with σ this approach is suited for lattice-based encryption or key exchange schemes. Signature schemes usually require larger standard deviations.
- This is implemented in NewHope and Kyber

[Ref] Alkim, E., Ducas, L., Pöppelmann, T., Schwabe, P.: Post-quantum key exchange - a new hope. In: Proceedings of the 25th USENIX Security Symposium

PQ resistant hard problems : Code-based

- Code-based:

Problem **((Decisional) Syndrome Decoding problem)** *Given an $(n-k)\times n$ parity-check matrix H for C, a vector $y \in \mathbb{F}_2^{n-k}$, and a target $t \in \mathbb{N}$, determine whether there exists $x \in \mathbb{F}_2^n$ that satisfies $Hx^T = y$ and $|x| \leq t$.*

- **Problem description**: given a parity matrix **H** and a binary target **y**, find **x**

PQ resistant hard problems : Code-based

- Code-based:

Problem **((Decisional) Syndrome Decoding problem)** *Given an* $(n-k) \times n$ *parity-check matrix* H *for* C*, a vector* $y \in \mathbb{F}_2^{n-k}$*, and a target* $t \in \mathbb{N}$*, determine whether there exists* $x \in \mathbb{F}_2^n$ *that satisfies* $Hx^T = y$ *and* $|x| \leq t$.

- **Problem description:** given a parity matrix **H** and a binary target **y**, find "small" preimage **x**
- **Why is it *hard*:** Because the **x** to be recovered is bounded by Hamming weight **t**. Small **t** means we need to find a *small* **x**, and that is provably difficult for "correct parameterization" of the problem (i.e. for large enough values of **n** and **k**).

PQ resistant hard problems : Code-based

- Code-based:

Problem ((Decisional) Codeword Finding problem) *Given an $(n-k) \times n$ parity-check matrix H for C and a target $w \in \mathbb{N}$, determine whether there exists $x \in \mathbb{F}_2^n$ that satisfies $Hx^T = 0$ and $|x| = w$.*

- **Problem description**: given a parity matrix **H** and an integer target **w** (**w** > 0), find "small" **x** in the kernel of **H**

PQ resistant hard problems : Code-based

- Code-based:

Problem ((Decisional) Codeword Finding problem) *Given an* $(n-k)\times n$ *parity-check matrix* H *for* C *and a target* $w \in \mathbb{N}$*, determine whether there exists* $x \in \mathbb{F}_2^n$ *that satisfies* $Hx^T = 0$ *and* $|x| = w$.

- **Problem description**: given a parity matrix **H** and an integer target **w** (**w** > 0), find "small" **x** in the kernel of **H**
- **Why is it *hard*:** Because the **x** to be recovered is bounded by Hamming weight **w**. Small **w** means we need to find a *small* **x**, and that is provably difficult for "correct parameterization" of the problem (i.e. for large enough values of **n** and **k**).

PQ resistant hard problems : Multivariate-based

- Multivariate-based:

Problem **((Decisional) Multivariate Quadratic ($\mathcal{MQ}$) polynomial problem)** *Given a finite field $\mathbb{F}$ and a system of m quadratic polynomials of n variables x_i:*

$$f_k(x_1,\ldots,x_n) = \sum_{1\le i\le j\le n} a_{ij}^{(k)} x_i x_j + \sum_{1\le i\le n} b_i^{(k)} x_i + c^{(k)} = 0,$$

for k from 1 to m, where $a_{ij}^{(k)}, b_i^{(k)}, c^{(k)}$ are all in $\mathbb{F}$, determine if there exists a solution in $\mathbb{F}^n$.

- **Problem description**: Given **m** quadratic polynomials with **n** variables each, find a solution "common" to the kernel of *each* polynomial

PQ resistant hard problems : Multivariate-based

- Multivariate-based:

Problem **((Decisional) Multivariate Quadratic ($\mathcal{MQ}$) polynomial problem)** *Given a finite field $\mathbb{F}$ and a system of m quadratic polynomials of n variables x_i:*

$$f_k(x_1,\ldots,x_n) = \sum_{1\le i\le j\le n} a_{ij}^{(k)} x_i x_j + \sum_{1\le i\le n} b_i^{(k)} x_i + c^{(k)} = 0,$$

for k from 1 to m, where $a_{ij}^{(k)}, b_i^{(k)}, c^{(k)}$ are all in $\mathbb{F}$, determine if there exists a solution in $\mathbb{F}^n$.

- **Problem description**: Given **m** quadratic polynomials with **n** variables each, find a solution "common" to the kernel of *each* polynomial

- **Why is it *hard*:** Finding an element in the kernel of a multivariate polynomial amounts to finding its root. The "hardness" here comes from the requirement of finding a "common" root to all polynomials (or one solution to all polynomials).

PQ resistant hard problems : Multivariate-based

- Multivariate-based:

Problem **((Decisional) MinRank problem)** *Given a finite field* $\mathbb{F}$*, k matrices* M_i *of size* $m \times n$ *with entries in* $\mathbb{F}$*, and a rank bound r, determine if there exist values* $c_i \in \mathbb{F}$ *to satisfy the following equation:*

$$rank\left(\sum_{i=1}^{k} c_i M_i\right) \leq r.$$

- **Problem description**: Given **k** matrices in some field, find a linear combination of these matrices such that the rank of the resultant matrix is bounded by **r**

PQ resistant hard problems : Multivariate-based

- Multivariate-based:

Problem **((Decisional) MinRank problem)** *Given a finite field $\mathbb{F}$, k matrices M_i of size $m \times n$ with entries in $\mathbb{F}$, and a rank bound r, determine if there exist values $c_i \in \mathbb{F}$ to satisfy the following equation:*

$$rank\left(\sum_{i=1}^{k} c_i M_i\right) \leq r.$$

- **Problem description**: Given **k** matrices in some field, find a linear combination of these matrices such that the rank of the resultant matrix is bounded by **r**

- **Why is it *hard*:** Finding a linear combination of matrices **M** is straightforward, however, the "hardness" comes from the requirement to bound the final result by small **r**

PQ resistant hard problems : Lattice-based

- Lattice-based:

Problem (The Short Integer Solution ($SIS_{n,m,q,\beta}$) problem) *Let n, m, q be positive integers, and let β be a positive real number. Given a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, chosen uniformly at random, find a nonzero integer vector $\mathbf{z} \in \mathbb{Z}^m$ of Euclidean norm $||\mathbf{z}|| \leq \beta$ such that $\mathbf{Az} = \mathbf{0} \in \mathbb{Z}_q^n$.*

- **Problem description**: Given a matrix **A**, find a "short" **z** in the kernel of **A**

PQ resistant hard problems : Lattice-based

- Lattice-based:

Problem **(The Short Integer Solution ($SIS_{n,m,q,\beta}$) problem)** *Let n, m, q be positive integers, and let β be a positive real number. Given a matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, chosen uniformly at random, find a nonzero integer vector $\mathbf{z} \in \mathbb{Z}^m$ of Euclidean norm $||\mathbf{z}|| \leq \beta$ such that $\mathbf{Az} = \mathbf{0} \in \mathbb{Z}_q^n$.*

- **Problem description**: Given a matrix **A**, find a "short" **z** in the kernel of **A**
- **Why is it *hard*:** Finding an point in the kernel of lattice defined by **A** is straigtforward through methods like Gaussian Elimination. The "hardness" comes from the requirement of bounding the norm of **z** (i.e. find a "short" element in the kernel of **A**).

PQ resistant hard problems : Lattice-based

- Lattice-based:

Problem **(The** $Search - NTRU_{R,q,\mathcal{D},\gamma}$ **problem)** *Let q be a positive integer, γ be a positive real number, and R be a ring of the form $R = \mathbb{Z}_q[x]/\Phi$ (where Φ is a monic polynomial). Given an element $h \in R$ drawn from some distribution $\mathcal{D}$, such that there exists nonzero $(f,g) \in R^2$ that satisfy $h \cdot f = g \bmod q$ and have small Euclidean norms $||f||, ||g|| \leq \sqrt{q}/\gamma$, find such a pair (f,g).*

- **Problem description**: Given an element **h** from the polynomial ring **R,** find two "short" polynomials **f** and **g** such that **h . f = g mod q** with overwhelming probability.

PQ resistant hard problems : Lattice-based

- Lattice-based:

Problem **(The** *Search* $-$ $NTRU_{R,q,\mathcal{D},\gamma}$ **problem)** *Let q be a positive integer, γ be a positive real number, and R be a ring of the form $R = \mathbb{Z}_q[x]/\Phi$ (where Φ is a monic polynomial). Given an element $h \in R$ drawn from some distribution $\mathcal{D}$, such that there exists nonzero $(f,g) \in R^2$ that satisfy $h \cdot f = g \bmod q$ and have small Euclidean norms $||f||, ||g|| \leq \sqrt{q}/\gamma$, find such a pair (f,g).*

- **Problem description**: Given an element **h** from the polynomial ring **R,** find two "short" polynomials **f** and **g** such that **h . f = g mod q** with overwhelming probability.
- **Why is it *hard*:** The "hardness" is derived from two requirements:
  - Both **f** and **g** are norm-bounded
  - Both **f** and **g** are in **R**.
    - **Why the second requirement:** Otherwise, the adversary samples a "small" **f** in **R**, then simply computes (**h . f) mod q**. If this result need not be in **R,** then it is easy to solve

PQ resistant hard problems : Lattice-based

- Lattice-based:

Problem **(The** *Search-LWE*$_{n,m,q,\mathcal{B},\chi}$ **problem)** *Let* $\mathbf{s} \in \mathbb{Z}_q^n$ *be chosen from some distribution* $\mathcal{B}$. *Given* m *samples* $(\mathbf{a}_1, b_1), \ldots, (\mathbf{a}_m, b_m) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ *drawn independently at random from the distribution* $A_{\mathbf{s},\chi}$, *find* $\mathbf{s}$.

- **Problem description**: Given a public matrix **A** and a secret **s**, invert the functional evaluation of **(A.s + e)** where **e** is some error drawn from a "narrow" distribution (like Gaussian).

PQ resistant hard problems : Lattice-based

- Lattice-based:

Problem **(The** *Search-LWE*$_{n,m,q,\mathcal{B},\chi}$ **problem)** *Let* $\mathbf{s} \in \mathbb{Z}_q^n$ *be chosen from some distribution* $\mathcal{B}$*. Given m samples* $(\mathbf{a}_1, b_1), \ldots, (\mathbf{a}_m, b_m) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ *drawn independently at random from the distribution* $A_{\mathbf{s},\chi}$*, find* $\mathbf{s}$.

- **Problem description**: Given a public matrix **A** and a secret **s**, invert the functional evaluation of **(A.s + e)** where **e** is some error drawn from a "narrow" distribution (like Gaussian).

- **Why is it *hard*:** Given **b = (A.s + e)** , it is difficult to invert for the correct choice of the distribution of error **e**

NTT Multiplier

Algorithm for NTT

Require: $p, N, q; twiddle\_factor\_array[N]$
Ensure: $\hat{p}$

```
twiddle_count = 1
for s = 2^(N-1) to 1 by s/2 do
    for start = 0 to N - 1 by j+s do
        zeta = twiddle_factor_array[++twiddle_count]
        for j = start to start + s do
            t = zeta · p[j + s] mod q
            p[j + s] = p[j] - t mod q
            p[j] = p[j] + t mod q
        end for
    end for
end for
```

Algorithm for Inverse-NTT

Require: $\hat{p}, N, q; twiddle\_factor\_array[N]$
Ensure: p

```
twiddle_count = N
for s = 1 to N - 1 by s · 2 do
    for start = 0 to N - 1 by j+s do
        zeta = twiddle_factor_array[--twiddle_count]
        for j = start to start + s do
            t = p[j]
            p[j] = (t + p[j + s])/2 mod q
            p[j + s] = (t - p[j + s])/2 mod q
            p[j + s] = zeta · p[j + s] mod q
        end for
    end for
end for
```

Compress/Decompress unit of Kyber

- The compress operation requires division by q and rounding.
  - $Compress(x) = \lceil \left(\frac{2^d}{q}\right) \cdot\ x \rfloor\ (mod\ 2^d)$

- The decompress unit performs division by power-of-two and rounding operation
  - $Decompress(x) = \lceil \left(\frac{q}{2^d}\right) \cdot\ x \rfloor$

- The value of d varies as follows: $\{1, 4, 5, 10, 11\}$

Compress/Decompress unit of Kyber

- The compress operation requires division by q and rounding.
  - $Compress(x) = \lceil \left(\frac{2^d}{q}\right) \cdot x \rfloor \ (mod\ 2^d)$
- The decompress unit performs division by power-of-two and rounding operation
  - $Decompress(x) = \lceil \left(\frac{q}{2^d}\right) \cdot x \rfloor$
- The value of d varies as follows: $\{1, 4, 5, 10, 11\}$

Compress Algorithm used for Kyber

$$\begin{aligned}
&\boldsymbol{if\ d == 1: t = (10079 \cdot x); y = (t \gg 24) + (t[23] \gg 23)}\\
&\boldsymbol{if\ d == 4: t = (315 \cdot x); y = (t \gg 16) + (t[15] \gg 15)}\\
&\boldsymbol{if\ d == 5: t = (630 \cdot x); y = (t \gg 16) + (t[15] \gg 15)}\\
&\boldsymbol{if\ d == 10: t = (5160669 \cdot x); y = (t \gg 24) + (t[23] \gg 23)}\\
&\boldsymbol{if\ d == 11: t = (10321339 \cdot x); y = (t \gg 24) + (t[23] \gg 23)}\\
&\boldsymbol{return\ y\ (mod\ 2^d)}
\end{aligned}$$