Post-Quantum Cryptography, Lattice Methods, and Quantum Key Distribution

Zuhair Khan

What Will We Cover?

1. **Background & Prerequisites:** overview of classical cryptography, recap of shor’s, how does it break classical cryptography
2. **PQC**: What it is, why it matters
3. **Lattice Methods**: The math behind modern PQC
4. **QKD**: Secure communication through quantum physics
5. **Comparisons & Outlook**: When and where each is useful

Why do we care about shor’s algorithm

- RSA, ECC, DH rely on the hardness of number-theoretic problems
- Shor’s algorithm breaks these using a quantum computer
- Key breakthrough: reduces factoring to order-finding

The Problem: Factoring N = p × q

- Classical factoring is hard (no known poly-time algorithm)
- Goal: Given N (a large composite), recover its prime factors p and q
- Shor’s insight: Turn this into a periodicity problem

Shor’s Algorithm Overview

$N = p_1 p_2 \qquad p_1, p_2 > 2 \qquad N = 2k+1 \qquad p_1 \neq p_2 \qquad N \neq p^{\alpha}$

SHOR'S AL. INPUT: N OUTPUT: $\{p_1, p_2\}$

SELECT RANDOM a s.t. $1 < a < N$

→ IF $\gcd(a,N) > 1$, $\{p_1, p_2\} = \left\{\gcd(a,N), \dfrac{N}{\gcd(a,N)}\right\}$ END

→ IF $\gcd(a,N) = 1$, FIND MIN $r > 0$ s.t. $a^r = 1 \bmod N$ (QUANTUM)

→ IF $2 \mid r \wedge a^{\frac{r}{2}} \neq -1 \bmod N$,

$\{p_1, p_2\} = \left\{\gcd\left(a^{\frac{r}{2}} - 1, N\right), \gcd\left(a^{\frac{r}{2}} + 1, N\right)\right\}$ END

→ ELSE RESTART WITH NEW a

Order-Finding Example (Classical Simulation)

$N = p_1 p_2 \quad p_1 \neq p_2 \quad p_1, p_2 > 2$

$1 < a < N \qquad \gcd(a, N) = 1$

MIN $r > 0$ s.t. $a^r = 1 \mod N$

$a^r - 1 = (a^{\frac{r}{2}} - 1)(a^{\frac{r}{2}} + 1) = 0 \mod N$

EVEN $r \qquad a^{\frac{r}{2}} \neq -1 = N-1 \mod N$

$21 = 3 \times 7$

$20 = -1$

$3, 6, 9, 12, 15, 18 \rightarrow \gcd(a, 21) = 3$

$7, 14 \rightarrow \gcd(a, 21) = 7$

a^1	8	13	$\boxed{20}$	4	16	2	11	5	17	10	19
a^2	1	1	1	16	4	4	16	4	16	16	4
a^3	8	13	20	1	1	8	8	$\boxed{20}$	$\boxed{20}$	13	13
a^4	1	1	1	4	16	16	4	16	4	4	16
a^5	8	13	20	16	4	11	2	17	5	19	10
a^6	1	1	1	1	1	1	1	1	1	1	1

$8 \pm 1 = 7, 9 \qquad 13 \pm 1 = 12, 14$

Quantum Step: Why Classical Fails

- **Goal:** Find the *period* x such that $a^x \equiv 1 \bmod N$
- Classically, this requires exponential time (no known efficient algorithm)
- Quantumly, we leverage **superposition** and **interference** to spot periodicity

Prepare a superposition over all x from 0 to $N-1$:

$$\frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle$$

Compute $f(x) = a^x \mod N$ and entangle:

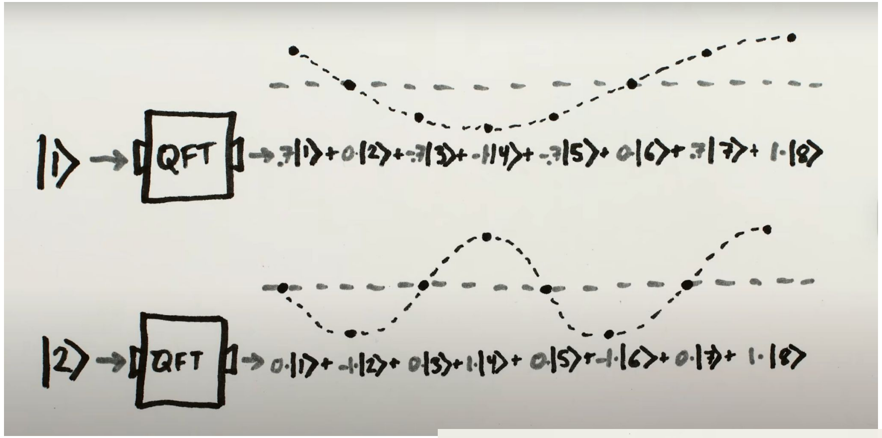
$$|x\rangle|f(x)\rangle$$

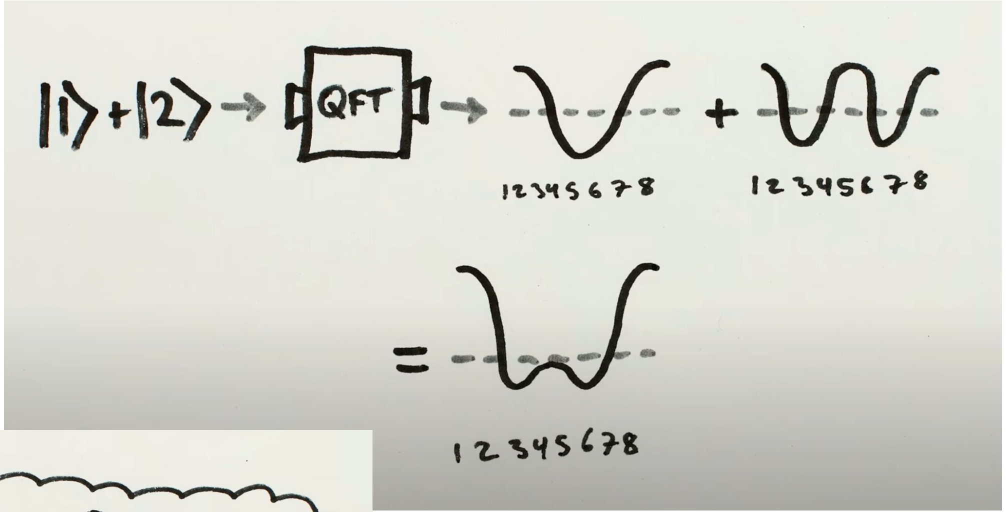
Measure second register → collapses to values of x such that $f(x) = y$

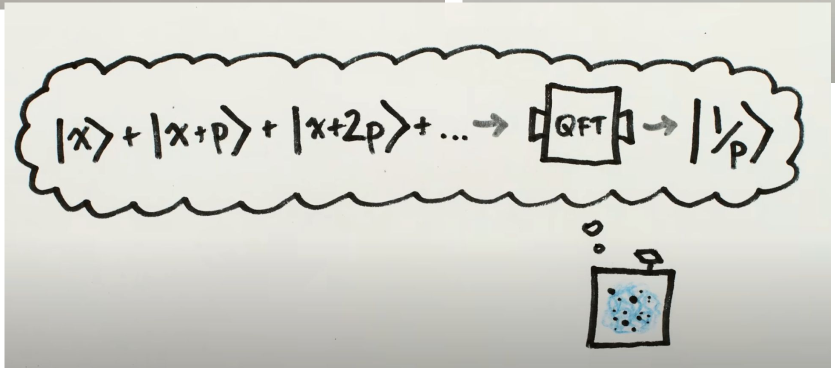
Leaves a periodic superposition in the first register:

$$|x\rangle + |x+r\rangle + |x+2r\rangle + \ldots$$

Why QFT Works — Extracting the Period (visually)







Classical Cryptography

Why is Classical Cryptography at Risk?

- Shor’s Algorithm breaks:
  - RSA (Factoring integers)
  - ECC (Discrete logarithm)
  - DH (Key exchange)
- Quantum computers threaten the foundation of current internet security.
- “What happens when encryption no longer protects your bank, government, or health data?”

The Essence of Encryption

1. **Encryption** is the process of transforming readable information (plaintext) into an unreadable format (ciphertext) using a key, so only authorized parties can access it.
2. **Decryption** is the reverse process, turning ciphertext back into plaintext using the correct key.
3. **Cryptography** is the science of secure communication. It ensures:
   - Confidentiality
   - Integrity
   - Authenticity
   - Non-repudiation

Kerckhoffs's Principle:
A cryptographic system should remain secure even if everything about the system is public — except the key.

"If someone intercepts the message, they can't read it unless they have the right key — even if they know the algorithm."

How Do Classical Encryption Schemes Work?

🔐 Symmetric Encryption

- Same key for encryption & decryption.
- Fast, efficient, but **key distribution is a challenge**.
- Examples: **AES**, **ChaCha20**
- Key idea: Dk(Ek(m))=mD_k(E_k(m)) = mDk(Ek(m))=m

🧠 *Used for bulk data encryption after secure key exchange.*

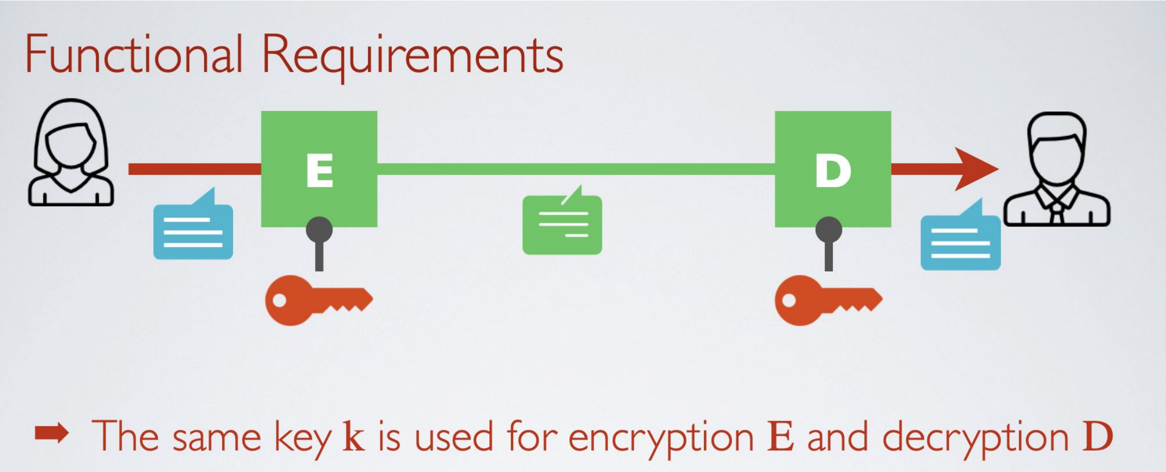
	Symmetric	Asymmetric
pro	Fast	No key agreement
cons	Key agreement	Very slow

🔐 Asymmetric Encryption

- Uses a **key pair**: public key (for encryption), private key (for decryption).
- Slower but supports **secure key exchange and digital signatures**.
- Examples: **RSA**, **ECC**
- Key idea:
 DKs(EKp(m))=mD_{K_s}(E_{K_p}(m)) =
mDKs(EKp(m))=m

🧠 *Used to establish secure channels (e.g. HTTPS), then switch to symmetric.*

Symmetric Encryption



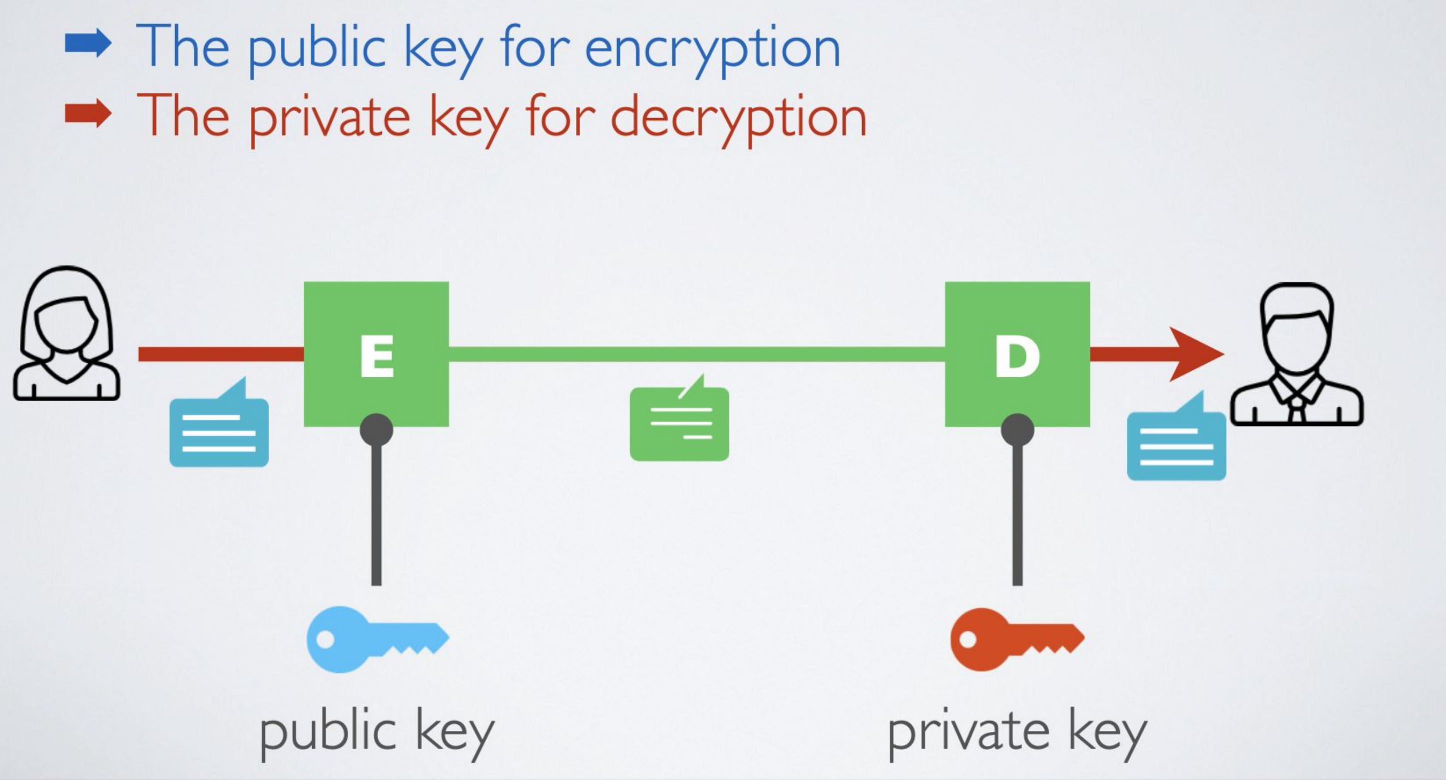
1. Dk(Ek(m))=m for every k, Ek is an injection with inverse Dk

2. Ek(m) is easy to compute (either polynomial or linear)

3. Dk(c) is easy to compute (either polynomial or linear)

4. c = Ek(m) finding m is hard without k (exponential)

Asymmetric Encryption



RSA - Rivest, Shamir, and Adleman

What is RSA?

- RSA is one of the first public-key (asymmetric) cryptosystems.
- Named after its inventors: Rivest, Shamir, and Adlerman.
- Used for secure data transmission, digital signatures, and key exchange

RSA Key Generation

1. Choose two large prime numbers: p and q.
2. Compute $n = p \times q$ → this becomes the modulus.
3. Compute $z = (p - 1)(q - 1)$ (Euler's totient).
4. Choose e such that $1 < e < z$ and $\gcd(e, z) = 1$.
5. Compute d such that $e \cdot d \equiv 1 \pmod{z}$.

➡️ Public key: (e, n)
➡️ Private key: (d, n)
Both p and q must be kept secret.

Computational complexity

Easy problems with prime numbers

- Generating a prime number p
- Addition, multiplication, exponentiation
- Inversion, solving linear equations

Hard problem with prime numbers

- Factoring primes
  e.g. given $\mathbf{n}$ find $\mathbf{p}$ and $\mathbf{q}$ such that $\mathbf{n} = \mathbf{p} \cdot \mathbf{q}$

RSA - Rivest, Shamir, and Alderman (cont.)

RSA Encryption

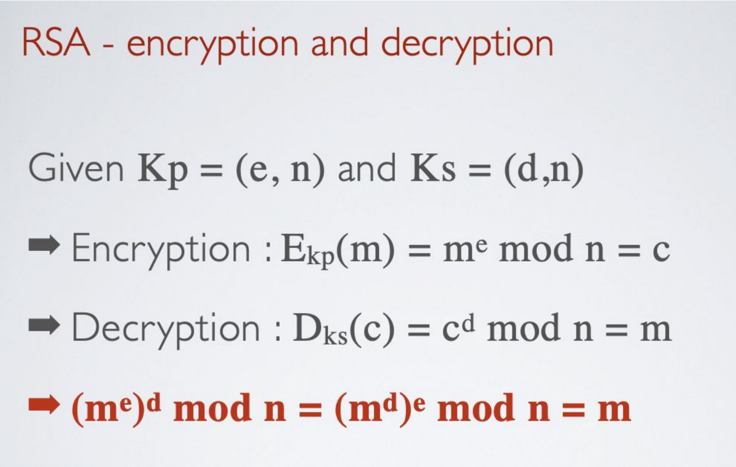
To send a message `m`:

- Use the receiver's public key `(e, n)`
- Compute ciphertext:
  `c = m^e mod n`

RSA Decryption

To decrypt the message:

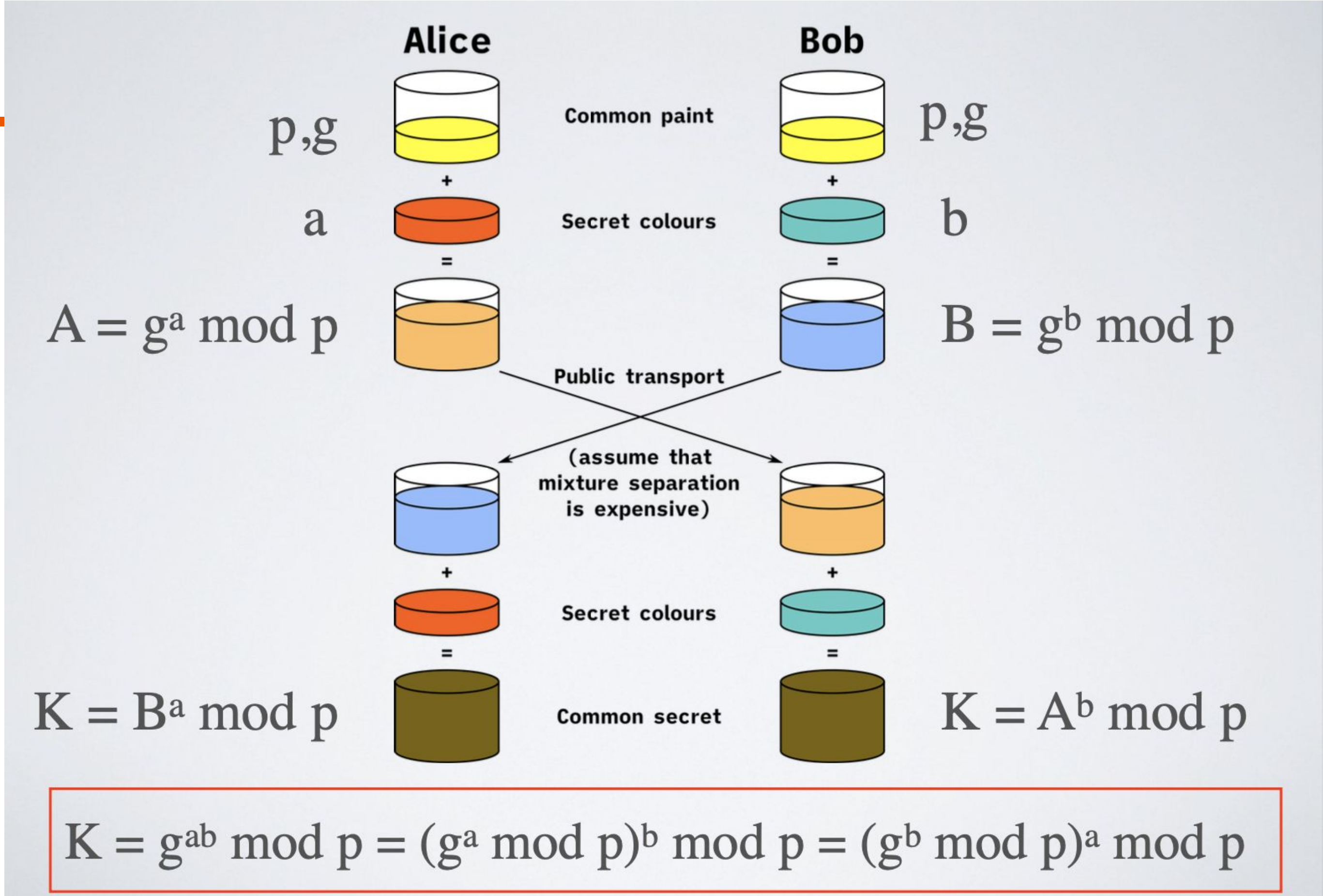
- Use your private key `(d, n)`
- Recover plaintext:
  `m = c^d mod n`



- Easy: multiplying large primes.
- Hard: factoring large numbers.
- RSA security depends on the difficulty of factoring `n` into `p` and `q`.

⚠️ Vulnerable to quantum attacks (e.g., Shor’s Algorithm).

Key Exchange Protocol: Diffie-Hellman-Merkle



So what's the problem

✔️ **The Good News:**

- Diffie-Hellman securely establishes a shared key over a public channel
- Based on the **hardness of the Discrete Logarithm Problem (DLP)**:

  Given $g^x \equiv y \mod p$ it's hard to find x

⚠️ **The Catch:**

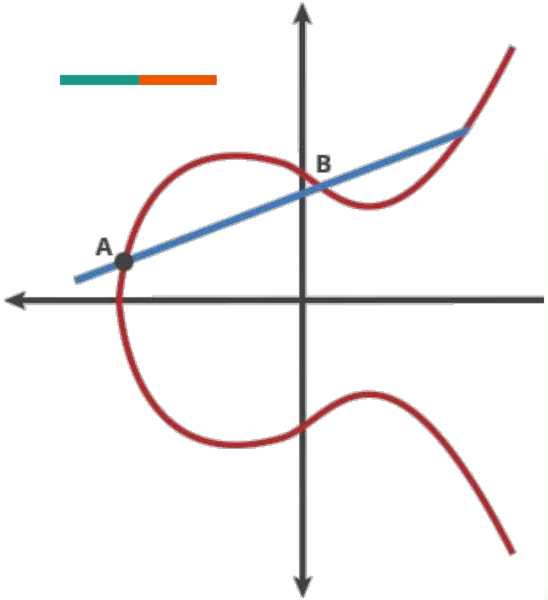
- Classical DLP **relies on large key sizes** (e.g. 2048-bit primes) to stay secure
- But with **faster computers**, smarter **number theory algorithms**, and eventually **quantum computing**...

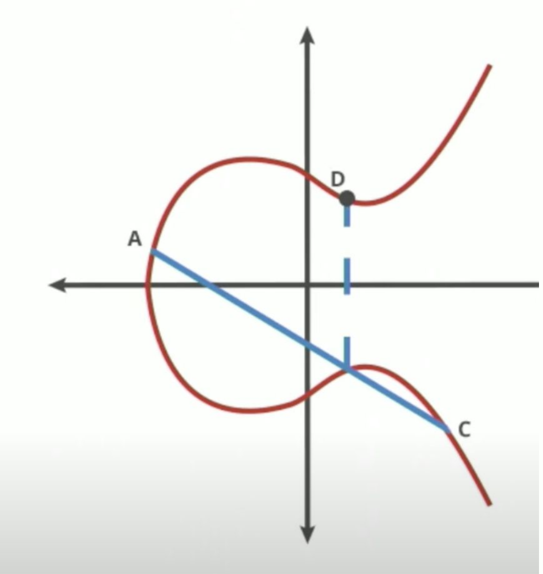
  The DLP is no longer as hard as we'd like.

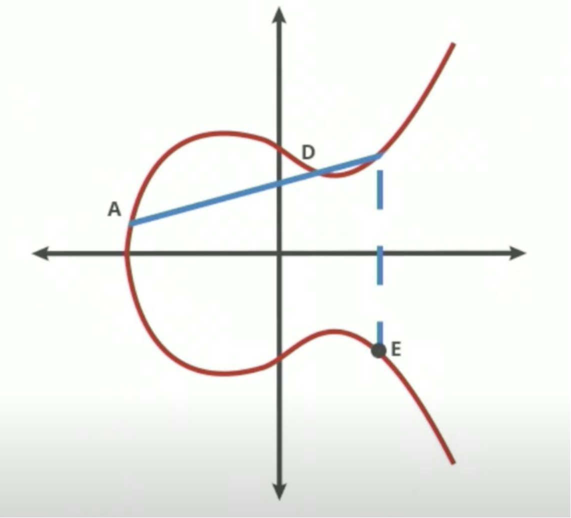
🚪 **This opens the door to:**

- Index Calculus attacks (sub-exponential for DLP)
- Faster brute-force on modern hardware
- Future Shor's Algorithm (breaks DH on a quantum computer in poly-time)

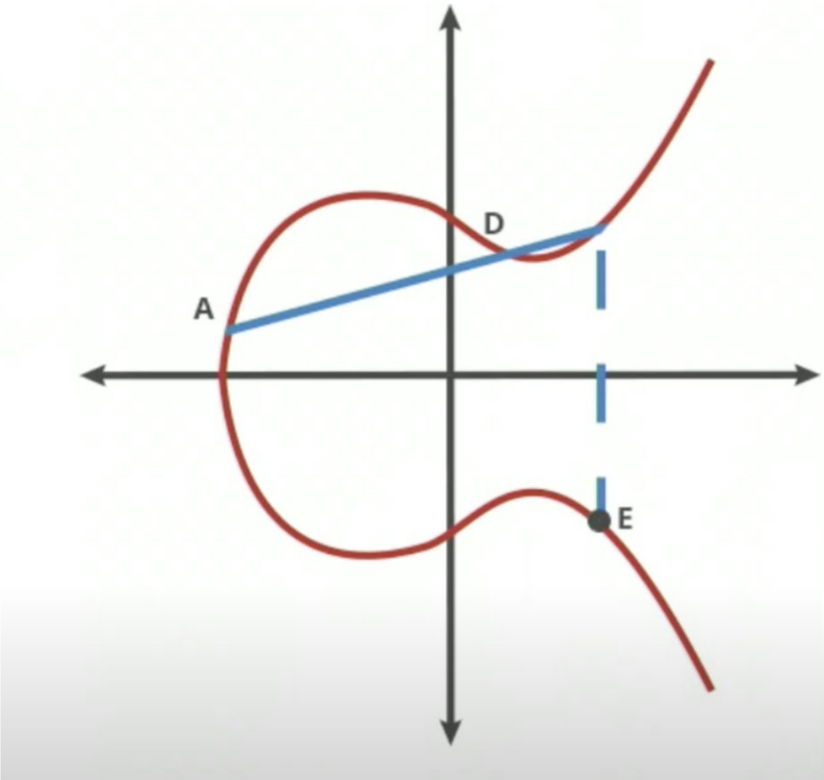
Elliptic Curves Cryptography

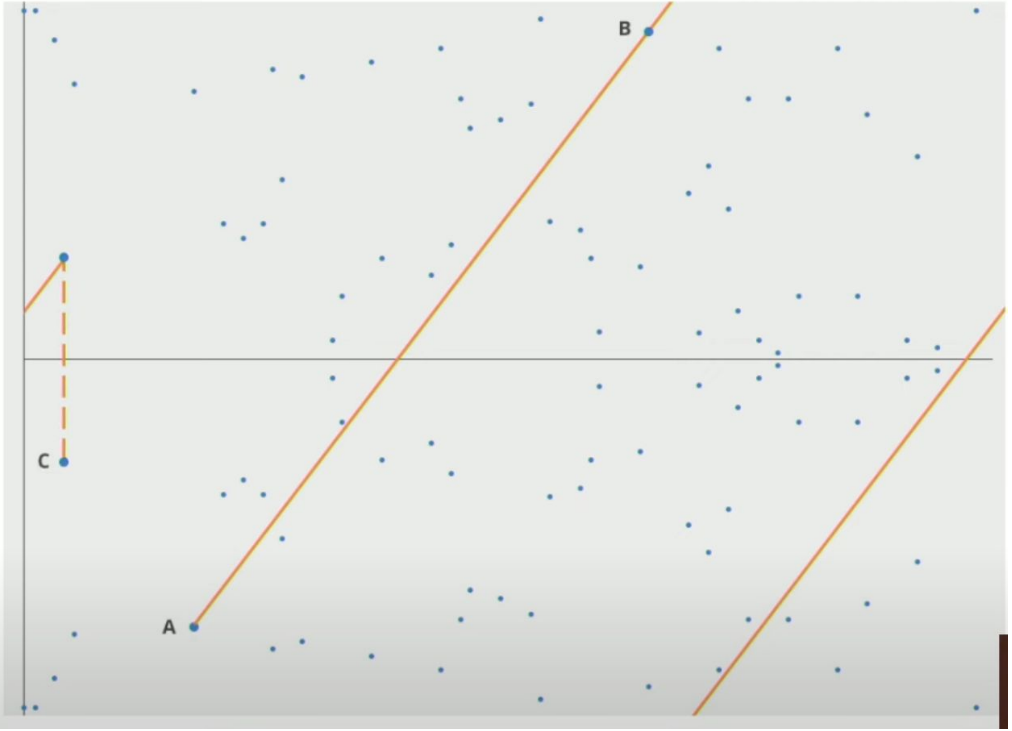




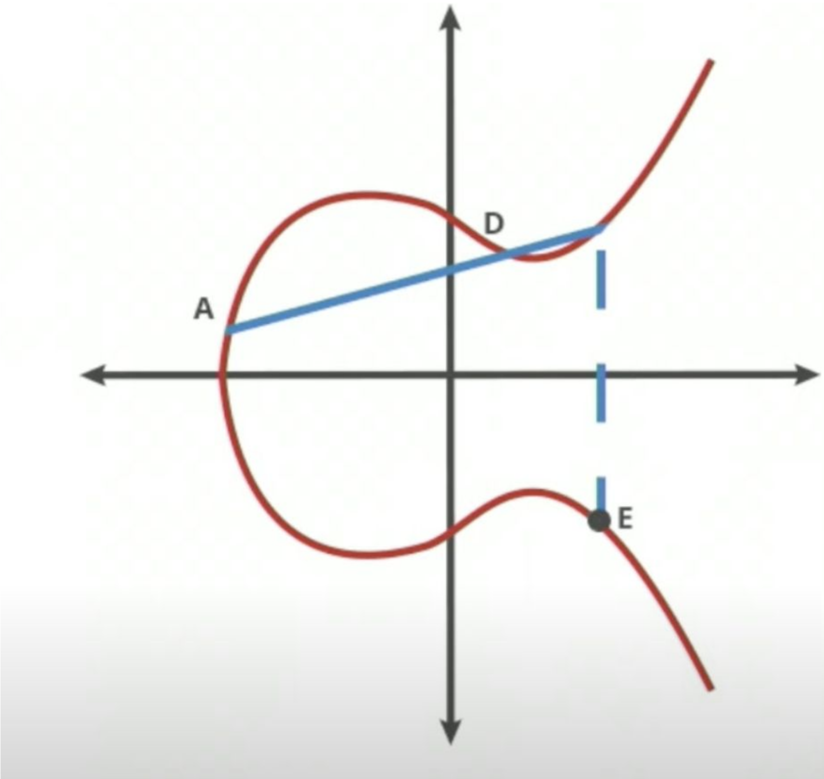


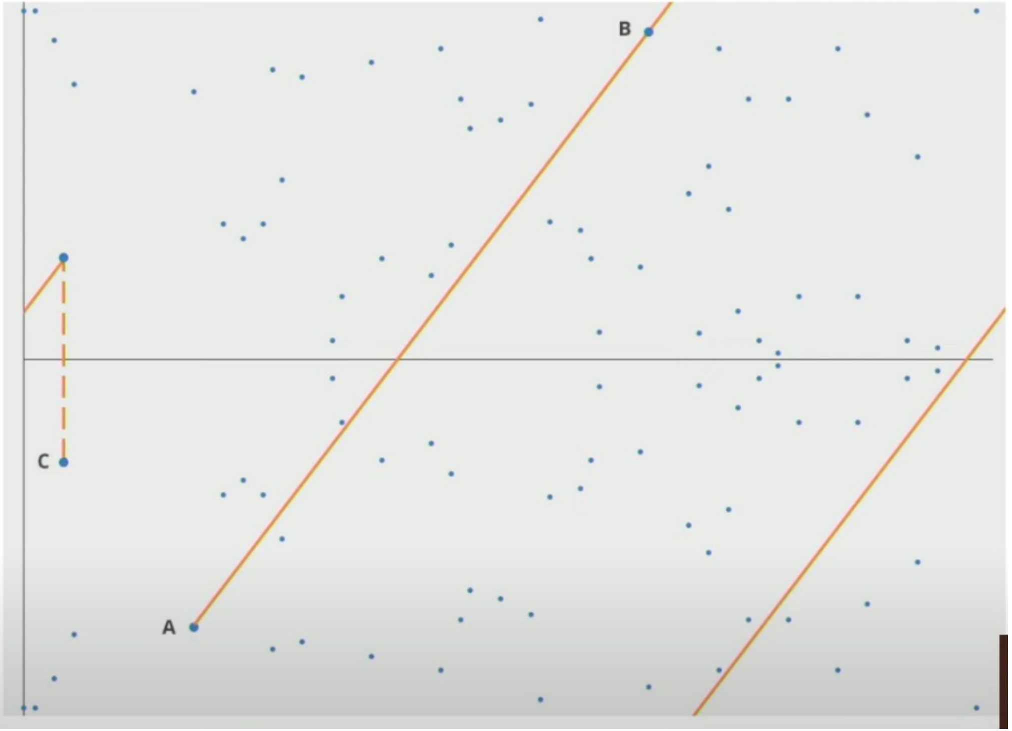
Elliptic Curves Cryptography





Elliptic Curves Cryptography





Elliptic Curves Cryptography

Elliptic Curve Discrete Log Problem

$$nA = E$$

where nA is:

A dot A dot A...

DH	EC
$(G^n)^c = (G^c)^n$	$c(nG) = n(cG)$

Elliptic Curves Diffie-Hellman (ECDH)

DH	EC
Represent secret as a number	*Represent secret as a point*
$G^n \bmod p = H_n$	$nG = H_n$*
m * m * m…	P dot P dot…
2380 bit	*228 bit*

*with a discrete field

ECC - Elliptic Curve Cryptography

Category	Info
Key Size	256 or 448 bits (much smaller than RSA 2048 or 3072)
Speed	~10^6 cycles/operation, Key generation: 1–5 ms, Encryption/Decryption: 1–5 ms
Mathematical Basis	Built on the algebra of **elliptic curves over finite fields**
Use Cases	Secure key exchange (ECDH), digital signatures (ECDSA)
Security Assumption	Hardness of the **Elliptic Curve Discrete Logarithm Problem (ECDLP)**

Why Shor's Algorithm Breaks DH and ECC

Shor's algorithm efficiently solves:

- Integer Factorization Problem (breaks RSA)
- Discrete Logarithm Problem (breaks DH)
- Elliptic Curve Discrete Log Problem (ECDLP) (breaks ECDH)

DH and ECC security rely on the **hardness of DLP**:

- ▸ DH: $g^a \bmod p$ → recover a from $g^a \bmod p$
- ▸ ECDH: $E=n \cdot A$ → recover n from E
- ▸ RSA: $c \equiv m^e \bmod N$ → recover m

Classically: No efficient algorithm to solve these problems.
Quantumly: Shor's algorithm solves all of them in **polynomial time**.

🔓 **Result**: All public-key cryptosystems based on factoring or DLP — including RSA, DH, and ECDH — become insecure.

How Can We Prepare for the Post-Quantum Era?

1. Post-Quantum Cryptography (PQC)

- New classical algorithms based on quantum-resistant math problems
- Works with current internet infrastructure (TLS, VPNs)

2. Quantum Key Distribution (QKD)

- Uses quantum mechanics to securely share keys
- Guarantees security by the laws of physics

Post Quantum Cryptography

What is Post-Quantum Cryptography (PQC)?

🔐 **Definition**:
Cryptographic methods designed to be secure against attacks by both classical and quantum computers.

🚫 **Why it matters**:
Shor's algorithm breaks RSA, DH, ECC. PQC provides quantum-resistant alternatives — without requiring quantum hardware.

💡 **Key idea**:
PQC algorithms are based on hard mathematical problems with *no known efficient quantum solution* (e.g., lattices, codes, hashes, multivariate equations).

📦 **Classical-Compatible**:
Can be implemented with today's computers and integrated into protocols like TLS, SSH, VPNs.

NIST PQC Standardization

🏛️ **NIST PQC Project** (since 2016)

- Global competition to find quantum-safe cryptographic algorithms
- Round 3 results announced July 2022

✅ **Standardized Algorithms**:

- **Kyber** (KEM)
- **Dilithium** (Signature)
- **SPHINCS+** (Signature – hash-based fallback)

🧩 **Ongoing**: NTRU, BIKE, and others under review for future standardization

📦 **Integration Examples**:

- OpenSSL 3.0+
- Google Chrome experiments (Kyber)
- Cloudflare & AWS pilot deployments

Families of PQC Algorithms

Post-Quantum Cryptography relies on hard problems not vulnerable to Shor's algorithm:

- 🧱 **Lattice-based** (e.g., Kyber, Dilithium, NTRU) — 🔥 most promising!
- 📧 **Code-based** (e.g., McEliece) — large keys, used in some legacy systems.
- 🌻 **Multivariate** (e.g., Rainbow) — broken during NIST process.
- 🌀 **Hash-based** (e.g., SPHINCS+) — extremely conservative, provably secure, but slower.

Classical vs. Post-Quantum Algorithms

Classical Algorithm	Broken by Shor?	PQC Replacement
RSA	✅ Yes	Kyber (encryption)
DH (Diffie-Hellman)	✅ Yes	NTRU (key exchange)
ECC (ECDH, ECDSA)	✅ Yes	Dilithium (signatures)

- **Kyber** is a key encapsulation mechanism (KEM) based on lattice problems.
  NTRU is an older lattice-based cryptosystem.
  Dilithium is a digital signature algorithm based on lattice techniques.

Let's Dive Deeper: Lattice-Based Cryptography

🚧 **Why Lattices?**

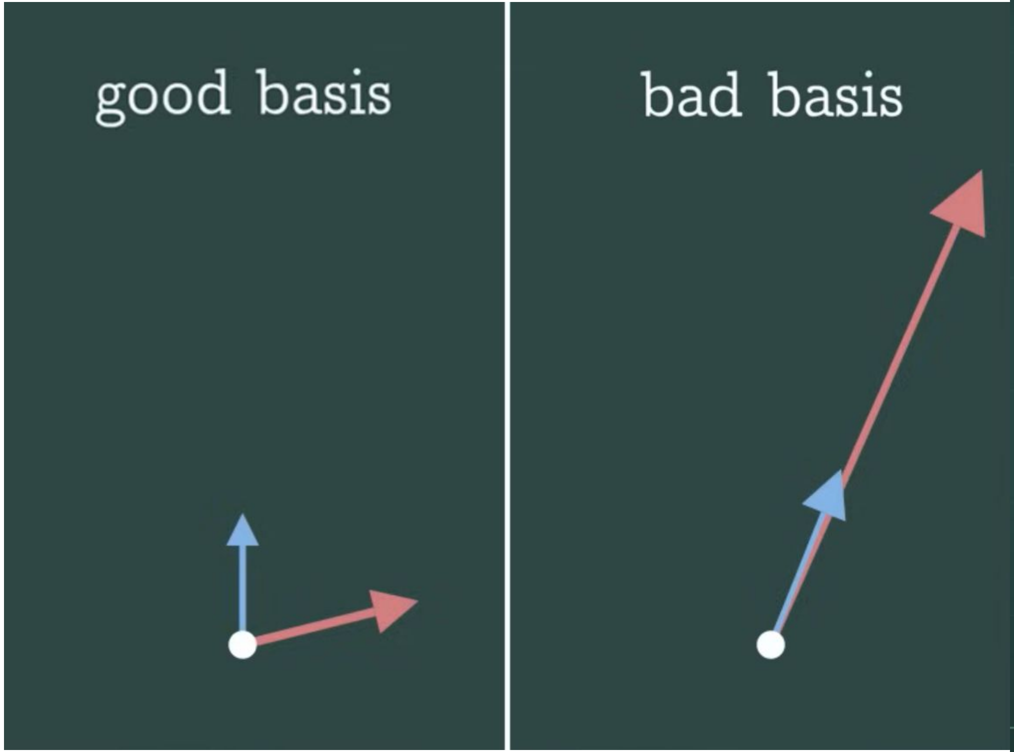
- Flexible
- Efficient
- No known quantum algorithms to break them
- Foundation of NIST winners Kyber and Dilithium

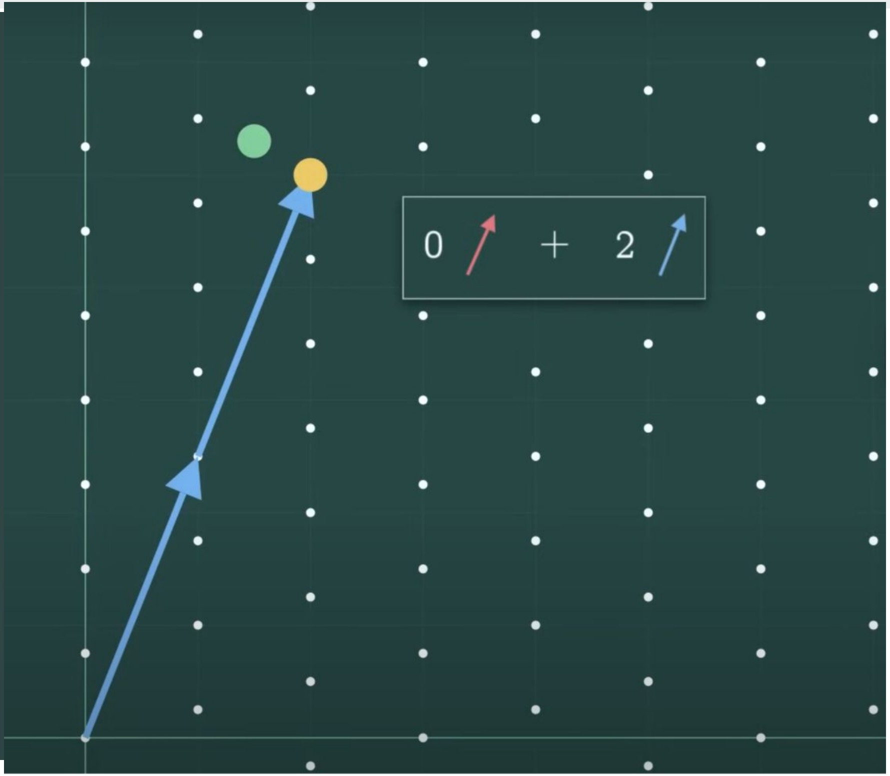
🔍 **Coming up**:

- What are lattices?
- Why are they hard to solve?
- How are they used in PQC?

Lattice Methods in PQC

Good and bad basis - Alice and Bob





The **GGH encryption scheme**
(Goldreich–Goldwasser–Halevi, 1997)

GGH - Broken?

Why it was broken

- **Attack types**: Embedding attacks, Babai rounding attacks.
- **Core issue**: The noise in GGH is small and deterministic, allowing attackers to “peel off” the message without the secret key.
- **Consequence**: Attackers can easily solve a simplified version of the Closest Vector Problem (CVP) and recover the message.

Timeline of attacks

- **1999**: Nguyen showed practical attacks using lattice reduction (LLL) to approximate the good basis.
- **2001**: Hoffstein, Pipher, and Silverman moved to NTRU, which resists these attacks.
- **2009**: Nguyen & Regev reinforced that GGH is insecure for practical parameters.

Lesson learned

- Simply publishing a “bad” lattice basis isn’t enough. The noise must be randomized to hide any structure.
- This led to the development of **Learning With Errors (LWE)** and **Ring-LWE**, which carefully choose noise distributions that are hard to remove.

Why Lattice Problems Are Hard - Summary

- **Exponential time to solve**: Even with quantum algorithms, lattice problems like SVP, CVP, and LWE remain **exponentially hard** to solve in higher dimensions.

- **No known quantum advantage**: Unlike other problems like factoring (used in RSA), **lattice problems** do not show a clear quantum speedup, making them a reliable foundation for post-quantum cryptography.

Why It Matters:

- The hardness of lattice problems provides the security foundation for many post-quantum cryptosystems. Lattice-based schemes like **Kyber**, **Dilithium**, and **NTRU** offer **quantum-resistance**, ensuring they remain secure even in the quantum computing age.

Kyber - Encryption and Key Encapsulation

What is Kyber?

- **Kyber** is a **lattice-based public-key encryption** scheme and **key encapsulation mechanism (KEM)**.
- **Key Encapsulation**:
  - A process where the sender "encapsulates" the secret key (like a symmetric encryption key) in an encrypted form and sends it to the receiver.
  - The receiver **decapsulates** it using their private key to retrieve the shared secret key.

How Kyber Works:

- **Encryption**: Encrypts messages using the **shared secret** and lattice-based hard problems (LWE).
- **Key Encapsulation**: Used for key exchange protocols. The encrypted key (ciphertext) is sent over to the receiver, and they retrieve the secret key by applying their private key.

Why it's Important:

- Kyber is one of the **NIST standardization winners** for **PQC**, chosen for its balance of **security** and **efficiency**.

Dilithium - Digital Signatures

What is Dilithium?

- **Dilithium** is a **lattice-based digital signature** scheme designed for post-quantum security.
- **Digital Signatures**:
  - Used to prove the authenticity of a message or document. The sender signs the message using their private key, and the recipient can verify the signature using the sender's public key.

How Dilithium Works:

- **Signing**: The signer generates a **digital signature** by using their private key and a **lattice-based problem** (learning with errors).
- **Verification**: The recipient verifies the signature using the public key, ensuring that the signature is legitimate and the message hasn't been tampered with.

Why it's Important:

- **Dilithium** is widely used for ensuring **message authenticity** and is one of the winners in the NIST PQC process for **digital signatures**.

NTRU - Key Exchange

What is NTRU?

- **NTRU** is a **lattice-based public-key cryptosystem** that focuses on **key exchange**.
- **Key Exchange**:
  - Allows two parties to securely exchange a secret key over an insecure channel without actually transmitting the key. The exchanged key is then used for symmetric encryption (e.g., AES).

How NTRU Works:

- NTRU uses **polynomials** to form the public and private keys.
- The encryption and decryption rely on **lattice problems** (like finding the closest vector in the lattice).
- The key exchange happens via **secure polynomial operations** that are hard to break even for quantum computers.

Why it's Important:

- **NTRU** is one of the oldest and most studied lattice-based cryptosystems, and it's gaining attention in the post-quantum cryptography world for its **speed** and **security**.

Quantum Key Distribution - QKD

Introduction to Quantum Key Distribution (QKD)

What is QKD?

- **QKD** uses **quantum mechanics** to securely share keys (unlike lattice-based methods which rely on mathematical problems).
- If **Eve** (an eavesdropper) tries to listen in on the communication, it **introduces disturbance** to the system, which can be detected.

How QKD is Different from Lattice-Based Methods:

- Lattice-based cryptosystems (like **Kyber** and **Dilithium**) rely on the **hardness** of lattice problems (e.g., LWE) to secure keys and data.
- **QKD**, on the other hand, leverages the **fundamental principles of quantum mechanics — superposition** and **entanglement** — to ensure **secure key exchange** without relying on hard computational problems.

E91 Protocol (Entanglement-Based QKD)

What is E91?

- The **E91 protocol** is an **entanglement-based Quantum Key Distribution** protocol.
- Unlike BB84, which relies on **single qubit states**, **E91** uses **entangled quantum states**.
- Alice and Bob each receive one half of an **entangled pair** of qubits.
- The key is created by measuring the qubits in randomly chosen bases, and the results are **correlated** if no eavesdropping occurs.

How E91 Works:

- **Entangled pairs**: Alice and Bob share pairs of entangled qubits, meaning their states are instantaneously correlated, no matter the distance between them.
- **Measurements**: Alice and Bob measure their qubits in randomly chosen bases (similar to BB84).
- **Key Generation**: The measurement results will be correlated, and any **basis mismatches** are discarded. The remaining results form their shared secret key.
- **Eavesdropping**: If an eavesdropper, Eve, tries to measure the entangled qubits, she will disturb the entanglement, resulting in detectable errors.

Why It's Important:

- E91 offers a **stronger** level of security because it's based on **quantum entanglement**, which is a **fundamental quantum phenomenon**.
- It provides a **guaranteed correlation** between Alice and Bob's results if no one is eavesdropping.

Real-World Implementations of QKD

China's Micius Satellite (Global QKD):

- China has successfully deployed **global QKD** using its **Micius satellite**, enabling secure key exchange between distant locations, including **intercontinental communication**.
- This is a milestone for quantum communication on a **global scale**, showing the practicality of QKD for **secure long-distance communication**.

Swiss Quantum (Banking):

- **Swiss Quantum** is pioneering the use of QKD for **banking** and **finance**, ensuring that sensitive transactions and communications remain secure in the presence of quantum threats.
- The implementation is focused on integrating QKD into existing **financial infrastructure** for secure transactions and information exchange.

Limitations:

- **Quantum channel required**: QKD systems need a **quantum channel** (e.g., fiber optics or satellite links) for secure transmission.
- **Distance constraints**: Currently, the range for ground-based QKD is limited, though satellite-based solutions (like Micius) offer longer distances.

Comparison & Synthesis (PQC vs. QKD)

Feature	PQC	QKD
Based on	Classical hard problems	Quantum physics
Requires Quantum HW	❌	✅
Ready for web	✅ (TLS 1.3)	❌ (Not scalable yet)
Long-term	🤔 Depends on hardness	✅ Provable security

Why We May Need Both:

- Hybrid systems (PQC + QKD) may be needed to ensure long-term security.
- The future may involve both **quantum-safe cryptography** (PQC) and **quantum communication** (QKD).

Where is the field going?

- **Standardization** of PQC algorithms is underway (NIST process).
- **Quantum internet** and **global QKD networks** are the future of secure communication.