Foundations of Post-Quantum Cryptography
A 3-Hour Introduction for Quantum Technologies Students

Dr. José Ramón Martínez Saavedra
Quside Technologies SL

March 30, 2026

Abstract

The advent of large-scale quantum computers poses an existential threat to modern cryptography, with Shor's algorithm capable of breaking RSA, elliptic curve cryptography, and Diffie-Hellman key exchange—the foundations protecting virtually all digital communications. This lecture provides a comprehensive introduction to post-quantum cryptography for physicists, bridging theoretical quantum mechanics with practical cryptographic applications. We begin by examining how current cryptographic systems work and why they fail against quantum attack, providing precise analysis of the quantum resources required. We then explore the mathematical structures believed to resist quantum algorithms, focusing on lattice-based cryptography and the Learning With Errors problem. The lecture addresses the urgent "harvest now, decrypt later" threat, where adversaries store encrypted data today for future quantum decryption, making immediate action necessary. By connecting quantum physics intuition with cryptographic security, students will understand both the theoretical foundations and practical implications of the transition to quantum-resistant cryptography. The material prepares physicists to contribute to this critical intersection of quantum technology and information security, whether through research, implementation, or policy development.

Contents

1 Introduction

1.1 The Quantum Threat to Cryptography

In 1994, Peter Shor published an algorithm that efficiently factors large integers and computes discrete logarithms—but only on a computer that doesn't yet exist. This peculiar situation, where we know exactly how to break the cryptographic systems protecting trillions of dollars in global commerce using hardware we cannot yet build, represents one of the most fascinating challenges in modern computer science.

The implications are staggering. Every secure communication you make—from credit card transactions to WhatsApp messages—relies on mathematical problems that quantum computers will solve efficiently. RSA encryption, elliptic curve cryptography, and Diffie-Hellman key exchange all fail catastrophically against a sufficiently large quantum computer. This isn't speculation or theoretical concern; the mathematics is proven and unambiguous. The only questions are *when* such computers will exist and *how* we prepare our infrastructure for their arrival.

Current State of Quantum Computing: As of 2025, the gap between current quantum computers and cryptographically relevant machines remains substantial but is closing. Google's Sycamore processor achieved quantum supremacy in 2019 with 53 qubits, while IBM's current roadmap targets 100,000-qubit systems by 2033. However, breaking RSA-2048 requires approximately 20 million noisy qubits or 4,000 perfect logical qubits—a gap measured not in years but in orders of magnitude. Current systems struggle with coherence times measured in microseconds and error rates around 10^{-3}, while cryptographically relevant computations need error rates below 10^{-12}. Yet the exponential improvement in quantum hardware over the past decade suggests this gap will close within 10-20 years.

This lecture addresses three fundamental questions that will shape the next decade of cryptographic practice:

1. **How does cryptography actually work in practice?**

   Before we can understand what breaks, we must understand what exists. How do the cryptographic primitives you use every day—encryption, digital signatures, and key exchange—actually protect information? What mathematical foundations underpin the security of literally every digital transaction on Earth?

2. **Why exactly do quantum computers break current cryptography?**

   Shor's algorithm doesn't try all possible keys; that would be mundane. Instead, it exploits the hidden periodic structure in the mathematical problems underlying RSA and elliptic curve cryptography. We'll see exactly how quantum Fourier transforms turn intractable problems into polynomial-time computations, and why this quantum advantage is so devastating.

3. **What mathematical structures can replace our current foundations?**

   The solution isn't to abandon public-key cryptography but to rebuild it on quantum-resistant foundations. We'll explore the mathematical problems that appear to resist quantum attack—from finding short vectors in high-dimensional lattices to solving systems of multivariate polynomials—and understand why we believe they remain hard even for quantum adversaries.

As physicists, you have a unique perspective on this challenge. You understand quantum mechanics not as mysterious magic but as a computational resource with specific capabilities and limitations. You're comfortable with high-dimensional spaces, basis transformations, and the interplay between discrete and continuous mathematics. This background will prove invaluable as we explore how lattice-based cryptography—deeply connected to crystallography and solid-state physics—has emerged as our best defense against the quantum threat.

The "Harvest Now, Decrypt Later" Timeline: The threat is not hypothetical but actively unfolding:

- **Today (2025):** Adversaries are harvesting encrypted data, storing petabytes of communications, financial records, and state secrets
- **2030-2035:** Early cryptographically relevant quantum computers may emerge in state laboratories
- **2035-2040:** Stored data from 2025 becomes readable; secrets with 10-15 year sensitivity are exposed
- **2040+:** Quantum computers become accessible to well-funded criminal organizations

This timeline means that data encrypted today needs protection against computers that won't exist for years—a unique challenge in the history of cryptography. Organizations must migrate to post-quantum cryptography not when quantum computers arrive, but years before, making this an urgent present concern rather than a future worry.

1.2 Prerequisites and Assumptions

This lecture is designed specifically for students in the Quantum Technologies master's program. I will assume certain theoretical knowledge while building practical understanding from the ground up.

What I assume you already know:

- **Quantum Mechanics:** You're comfortable with quantum states, superposition, entanglement, and quantum measurements. You understand the quantum Fourier transform and have seen basic quantum algorithms. When I mention computational basis states or unitary operations, you don't need explanation.
- **Linear Algebra:** You can work with vector spaces, basis transformations, eigenvalues, and matrix operations without difficulty. Concepts like orthogonality, norm, and linear independence are second nature. You're comfortable in both finite-dimensional and infinite-dimensional spaces.
- **Basic Number Theory:** You understand modular arithmetic, greatest common divisors, and prime numbers. You've seen groups, rings, and fields in at least an introductory context. While we'll use more advanced concepts, we'll build them as needed.
- **Quantum Key Distribution:** As students in a QKD course, you understand the BB84 protocol, the principles of information-theoretic security, and the physical requirements for quantum communication. This background will help us contrast QKD with post-quantum cryptography.
- **Computational Complexity:** You understand big-O notation and the difference between polynomial and exponential time. You've heard of P vs NP, even if you haven't studied it deeply.

What I do NOT assume you know:

- **Practical Cryptographic Deployment:** How TLS actually works, what happens during a certificate verification, or how cryptographic protocols are implemented in real systems. We'll build this knowledge from scratch.

- **Cryptographic Primitives:** The detailed workings of RSA, elliptic curve cryptography, or digital signature algorithms. While you may have heard these terms, we'll develop their mathematical foundations carefully.
- **Security Definitions:** Formal notions like semantic security, chosen-ciphertext attacks, or security reductions. We'll introduce these concepts as needed, focusing on intuition over formalism.
- **Implementation Details:** Side-channel attacks, constant-time programming, or cryptographic engineering. We'll touch on these for completeness but won't assume prior exposure.
- **Lattice Theory:** Despite its connections to crystallography, we won't assume you've studied lattices in the cryptographic context. We'll build this theory from your physics intuition.

Mathematical maturity: I expect you can follow mathematical proofs, understand algorithmic descriptions, and translate between different representations of the same concept. When I present a cryptographic protocol, you should be able to analyze its steps logically. When I describe a mathematical problem, you should be able to work with small concrete examples.

Bridging the gap: Throughout this lecture, I'll use analogies from physics when introducing cryptographic concepts. For instance, we'll think of computational hardness like energy barriers, and cryptographic security like thermodynamic irreversibility. Your intuition from statistical mechanics and quantum physics will often provide the right framework for understanding cryptographic ideas.

Quick review resources: If you feel uncertain about any prerequisite:

- For modular arithmetic: We'll review key concepts inline when first used
- For complexity basics: Section 2.1 provides a brief refresher
- For linear algebra notation: Our conventions are established as we introduce each concept

Remember: cryptography is where physics, mathematics, and computer science meet. Your physics background provides unique insights—embrace them.

1.3 Learning Objectives

By the end of this three-hour lecture, you will have acquired both theoretical understanding and practical insights necessary to navigate the transition to post-quantum cryptography. These objectives are designed to prepare you for research, implementation, or policy work in the quantum-safe cryptography era.

Theoretical Understanding: After this lecture, you will be able to:

1. **Analyze cryptographic vulnerability to quantum attack**
    - Explain precisely how Shor's algorithm breaks RSA and elliptic curve cryptography
    - Calculate the quantum resources needed to break specific key sizes
    - Distinguish between quantum-vulnerable and quantum-resistant mathematical problems
    - Evaluate whether a given cryptographic scheme is threatened by known quantum algorithms
2. **Understand the mathematical foundations of post-quantum cryptography**

- Define lattice problems (SVP, CVP, LWE) and explain their computational hardness
- Describe how Learning With Errors provides a foundation for encryption
- Compare the security assumptions of different PQC families (lattice, code, hash-based)
- Analyze the trade-offs between Ring-LWE and standard LWE

3. **Connect quantum and classical security models**

    - Contrast information-theoretic (QKD) vs computational (PQC) security
    - Explain why both QKD and PQC are necessary in a quantum world
    - Identify scenarios where each approach is most appropriate
    - Design hybrid protocols combining multiple security approaches

Practical Skills: After this lecture, you will be able to:

1. **Evaluate real-world cryptographic systems**

    - Identify the cryptographic primitives used in common protocols (TLS, Signal, Bitcoin)
    - Assess the quantum vulnerability of existing systems
    - Calculate the impact of PQC migration on performance and bandwidth
    - Read and understand NIST PQC standardization documents

2. **Work with post-quantum algorithms**

    - Implement toy examples of LWE encryption
    - Compare key sizes and performance between classical and PQC algorithms
    - Choose appropriate PQC algorithms for specific use cases
    - Understand parameter selection for security levels

3. **Plan for cryptographic transitions**

    - Develop a migration strategy for a hypothetical organization
    - Identify the challenges in deploying hybrid classical-PQC modes
    - Recognize implementation pitfalls (side-channels, randomness requirements)
    - Assess the urgency of migration for different threat models

Critical Analysis: After this lecture, you will be able to:

1. **Evaluate cryptographic claims critically**

    - Distinguish between proven security and heuristic arguments
    - Identify potential weaknesses in proposed PQC schemes
    - Understand why some PQC candidates (Rainbow, SIKE) were broken
    - Assess the maturity and risk level of different approaches

2. **Connect cryptography to broader quantum technologies**

    - Identify research opportunities at the intersection of physics and PQC
    - Propose physical implementations or attacks on PQC systems
    - Understand how quantum sensing might impact cryptographic security

- Envision the long-term landscape of quantum-era security

Connecting to Your Future: These objectives prepare you for several career paths:

- **Quantum technology development:** Understanding both what quantum computers threaten and what they cannot break
- **Security research:** Contributing to cryptanalysis or development of new quantum-resistant schemes
- **Systems engineering:** Implementing quantum-safe solutions in real products
- **Policy and standards:** Informing decisions about cryptographic transitions

Self-Assessment: Throughout the lecture, I'll provide concrete examples and exercises. If you can work through these independently by the end, you've achieved our learning objectives. Don't worry if everything isn't clear immediately—cryptography is subtle, and understanding deepens with practice.

2 Classical Cryptography in Practice

2.1 Real-World Cryptographic Infrastructure

2.1.1 The Invisible Infrastructure

[fragile] **Live Demonstration** Let me show you something. [Open browser, navigate to `https://github.com`]

Watch the padlock icon. In the 400 milliseconds it took this page to load, your browser:

- Verified 3 digital signatures in a certificate chain
- Performed an elliptic curve Diffie-Hellman key exchange
- Negotiated a shared secret using ECDHE with curve X25519
- Derived 6 different cryptographic keys from that secret
- Started encrypting data with AES-256-GCM

[Open developer tools → Security tab → View Certificate]

Every single HTTPS connection performs this cryptographic dance. With 100 billion HTTPS connections made daily, that's approximately **300 billion digital signatures verified every day**—more than 40 signatures for every person on Earth, every single day.

This is the invisible infrastructure of the digital age. Cryptography isn't an obscure mathematical curiosity—it's the foundation enabling every aspect of modern digital life. Yet it operates so seamlessly that most people never notice its existence until something goes wrong.

Consider what happens in a single minute on the internet:

- **500 hours** of video uploaded to YouTube—each chunk encrypted with AES
- **200 million** emails sent—using TLS, S/MIME, or PGP
- **69 million** WhatsApp messages—each using the Signal Protocol with five layers of encryption
- **150,000** financial transactions—protected by multiple cryptographic protocols
- **4 million** Google searches—each over an encrypted connection

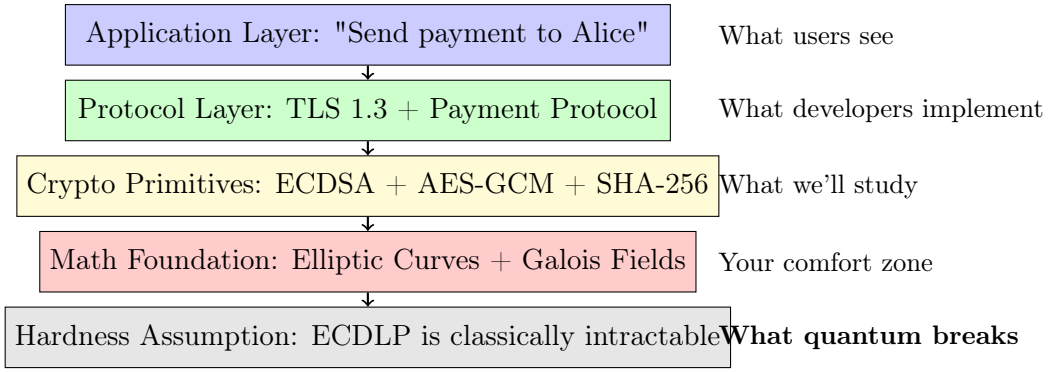


Figure 1: The cryptographic stack: From user action to mathematical foundation

Behind each of these actions lies a complex stack of cryptographic protocols:
The beautiful complexity extends beyond the obvious applications. Cryptography secures:

- **Your car:** Keyless entry uses rolling codes with AES encryption
- **Your passport:** Contains a cryptographic chip performing challenge-response authentication
- **Your credit card:** EMV chips perform dynamic signature generation for each transaction
- **Your phone's SIM:** Authenticates to cell towers using the MILENAGE algorithm
- **Your smart home:** IoT devices use DTLS (Datagram TLS) for secure communication
- **Software updates:** Every app update is cryptographically signed to prevent tampering

Example 1 (The Cryptographic Cascade)**.** *Let's trace a simple action—unlocking your phone with your fingerprint:*

1. *Fingerprint sensor captures biometric data*
2. *Data is hashed using SHA-256 (not stored in plain form)*
3. *Hash is compared using a secure enclave processor*
4. *Upon match, the enclave releases an AES-256 key*
5. *This key decrypts your phone's file system keys*
6. *Those keys decrypt your actual data*

Five cryptographic operations just to unlock your phone!

The infrastructure is so pervasive that its sudden failure would be catastrophic. If all cryptography stopped working instantly:

- All e-commerce would halt (no secure payments)
- Email would become postcards (no privacy)
- Software couldn't be trusted (no signatures)
- Digital identity would collapse (no authentication)
- Classified information would be exposed (no confidentiality)

This is why the quantum threat matters. We're not protecting some abstract mathematical constructs—we're protecting the fundamental trust layer of human civilization in the 21st century.

Remark 1 (For the skeptical)**.** *You might think: "Surely not* everything *uses cryptography." Try this experiment: Use Wireshark to monitor your network traffic for 60 seconds while doing nothing. You'll see dozens of encrypted connections—background app updates, cloud syncs, telemetry data, keepalive packets. The invisible infrastructure never sleeps.*

In the next sections, we'll dissect this infrastructure piece by piece, understanding not just what each component does, but why it's vulnerable to quantum attack and how we can rebuild it to be quantum-resistant.

2.1.2 The Three Pillars of Cryptography

Most people think cryptography equals secrecy—hiding messages from prying eyes. This mental model is dangerously incomplete. Modern cryptography rests on three equally critical pillars, and the failure of any one can be catastrophic.

Definition 1 (The Cryptographic Triad)**.** *Complete cryptographic security requires:*

1. ***Confidentiality:*** *Only authorized parties can read the data*
2. ***Integrity:*** *Any tampering with data is detectable*
3. ***Authentication:*** *You can verify who you're communicating with*

Let's see why each pillar is essential through real-world failures:

Pillar 1: Confidentiality (Encryption) This is what most people think of as "cryptography"—transforming plaintext into ciphertext that only authorized parties can decrypt.

- **Primitives:** AES (symmetric), RSA encryption (asymmetric), ElGamal
- **Purpose:** Prevent unauthorized reading of data
- **Without it:** Every message is a postcard, readable by anyone who intercepts it

Example 2 (Confidentiality Failure)**.** *In 2013, Edward Snowden revealed that the NSA was collecting unencrypted data from Google's internal networks. Between data centers, Google relied on physical security rather than encryption. Result: Massive confidentiality breach despite secure facilities.*

Pillar 2: Integrity (Hash Functions and MACs) Integrity ensures that any modification to data is detectable. This is *not* about preventing modification—it's about detecting it.

- **Primitives:** SHA-256 (hash), HMAC (keyed hash), GHASH (in AES-GCM)
- **Purpose:** Detect any bit flip, insertion, deletion, or reordering
- **Without it:** Attackers can modify messages undetected

Example 3 (Integrity Failure)**.** *In 2008, researchers demonstrated collision attacks on MD5 certificates. They created a rogue Certificate Authority certificate that had the same hash as a legitimate website certificate. This allowed them to impersonate* any *website because the integrity check (MD5) was broken.*

Pillar 3: Authentication (Digital Signatures) Authentication proves the identity of the sender and is provided by digital signatures—the digital equivalent of a handwritten signature but unforgeable.

- **Primitives:** RSA signatures, ECDSA, EdDSA
- **Purpose:** Prove who sent a message and that they approve its contents
- **Without it:** Anyone can impersonate anyone else

Example 4 (Authentication Failure)**.** *In 2011, DigiNotar, a Dutch Certificate Authority, was compromised. Attackers issued fraudulent certificates for major domains including *.google.com. Without proper authentication, users' browsers trusted fake websites. The company went bankrupt within months, and the Dutch government fell into crisis as DigiNotar secured government sites.*

The Interdependence of the Pillars Here's the critical insight: **these pillars are not independent**. Encryption without authentication is like having a secure phone line to a stranger—you don't know who you're talking to. Authentication without encryption means everyone can verify your signature but also read your messages.

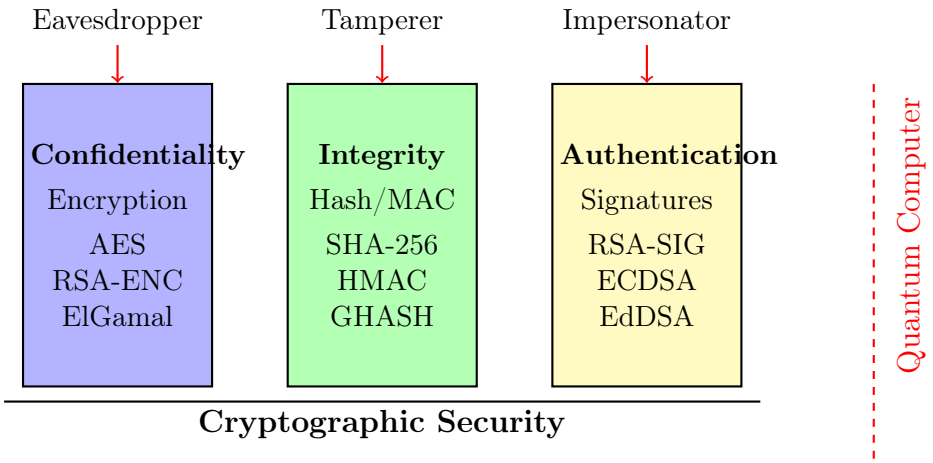


Figure 2: The three pillars of cryptography and their attackers

Real Protocol Example: Secure Email Let's see how all three pillars work together in PGP/GPG email:

1. **Authentication:** Alice signs her email with her private key
2. **Integrity:** The signature includes a hash of the message
3. **Confidentiality:** The signed message is encrypted with Bob's public key
4. Bob receives the encrypted package and:
   - Decrypts with his private key (confidentiality)
   - Verifies Alice's signature (authentication)
   - Checks the hash matches (integrity)

Remove any step and security collapses:

- No encryption? Everyone reads Alice's mail

- No signature? Eve can send mail claiming to be Alice
- No integrity check? Mallory can modify the message in transit

The Quantum Threat to All Three Pillars Here's what makes quantum computers so devastating: they don't just threaten one pillar—they demolish two completely and weaken the third:

- **Confidentiality:** Shor's algorithm breaks RSA encryption, ElGamal, and ECC
- **Authentication:** The same algorithm breaks RSA signatures, ECDSA, and DSA
- **Integrity:** Grover's algorithm gives quadratic speedup for finding hash collisions

This is why post-quantum cryptography must address all three pillars. It's not enough to have quantum-resistant encryption if signatures remain vulnerable—the entire security architecture would still collapse.

Remark 2 (A Common Misconception)**.** *Students often ask: "Why not just use AES for everything? It's quantum-resistant!" Because AES (symmetric encryption) only provides confidentiality, and only if you already share a key. It cannot provide authentication (no public/private key pairs) and provides integrity only in special modes like GCM. The three pillars require different mathematical structures.*

Understanding these three pillars is crucial for the rest of our journey. Every cryptographic protocol we examine will use all three, and every quantum threat we discuss will target one or more of these fundamental security properties.

2.1.3 The Two-Key Revolution

Before 1976, all cryptography faced a fundamental paradox: to communicate securely, you first had to communicate securely. This circular dependency meant that every encrypted communication required a prior secure exchange of secret keys—a logistical nightmare that limited cryptography to governments, militaries, and large corporations with trusted courier networks.

The Pre-Digital Key Distribution Problem Consider the absurd logistics of symmetric-key cryptography at scale:

Example 5 (The Key Distribution Nightmare)**.** *Imagine 1,000 people wanting to communicate securely with each other:*

- *Each pair needs a unique shared key:* $\binom{1000}{2} = 499,500$ *keys total*
- *Each person must securely store 999 keys*
- *Adding one new person requires distributing 1,000 new keys*
- *Key compromise affects only one communication pair (good), but requires physical key redistribution (bad)*

Historical solutions were expensive and limited:

- **Diplomatic bags:** Physical couriers with armed protection
- **Code books:** Pre-distributed keys (vulnerable if captured)
- **One-time pads:** Perfect security but requires key length = message length

- **Key distribution centers:** Single point of failure and trust

The Internet was emerging in the 1970s, but secure communication seemed impossible. How could strangers establish secure channels without meeting? How could e-commerce exist if every customer needed pre-shared keys with every merchant?

The "Impossible" Made Possible In 1976, Whitfield Diffie and Martin Hellman published "New Directions in Cryptography," solving what experts considered impossible. Their insight was profound yet simple: **separate the key into two parts with different capabilities**.

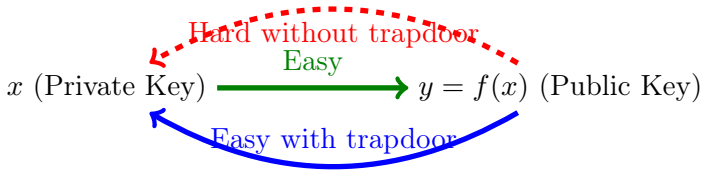
> "We stand today on the brink of a revolution in cryptography. The development of cheap digital hardware has freed it from the design limitations of mechanical computing and brought the cost of high-grade cryptographic devices down to where they can be used in such commercial applications as remote cash dispensers and computer terminals." —Diffie & Hellman, 1976

They were spectacularly right. But even they underestimated the impact—they thought cryptography would be important for "remote cash dispensers," not that it would underpin all human digital interaction.

The Mathematical Magic: One-Way Functions with Trapdoors The key insight is that certain mathematical operations are easy in one direction but hard to reverse—unless you have special information (the "trapdoor").

Definition 2 (One-Way Trapdoor Function)**.** *A function $f : X \to Y$ where:*

1. ***Easy forward:*** *Given x, computing $f(x)$ is efficient*
2. ***Hard backward:*** *Given y, finding x where $f(x) = y$ is computationally infeasible*
3. ***Trapdoor:*** *With special information t, finding x from y becomes efficient*



RSA: d (private)	**RSA:** (n, e) (public)
DH: a (private)	**DH:** $g^a \bmod p$ (public)
ECC: k (private)	**ECC:** kG (public)

Figure 3: The one-way trapdoor principle enabling public key cryptography

The Diffie-Hellman Key Exchange: The First Miracle The first public key protocol wasn't encryption—it was key agreement. Two strangers can establish a shared secret over an insecure channel:

The eavesdropper sees g, p, g^a, and g^b but cannot compute g^{ab} without solving the discrete logarithm problem!

Algorithm 1 Diffie-Hellman Key Exchange

Public Parameters: Prime p, generator g

Alice:

1. Choose secret $a \in \{2, ..., p-2\}$
2. Compute $A = g^a \bmod p$
3. Send A to Bob

Bob:

1. Choose secret $b \in \{2, ..., p-2\}$
2. Compute $B = g^b \bmod p$
3. Send B to Alice

Shared Secret:

- Alice computes: $K = B^a = g^{ab} \bmod p$
- Bob computes: $K = A^b = g^{ab} \bmod p$

Property	**Symmetric Cryptography**	**Asymmetric Cryptography**
Keys	Single shared secret	Public/private key pair
Speed	Very fast (Gbps)	Slow (Mbps)
Key size	Small (128-256 bits)	Large (2048-4096 bits)
Use case	Bulk data encryption	Key exchange, signatures
Examples	AES, ChaCha20	RSA, ECDH, ECDSA
Quantum threat	Grover (manageable)	Shor (catastrophic)

Table 1: Complementary strengths of symmetric and asymmetric cryptography

Symmetric vs Asymmetric: A Perfect Partnership The revolution created two complementary cryptographic worlds:

This led to a beautiful division of labor:

- **Asymmetric:** Establish initial trust and exchange keys
- **Symmetric:** Encrypt actual data efficiently

The Hybrid Approach: Best of Both Worlds Modern protocols universally use hybrid cryptography. Here's how TLS 1.3 works:

1. **Authentication:** Server proves identity with certificate (digital signature)
2. **Key Exchange:** ECDHE establishes shared secret (asymmetric)
3. **Key Derivation:** Shared secret generates multiple symmetric keys
4. **Bulk Encryption:** All data encrypted with AES-GCM (symmetric)

Listing 1: Simplified Hybrid Encryption

```
def hybrid_encrypt(message, recipient_public_key):
```

```
    # Generate ephemeral symmetric key
    symmetric_key = generate_random_key(256)  # 256 bits

    # Encrypt message with fast symmetric crypto
    ciphertext = AES_encrypt(message, symmetric_key)

    # Encrypt symmetric key with recipient's public key
    encrypted_key = RSA_encrypt(symmetric_key, recipient_public_key)

    return (encrypted_key, ciphertext)

def hybrid_decrypt(encrypted_key, ciphertext, private_key):
    # Decrypt symmetric key with private key
    symmetric_key = RSA_decrypt(encrypted_key, private_key)

    # Decrypt message with symmetric key
    message = AES_decrypt(ciphertext, symmetric_key)

    return message
```

This hybrid approach is so fundamental that breaking asymmetric crypto breaks everything, even though symmetric crypto does the actual encryption!

Why This Revolution is Quantum-Vulnerable The tragic irony is that the mathematical structures enabling public key cryptography—discrete logarithms and factoring—have hidden periodic structures that quantum computers exploit:

- **Factoring (RSA):** Finding p, q from $n = pq$ reduces to period-finding
- **Discrete Log (DH, DSA, ECDSA):** Finding x from g^x also reduces to period-finding
- **Shor's Algorithm:** Quantum Fourier Transform finds periods exponentially fast

This means the entire public key revolution—forty years of deployed infrastructure—collapses against quantum computers. We don't go back to symmetric-only cryptography (that would be impossible at Internet scale). Instead, we must find new mathematical one-way trapdoor functions that resist quantum attack.

Remark 3 (Historical Irony)**.** *The NSA knew about public key cryptography before 1976 (classified as "non-secret encryption"), but didn't pursue it because they thought key distribution was a solved problem for government use. Diffie and Hellman, outsiders to the cryptographic establishment, revolutionized the field precisely because they saw the civilian need for scalable key distribution. Today's post-quantum transition similarly requires thinking beyond established patterns.*

The two-key revolution made the Internet possible. Every secure connection, digital signature, and cryptocurrency transaction depends on the separation of public and private keys. As we'll see, post-quantum cryptography must preserve this revolutionary capability while replacing its vulnerable mathematical foundations.

2.1.4 Public Key Infrastructure (PKI)

Public key cryptography solved key distribution, but created a new problem: how do you know a public key actually belongs to who claims to own it? If you receive a public key claiming to

be "Bank of America," how can you verify it's genuine? Without solving this authentication problem, attackers could simply publish their own keys under any name. The solution—Public Key Infrastructure—has become the most successful distributed system in human history, though most people have never heard of it.

The Authentication Problem at Scale Consider the challenge: billions of users need to verify the authenticity of millions of websites, without any prior relationship. The naive solutions fail immediately:

- **Manual verification:** "Call the bank to verify their key" —impossible at scale
- **Central directory:** Single point of failure, massive target
- **Web of trust:** Works for small communities, fails at Internet scale
- **Trust on first use:** Vulnerable to initial impersonation

The Elegant Solution: Hierarchical Trust PKI solves this through a hierarchy of trust, similar to how government IDs work. You trust your passport because it's issued by your government; you trust a website's certificate because it's signed by a Certificate Authority (CA).

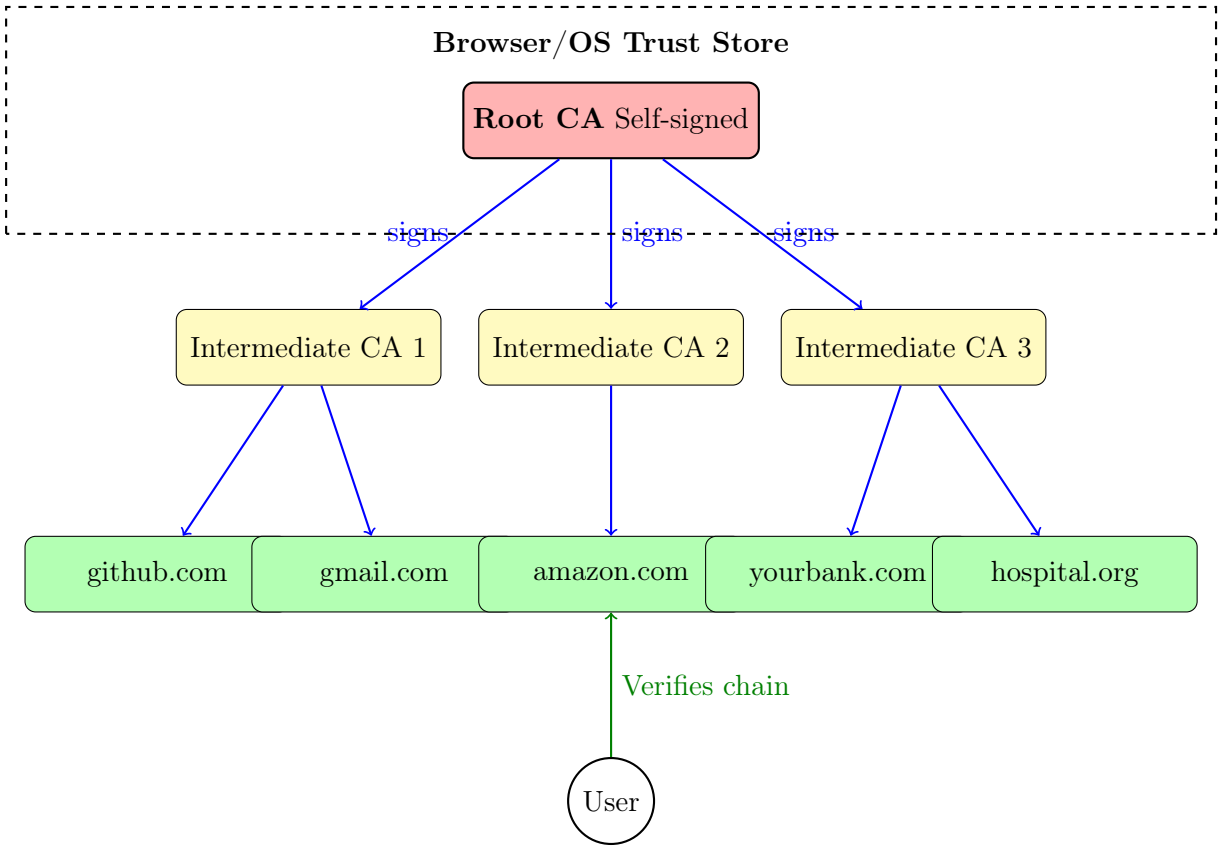


Figure 4: PKI hierarchy: From root trust to website authentication

Certificate Anatomy Let's examine a real certificate to understand what's inside:

Listing 2: Examining a Real Certificate

```
$ openssl s_client -connect github.com:443 -showcerts | \
```

```
openssl x509 -text -noout

Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            0c:1f:cb:18:45:0f:7f:c0:1a:8f:7e:f6:7b:6d:69:51
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=US, O=DigiCert Inc, CN=DigiCert TLS RSA SHA256 2020 CA1
        Validity
            Not Before: Mar 15 00:00:00 2024 GMT
            Not After : Mar 15 23:59:59 2025 GMT
        Subject: C=US, ST=California, L=San Francisco, O=GitHub, Inc.,
                 CN=github.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                RSA Public-Key: (2048 bit)
                Modulus:
                    00:b1:c3:b3:da:c7:3f:3e:...
        X509v3 extensions:
            X509v3 Subject Alternative Name:
                DNS:github.com, DNS:*.github.com
            X509v3 Key Usage: critical
                Digital Signature, Key Encipherment
    Signature Algorithm: sha256WithRSAEncryption
         65:f3:97:00:de:3d:8e:42:...
```

Each certificate contains:

- **Subject:** Who this certificate identifies
- **Issuer:** Who vouches for this identity
- **Public key:** The actual cryptographic key
- **Validity period:** When the certificate expires
- **Digital signature:** Issuer's cryptographic proof

Certificate Chain Validation When you connect to a website, your browser performs a complex validation:

The Astounding Scale of Success PKI operates at a scale that's difficult to comprehend:

- **4+ billion users** rely on PKI daily
- **200+ million certificates** currently active
- **100+ root CAs** in typical trust stores
- **10+ billion certificate validations** performed daily
- **99.999%+ reliability** across the entire Internet

Algorithm 2 Certificate Chain Validation

Input: Server certificate chain $[C_0, C_1, ..., C_n]$
Output: Valid or Invalid

1. Verify C_0 (server cert) matches requested domain
2. For each certificate C_i in chain:
   (a) Check current time within validity period
   (b) Verify signature: C_{i+1} signed C_i
   (c) Check C_i not revoked (OCSP/CRL check)
   (d) Verify key usage permits operation
3. Verify C_n (root) is in trust store
4. Verify chain length $\leq$ maximum (typically 5)

Major CA	Certificates Issued	Market Share
Let's Encrypt	400+ million	40%
DigiCert	50+ million	15%
Sectigo	100+ million	20%
GlobalSign	25+ million	5%
Others	100+ million	20%

Table 2: The certificate authority ecosystem (2024 data)

When PKI Fails: Catastrophic Consequences The few PKI failures demonstrate its critical importance:

Example 6 (DigiNotar Collapse (2011))**.**

- *Iranian hackers compromised Dutch CA DigiNotar*
- *Issued fraudulent certificates for *.google.com*
- *Used for man-in-the-middle attacks against Iranian citizens*
- *DigiNotar bankrupt within weeks*
- *Dutch government fell into crisis (DigiNotar secured government sites)*
- *Led to certificate pinning and Certificate Transparency*

Example 7 (Symantec CA Distrust (2017))**.**

- *Google discovered Symantec improperly issued 30,000+ certificates*
- *Chrome gradually distrusted all Symantec certificates*
- *Affected millions of websites*
- *Symantec sold CA business to DigiCert*
- *Largest CA migration in Internet history*

The Invisible Success Story PKI's success is measured by what doesn't happen:

- You don't think about verifying website identity
- You don't manually exchange keys with every service
- You don't worry about impostor websites (with valid HTTPS)
- Software updates install without malware injection
- Email servers authenticate automatically

This invisibility is PKI's greatest achievement and biggest vulnerability—users don't understand what they depend on.

Why PKI is Quantum-Vulnerable The entire PKI hierarchy depends on digital signatures:

- Root certificates: Self-signed with RSA or ECDSA
- Intermediate certificates: Signed by roots
- End-entity certificates: Signed by intermediates
- Certificate revocation lists: Signed by CAs
- OCSP responses: Signed by CA delegates

When quantum computers can forge signatures:

- Attackers can create fake certificates for any domain
- Can sign malware as Microsoft, Apple, or Google
- Can impersonate any certificate authority
- Trust hierarchy collapses completely

Remark 4 (The Migration Challenge)**.** *Replacing PKI with post-quantum signatures isn't just a technical challenge—it's a coordination problem. Every browser, operating system, server, and IoT device must update. A single widely-used system that doesn't update becomes a weakness. This is why PKI migration to PQC must start now, even though the quantum threat is years away.*

Example 8 (Certificate Size Impact)**.** *Current certificate chain (RSA-2048): 4 KB*
Post-quantum chain (Dilithium3): 15 KB
Impact: 3.75× bandwidth increase for every HTTPS connection
For Google, serving 100 billion requests/day: 1.1 PB additional traffic daily

PKI represents humanity's largest successful distributed trust system. Its invisible operation enables the entire digital economy. The quantum threat to PKI isn't just about mathematics—it's about preserving the foundation of digital trust that billions depend on without even knowing it exists.

2.1.5 Case Study: TLS Protocol

Every time you see that padlock icon in your browser, one of the most sophisticated cryptographic protocols ever designed has just executed. Transport Layer Security (TLS) seamlessly combines every concept we've discussed—digital signatures, key exchange, certificates, symmetric encryption—into a protocol that secures trillions of connections daily. Let's dissect what actually happens when you visit an HTTPS website.

The Millisecond Miracle In approximately 400 milliseconds, TLS accomplishes what once required diplomatic pouches and armed couriers:

- **Authenticates** the server (and optionally client)
- **Negotiates** cryptographic parameters
- **Establishes** shared secrets with perfect forward secrecy
- **Derives** multiple encryption keys
- **Begins** encrypted communication

All while defending against dozens of possible attacks and maintaining backward compatibility with billions of devices.

The TLS 1.3 Handshake: A Cryptographic Symphony Let's trace through an actual TLS 1.3 connection:

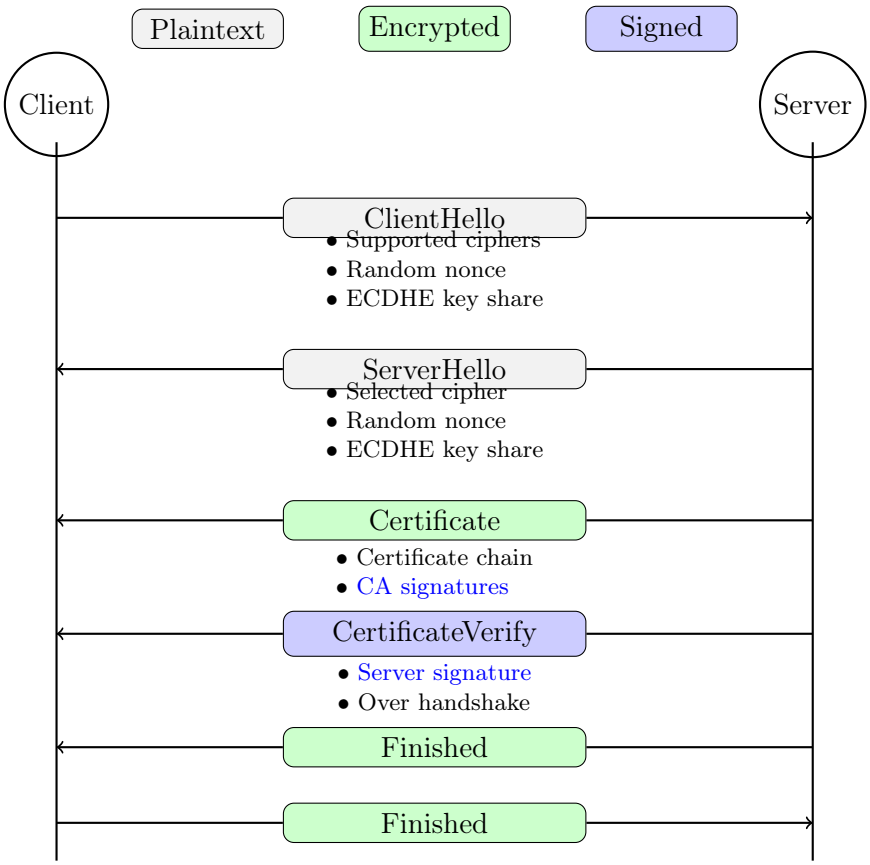


Figure 5: TLS 1.3 handshake: From first contact to encrypted channel

Step-by-Step Cryptographic Operations **Step 1: ClientHello (Client → Server)**

Listing 3: ClientHello Contents

```
ClientHello {
    legacy_version: 0x0303,          // Compatibility
    random: [32 random bytes],       // Freshness
    cipher_suites: [                 // Supported algorithms
```

```
        TLS_AES_128_GCM_SHA256,
        TLS_AES_256_GCM_SHA384,
        TLS_CHACHA20_POLY1305_SHA256
    ],
    extensions: {
        supported_groups: [x25519, secp256r1],   // ECDHE curves
        key_share: {                             // Client's public key
            group: x25519,
            key_exchange: [32-byte public key]
        },
        signature_algorithms: [                  // For certificates
            ecdsa_secp256r1_sha256,
            rsa_pss_rsae_sha256
        ]
    }
}
```

The client immediately offers its ECDHE public key, enabling the server to compute the shared secret in its first response.

Step 2: ServerHello (Server → Client)

- Server selects mutually supported cipher suite
- Provides its ECDHE public key share
- Both sides can now compute the shared secret!

Key Derivation: Both sides compute:

$$\text{ECDHE shared secret} = \text{ECDH}(\text{client\_private}, \text{server\_public}) \tag{1}$$

$$\text{Handshake Secret} = \text{HKDF}(\text{shared secret}, \text{transcript hash}) \tag{2}$$

$$\text{Multiple keys} = \text{HKDF-Expand}(\text{Handshake Secret}) \tag{3}$$

From one shared secret, TLS derives:

- client_handshake_traffic_secret
- server_handshake_traffic_secret
- client_application_traffic_secret
- server_application_traffic_secret
- resumption_master_secret

Step 3: Server Certificate (Encrypted!)
Unlike TLS 1.2, the certificate is now encrypted:

- Prevents passive observers from seeing which site you're visiting
- Certificate chain includes 2-4 certificates
- Each certificate contains an RSA or ECDSA public key
- Each signed by the next certificate in chain
- Root certificate must be in client's trust store

Step 4: CertificateVerify
The server proves it owns the private key by signing the handshake:

```
signature = Sign(server_private_key,
                 "TLS 1.3, server CertificateVerify" ||
                 0x20 * 64 ||  // Context string
                 SHA256(ClientHello ... ServerCert))
```

This signature prevents:

- Replay attacks (includes fresh randoms)
- Downgrade attacks (covers negotiated parameters)
- Certificate substitution (signs exact certificate sent)

Step 5: Finished Messages
Both sides send a MAC over the entire handshake:

$$\text{finished\_key} = \text{HKDF-Expand}(\text{traffic\_secret}, \text{"}finished\text{"}) \tag{4}$$

$$\text{verify\_data} = \text{HMAC}(\text{finished\_key}, \text{transcript\_hash}) \tag{5}$$

The Resulting Cryptographic Properties After these 400 milliseconds, TLS provides:

1. **Server Authentication:** Certificate chain + signature prove server identity
2. **Key Exchange:** ECDHE establishes shared secrets
3. **Forward Secrecy:** Ephemeral keys mean past sessions stay secure
4. **Confidentiality:** AES-GCM encrypts all application data
5. **Integrity:** AEAD prevents any tampering
6. **Replay Prevention:** Fresh randoms prevent reuse
7. **Protocol Negotiation:** Secure agreement on parameters

Performance Engineering TLS 1.3 optimizations show how critical performance is:

- **1-RTT handshake:** Full security in one round trip
- **0-RTT resumption:** Repeat connections send data immediately
- **Fewer messages:** Compared to TLS 1.2's 2-RTT handshake
- **Encrypted certificates:** Better privacy without extra cost

What Makes TLS Remarkable TLS succeeds because it:

- **Hides complexity:** Users see only a padlock
- **Adapts constantly:** New versions address emerging threats
- **Scales massively:** Same protocol for IoT devices and supercomputers
- **Maintains compatibility:** Graceful fallback for older systems
- **Prevents diverse attacks:** Defends against dozens of attack types

Operation	Time (ms)	CPU Cycles	Frequency
ECDHE key generation	0.25	500,000	Once
ECDHE shared secret	0.30	600,000	Once
Certificate verify (RSA)	0.05	100,000	Once
Certificate sign (RSA)	2.00	4,000,000	Once
AES-GCM setup	0.01	20,000	Once
AES-GCM per KB	0.002	4,000	Continuous
Total handshake	**3ms**	**6M cycles**	Per connection

Table 3: Computational cost of TLS 1.3 operations

Why Quantum Computers Devastate TLS Every single public-key operation in TLS fails against quantum computers:

- **Certificate signatures:** Shor's algorithm forges any certificate
- **ECDHE key exchange:** Quantum computers solve ECDLP
- **CertificateVerify:** Server authentication becomes forgeable
- **RSA key transport:** (TLS 1.2) Completely broken

The symmetric operations (AES-GCM, HMAC) survive with larger keys, but without functioning public-key crypto, you can't establish the symmetric keys securely!

Example 9 (Quantum Attack on TLS)**.** *Attacker with quantum computer:*

1. *Intercepts ClientHello with ECDHE public key*
2. *Uses Shor's algorithm to find private key*
3. *Computes all session keys*
4. *Decrypts entire communication*
5. *Can also forge certificates to impersonate any server*

Time required: Minutes to hours (vs. universe lifetime classically)

Remark 5 (The Migration Imperative)**.** *TLS protects approximately:*

- *5 billion web browsing users*
- *300 billion daily HTTPS connections*
- *$4 trillion in annual e-commerce*
- *Every software update and API call*

Migrating TLS to post-quantum cryptography isn't optional—it's existential for the digital economy.

This case study shows how modern cryptography weaves together multiple primitives into protocols that seem simple but embody decades of engineering. When we discuss post-quantum replacements, remember: we're not just replacing algorithms, we're preserving this entire intricate dance that happens invisibly billions of times daily.

2.2 The Computational Hardness Foundation

2.2.1 The Mathematical Bedrock

Here's a fact that should terrify you: the entire global digital security infrastructure—every secure message, every digital signature, every cryptocurrency transaction—depends on exactly three mathematical problems. Not three types of problems. Three specific problems. If any efficient algorithm is discovered for these problems, digital security collapses overnight. This mathematical monoculture is both cryptography's greatest success and its greatest vulnerability.

The Narrow Foundation Consider the asymmetry: millions of developers implement cryptography, billions of users depend on it, but only three mathematical problems protect everything:

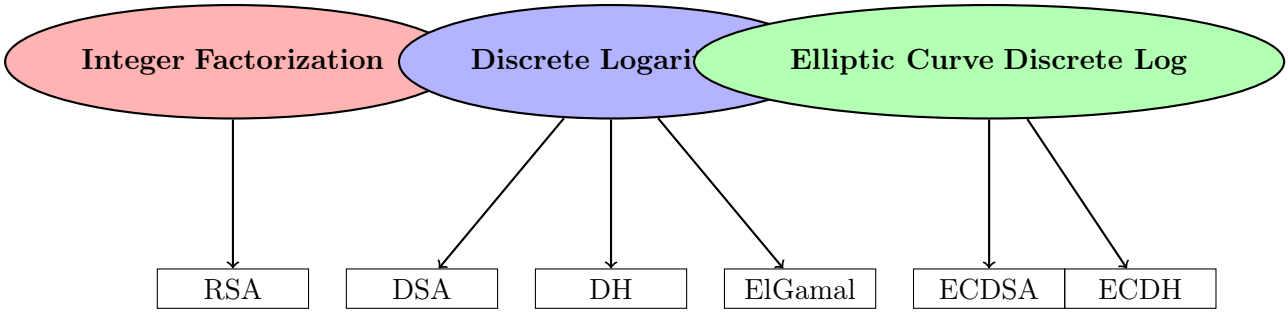


Figure 6: The three pillars of mathematical security

What Makes a Problem Cryptographically Useful? Not every hard mathematical problem can support cryptography. The requirements are stringent:

Definition 3 (Cryptographically Useful Hard Problem)**.** *A problem suitable for cryptography must have:*

1. ***Hardness gap:*** *Easy with secret, hard without*
2. ***Efficient verification:*** *Solutions checkable in polynomial time*
3. ***Algebraic structure:*** *Enables operations on encrypted data*
4. ***Scalable hardness:*** *Difficulty grows smoothly with parameter size*
5. ***Well-studied:*** *Decades without efficient algorithms*

Thousands of problems in computer science are believed hard (P vs NP contains infinitely many), but only a handful meet all these criteria. Let's examine the three that built the Internet.

Problem 1: The Integer Factorization Problem (IFP)

Definition 4 (Integer Factorization Problem)**.** *Given: A composite integer $n = pq$ where p and q are large primes*
Find: The factors p and q

The beauty lies in the asymmetry:

- **Multiplication:** Given $p = 10^{300} + 357$ and $q = 10^{300} + 577$, computing $n = pq$ takes microseconds
- **Factorization:** Given only n, finding p and q takes longer than the universe has existed

Example 10 (Concrete Difficulty Scaling)**.**

Bits	***Example*** n	***Factoring Time (Classical)***
20	*1,048,309*	*Microseconds*
64	*18,446,744,073,709,551,437*	*Seconds*
128	*340,282,366,920,938,463,374,875,442,601*	*Hours*
512	*RSA-155 (broken 1999)*	*8,000 MIPS-years*
1024	*RSA-309*	$\sim 10^{12}$ *years*
2048	*Current standard*	$\sim 10^{20}$ *years*

Best classical algorithm: General Number Field Sieve

$$\text{Time} = \exp\left(c \cdot (n)^{1/3} \cdot (\log\log n)^{2/3}\right) \tag{6}$$

where $c \approx 1.923$. This sub-exponential but super-polynomial growth creates the cryptographic gap.

Problem 2: The Discrete Logarithm Problem (DLP)

Definition 5 (Discrete Logarithm Problem)**.** *Given: A group G, generator $g \in G$, and element $h \in G$*
Find: Integer x such that $g^x = h$ in G

In the multiplicative group $\mathbb{Z}_p^*$ (integers modulo prime p):

- **Exponentiation:** Given $g = 5$, $x = 76543$, $p = 10^{308} + 129$, computing $h = g^x \bmod p$ is fast
- **Logarithm:** Given g, h, and p, finding x is infeasible

The discrete logarithm is the modular arithmetic analog of ordinary logarithms, but dramatically harder:

$$\text{Real numbers:} \quad 2^x = 1024 \implies x = \log_2(1024) = 10 \quad \text{(Easy)} \tag{7}$$

$$\text{Modular arithmetic:} \quad 2^x \equiv 1024 \pmod{p} \implies x = ? \quad \text{(Hard)} \tag{8}$$

Problem 3: The Elliptic Curve Discrete Logarithm Problem (ECDLP)

Definition 6 (Elliptic Curve Discrete Logarithm Problem)**.** *Given: Elliptic curve E over finite field, base point $G \in E$, point $P \in E$*
Find: Integer k such that $kG = P$ (using elliptic curve addition)

Elliptic curves provide the same discrete logarithm hardness with much smaller numbers:

Example 11 (Elliptic Curve Efficiency)**.**

Security Level	***RSA/DLP***	***ECC***	***Ratio***
80 bits	*1,024 bits*	*160 bits*	*6.4:1*
128 bits	*3,072 bits*	*256 bits*	*12:1*
256 bits	*15,360 bits*	*512 bits*	*30:1*

The curve equation (simplified Weierstrass form):

$$y^2 = x^3 + ax + b \pmod{p} \tag{9}$$

Point addition uses geometric intuition translated to algebra:

$$P + Q = R \text{ where line through } P, Q \text{ intersects curve at } -R \tag{10}$$

$$P + P = 2P \text{ using tangent line at } P \tag{11}$$

The Deep Connection These problems share a hidden connection that explains their cryptographic utility:

Theorem 1 (Hidden Subgroup Problem)**.** *All three problems reduce to finding hidden subgroups:*

- ***Factoring:*** *Find subgroup of* $\mathbb{Z}_n^*$ *generated by* $x \mapsto x^2$
- ***DLP:*** *Find period of* $f(a, b) = g^a h^b$
- ***ECDLP:*** *Find period in elliptic curve group*

This connection explains why: 1. Similar classical algorithms work (Pollard ρ, index calculus variants) 2. The same quantum algorithm (Shor) breaks all three 3. They provide similar cryptographic functionality

Why These Three Dominated Historical and mathematical reasons created this monoculture:

1. **Early discovery:** RSA (1977), DH (1976) came first
2. **Patent landscape:** RSA patent expired 2000, allowing widespread use
3. **Implementation maturity:** 40+ years of optimization
4. **Mathematical elegance:** Clean algebraic structure
5. **Efficiency tradeoffs:** Good balance of key size vs. computation
6. **Resistance to classical attacks:** No breakthrough despite huge efforts

The Monoculture Risk This narrow foundation creates systemic risk:

- **Single point of failure:** One algorithmic breakthrough breaks everything
- **Quantum vulnerability:** All three fall to the same quantum algorithm
- **Implementation similarity:** Side-channel attacks often transfer
- **Limited diversity:** Few backup options if problems fall

Remark 6 (The Search for Problem #4)**.** *Cryptographers have desperately sought a fourth problem for decades:*

- *Lattice problems (finally successful!)*
- *Knapsack problems (broken in 1980s)*
- *Code-based problems (large keys limited adoption)*
- *Multivariate polynomials (repeated breaks)*
- *Isogenies (SIKE broken in 2022)*

The difficulty of finding new problems underscores how special these three are.

Example 12 (Near Miss: The Knapsack Cryptosystem)**.** *In 1978, Merkle-Hellman proposed using the knapsack problem:*

- *Given: Set of weights* $\{a_1, ..., a_n\}$ *and sum* S

- *Find: Subset summing to exactly S*
- *Problem: NP-complete, should be hard!*
- *Reality: Broken by Shamir in 1982 using lattice reduction*
- *Lesson: Not all hard problems make good cryptography*

This mathematical bedrock supported the entire digital revolution. That it consists of only three problems is both a testament to their power and a warning about our vulnerability. When quantum computers arrive, this narrow foundation becomes a single point of catastrophic failure—motivating the urgent search for quantum-resistant alternatives we'll explore in the remainder of this lecture.

2.2.2 From Hard Problems to Cryptosystems

Having a hard mathematical problem is necessary but not sufficient for cryptography. The journey from "factoring is hard" to "RSA is secure" involves subtle constructions, security proofs, and countless implementation details. Many cryptosystems have failed not because their underlying problem was easy, but because the construction was flawed. Understanding this gap is crucial for evaluating post-quantum proposals.

The Reduction Concept: Proving Security Cryptographers cannot prove systems are secure in an absolute sense. Instead, they prove security through reductions:

Definition 7 (Security Reduction)**.** *A security reduction shows: If you can break the cryptosystem with resources (t, ϵ), then you can solve the underlying hard problem with resources (t', ϵ') where $t' \approx t$ and $\epsilon' \approx \epsilon$.*

Contrapositive: If the problem is hard, the cryptosystem is secure.

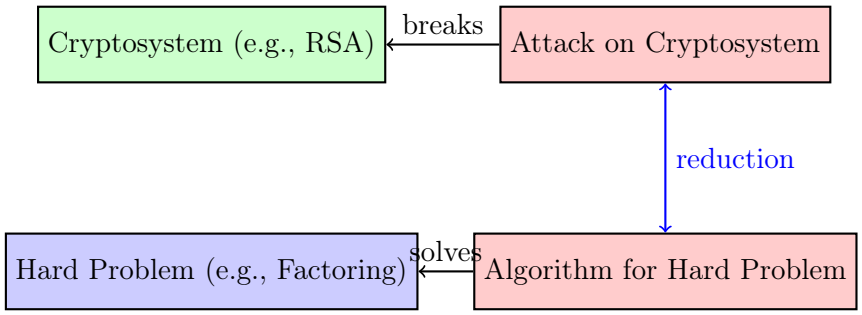


Figure 7: Security reduction: Cryptosystem security reduces to problem hardness

Case Study: From Factoring to RSA Let's trace the complete construction of RSA from the factoring assumption:

Step 1: The Mathematical Foundation

- Hard problem: Given $n = pq$, find p and q
- Euler's theorem: For $\gcd(m, n) = 1$: $m^{\phi(n)} \equiv 1 \pmod{n}$
- Where $\phi(n) = (p-1)(q-1)$ (Euler's totient function)

Algorithm 3 Textbook RSA Construction

KeyGen:
Choose large primes p, q (e.g., 1024 bits each)
Compute $n = pq$ and $\phi(n) = (p-1)(q-1)$
Choose e with $\gcd(e, \phi(n)) = 1$ (often $e = 65537$)
Compute d where $ed \equiv 1 \pmod{\phi(n)}$
Public key: (n, e); Private key: d

Encrypt$(m, (n, e))$:
Return $c = m^e \bmod n$

Decrypt(c, d):
Return $m = c^d \bmod n$

Step 2: The Basic Construction
Why it works:

$$c^d = (m^e)^d = m^{ed} \pmod{n} \tag{12}$$
$$= m^{1+k\phi(n)} \text{ for some integer } k \tag{13}$$
$$= m \cdot (m^{\phi(n)})^k \tag{14}$$
$$= m \cdot 1^k = m \pmod{n} \tag{15}$$

The Dangerous Gap: Textbook RSA is Insecure! The basic construction above is completely insecure in practice:

Example 13 (Malleability Attack on Textbook RSA)**.** *Given encryption of m: $c = m^e \bmod n$*

- *Attacker computes: $c' = c \cdot 2^e = (m \cdot 2)^e \bmod n$*
- *Gets decryption: $m' = 2m$*
- *Recovers: $m = m'/2$*

Attacker decrypts without knowing d!

Example 14 (Small Message Attack)**.** *If $m < n^{1/e}$, then $m^e < n$, so:*

- *$c = m^e \bmod n = m^e$ (no modular reduction!)*
- *Attacker computes: $m = c^{1/e}$ (ordinary e-th root)*

The Secure Construction: RSA with OAEP Real RSA uses Optimal Asymmetric Encryption Padding (OAEP):

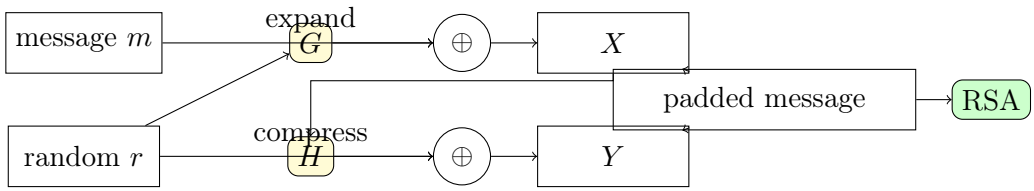


Figure 8: RSA-OAEP: Secure padding before encryption

OAEP provides:

- **Randomization:** Same message encrypts differently each time
- **All-or-nothing:** Cannot decrypt partially
- **Non-malleability:** Cannot create related ciphertexts
- **Chosen-ciphertext security:** Provably secure in random oracle model

Other Constructions from Our Three Problems **Diffie-Hellman Key Exchange:**

- Problem: Discrete logarithm in $\mathbb{Z}_p^*$
- Construction: Exchange g^a, g^b; compute g^{ab}
- Security: Computing g^{ab} from g^a, g^b is hard (CDH assumption)
- Subtlety: Needs authentication to prevent man-in-the-middle

DSA (Digital Signature Algorithm):

- Problem: Discrete logarithm in subgroup of $\mathbb{Z}_p^*$
- Construction: Complex - uses ephemeral keys, modular arithmetic
- Security: Forging requires solving discrete log
- Subtlety: Reusing ephemeral key reveals private key instantly!

ECDSA (Elliptic Curve DSA):

- Problem: Discrete logarithm in elliptic curve group
- Construction: Similar to DSA but over elliptic curves
- Security: Based on ECDLP hardness
- Subtlety: Biased nonces leak private key (PlayStation 3 hack)

When Constructions Fail History is littered with broken constructions despite hard problems:

Example 15 (Dual EC DRBG Backdoor)**.**
- *Based on: Elliptic curve discrete logarithm (hard!)*
- *Construction: Generate random numbers using EC points*
- *Failure: NSA chose parameters with hidden backdoor*
- *Lesson: Construction details matter as much as hardness*

Example 16 (Knapsack Cryptosystems)**.**
- *Based on: Subset sum problem (NP-complete!)*
- *Construction: Used special "easy" knapsacks*
- *Failure: Structure needed for decryption enabled attacks*
- *Lesson: Worst-case hardness $\neq$ average-case security*

Requirements for Secure Constructions A secure cryptosystem needs:

1. **Hard problem:** Computational assumption
2. **Tight reduction:** Security loss should be minimal
3. **Proper padding:** Prevent structural attacks
4. **Parameter selection:** Avoid weak instances
5. **Implementation care:** Prevent side channels
6. **Cryptanalysis:** Survive years of expert scrutiny

System	Problem	Construction Issue	Status
Textbook RSA	Factoring	No padding	Insecure
RSA-OAEP	Factoring	OAEP padding	Secure
Merkle-Hellman	Knapsack	Special structure	Broken
McEliece	Decoding	Random codes	Secure (PQC!)
Dual EC	ECDLP	Backdoored params	Broken

Table 4: Hard problems don't guarantee secure constructions

Implications for Post-Quantum Cryptography This construction gap explains why:

- Having lattice problems isn't enough—need LWE construction
- Multiple PQC candidates failed despite hard problems
- Parameter selection is crucial (SIKE fell to weak parameters)
- Implementation challenges persist (Gaussian sampling in lattices)
- Standardization takes years of cryptanalysis

Remark 7 (The Art of Cryptographic Design)**.** *Cryptography is equal parts mathematics and engineering. The most elegant mathematical problem is useless without a secure, efficient, implementable construction. This is why cryptographers say "Don't roll your own crypto"—the gap between theory and practice is filled with subtle traps that have caught even experts.*

Understanding this gap prepares us for the post-quantum transition. When we examine lattice-based cryptography, we'll see how Learning With Errors provides not just a hard problem, but a construction framework that enables encryption, signatures, and advanced protocols—making it the successor to RSA's throne.

2.2.3 The Classical-Quantum Divide

The quantum threat to cryptography isn't about raw computational power—it's about a fundamental difference in how quantum computers solve certain problems. Understanding this divide requires seeing beyond "quantum = faster" to grasp why specific mathematical structures become vulnerable while others remain hard.

The Complexity Chasm Classical and quantum computers differ not in speed but in computational paradigm:

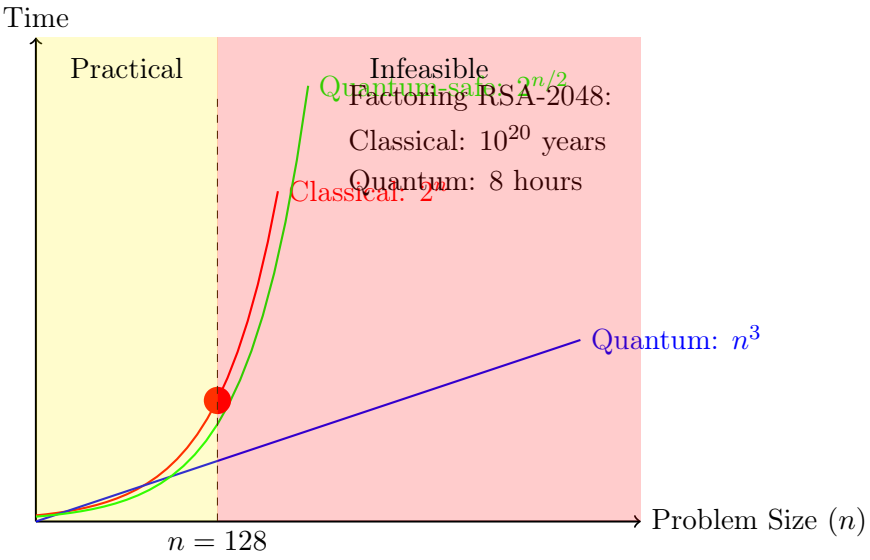


Figure 9: The exponential-polynomial divide: Where quantum changes everything

The Physics Analogy: Computational Energy Barriers For physicists, the best analogy is quantum tunneling through energy barriers:

Definition 8 (Computational Landscape Analogy)**.**

- ***Problem space:*** *Configuration space of possible solutions*
- ***Computational cost:*** *Energy barrier height*
- ***Classical algorithm:*** *Thermal activation over barrier*
- ***Quantum algorithm:*** *Tunneling through barrier*

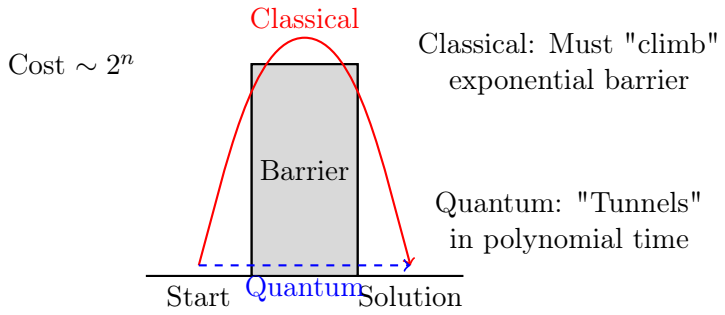


Figure 10: Quantum tunneling analogy for algorithmic speedup

Just as quantum particles tunnel through classically forbidden regions, quantum algorithms find solutions through classically inaccessible computational paths.

The Hidden Structure: Periodicity The quantum advantage isn't universal—it exploits specific mathematical structure:

Theorem 2 (The Period-Finding Connection)**.** *All three cryptographic problems hide periodic structures:*

1. ***Factoring:*** *Find period of* $f(x) = a^x \bmod n$

2. ***Discrete Log:*** *Find period of* $f(j,k) = g^j h^{-k} \bmod p$

3. ***ECDLP:*** *Find period in elliptic curve group operations*

The quantum Fourier transform excels at finding hidden periods:

Algorithm 4 Shor's Algorithm Structure (Simplified)

Goal: Factor $n = pq$

1. Choose random $a < n$ with $\gcd(a, n) = 1$
2. Define $f(x) = a^x \bmod n$ (periodic function)
3. **Quantum step:** Use QFT to find period r
4. If r is even and $a^{r/2} \not\equiv -1$:
   $p = \gcd(a^{r/2} - 1, n)$
   $q = \gcd(a^{r/2} + 1, n)$

Why Period Finding is Quantum-Easy The quantum advantage comes from superposition and interference:

1. **Superposition:** Evaluate $f(x)$ on all x simultaneously
2. **Entanglement:** Create $\sum_x |x\rangle|f(x)\rangle$
3. **Measurement:** Collapse to states with same $f(x)$ value
4. **Interference:** QFT extracts period from amplitude pattern

Mathematically:

$$|\psi\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle|f(x)\rangle \tag{16}$$

$$\text{After measuring } f(x) = y: \quad |\psi'\rangle = \frac{1}{\sqrt{|S_y|}} \sum_{x \in S_y} |x\rangle \tag{17}$$

$$\text{Where } S_y = \{x : f(x) = y\} = \{x_0, x_0 + r, x_0 + 2r, ...\} \tag{18}$$

$$\text{QFT extracts: } r \text{ (the period)} \tag{19}$$

Not All Hard Problems Are Quantum-Vulnerable Crucially, quantum computers don't make all hard problems easy:

Problem	**Structure**	**Classical**	**Quantum**
Integer Factorization	Hidden period	Exponential	Polynomial
Discrete Logarithm	Hidden period	Exponential	Polynomial
Shortest Vector (Lattice)	Geometric	Exponential	Still exponential*
Subset Sum (General)	Combinatorial	Exponential	Square root speedup
SAT (Boolean)	Constraint	Exponential	Square root speedup

Table 5: Not all hard problems have exploitable quantum structure. *Best known

The Quantum Algorithm Paradigm Quantum advantage requires problems with specific properties:

Definition 9 (Quantum-Exploitable Structure)**.** *A problem admits substantial quantum speedup if it has:*

1. ***Hidden periodicity:*** *Shor's algorithm applies*
2. ***Hidden subgroup:*** *Generalization of periodicity*
3. ***Unstructured search:*** *Grover gives square root speedup*
4. ***Quantum simulation:*** *Natural quantum problems*

Problems without these structures remain hard:

- **Random-looking problems:** No pattern to exploit
- **High-dimensional geometry:** Lattice problems
- **Non-linear structures:** Multivariate polynomials
- **Hash functions:** Only Grover speedup (manageable)

What Makes a Problem Quantum-Safe? Post-quantum cryptography seeks problems that resist quantum attack:

1. **No hidden period:** Defeats Shor-like algorithms
2. **No efficient quantum representation:** States grow exponentially
3. **Classical-quantum gap minimal:** At most polynomial speedup
4. **Grover-resistant parameters:** Account for square root speedup

Example 17 (Why Lattices Resist Quantum Attack)**.** *Shortest Vector Problem in lattices:*

- *No known periodic structure*
- *Geometric problem in high dimensions*
- *Best quantum algorithm: Minor speedup over classical*
- *Believed quantum-hard for decades*

The Fundamental Divide The classical-quantum divide for our three problems is absolute:

$$\text{RSA-2048 factoring:} \quad \text{Classical: } 10^{20} \text{ years} \tag{20}$$

$$\text{Quantum: 8 hours} \tag{21}$$

$$\text{Speedup factor:} \quad 10^{24}\times \tag{22}$$

This isn't an engineering improvement—it's a paradigm shift. No amount of classical optimization can bridge this gap.

Remark 8 (A Physicist's Perspective)**.** *Just as quantum mechanics revealed that classical physics fundamentally cannot explain certain phenomena (blackbody radiation, photoelectric effect), quantum computing reveals that classical computation fundamentally cannot efficiently solve certain problems. The periodicity in factoring/discrete log is like a "computational quantum phenomenon"—invisible to classical algorithms but obvious to quantum ones.*

This divide explains why post-quantum cryptography cannot simply use "bigger keys" for RSA or elliptic curves. The exponential-to-polynomial collapse is too severe. We need fundamentally different mathematical foundations—problems that remain exponentially hard even for quantum computers. Next, we'll see why lattice problems provide exactly this quantum resistance.

3 The Quantum Threat and PQC Landscape

3.1 Quantum Algorithms Breaking Classical Crypto

3.1.1 Shor's Algorithm: The Cryptographic Killer

Let me dispel a dangerous misconception immediately: Shor's algorithm does not try all possible factors in parallel. This naive view—that quantum computers are just massively parallel classical computers—fundamentally misunderstands both the algorithm and the quantum threat. Shor's algorithm does something far more subtle and devastating: it transforms the factoring problem into finding hidden periods, then exploits quantum interference to extract these periods efficiently.

The Period-Finding Paradigm The genius of Shor's algorithm lies in recognizing that factoring hides a periodic structure that quantum computers can detect through interference—much like how a diffraction grating reveals the wavelength of light.

Definition 10 (The Period-Finding Problem)**.** *Given a function $f : \mathbb{Z} \to \mathbb{Z}$ with hidden period r (i.e., $f(x) = f(x + r)$ for all x), find the smallest positive r.*

Classically, finding periods requires evaluating f at many points—essentially $O(\sqrt{r})$ evaluations for a period of size r. But quantum mechanically, we can create a superposition over all function values and use interference to extract the period in polynomial time.

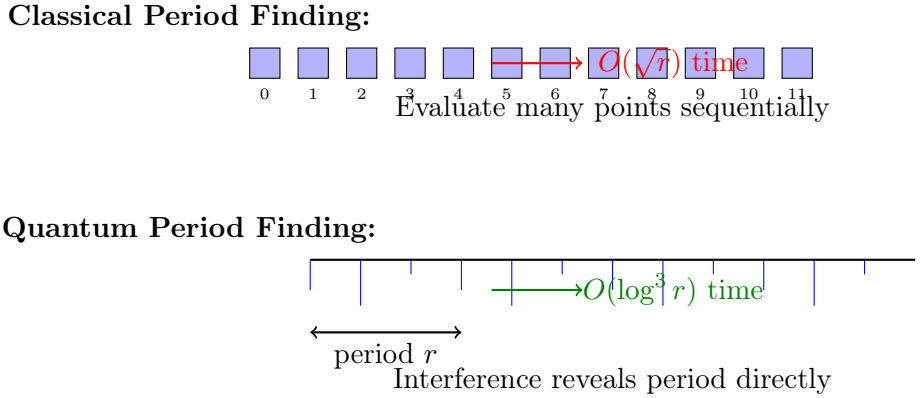


Figure 11: Classical sampling vs. quantum interference for period finding

The physics analogy is precise: just as optical interference patterns reveal spatial periodicity, quantum interference reveals computational periodicity. The Quantum Fourier Transform (QFT) acts like a computational diffraction grating.

From Factoring to Period Finding Here's the critical insight that makes factoring quantum-easy:

Theorem 3 (Factoring Reduces to Period Finding)**.** *To factor $n = pq$:*

1. *Choose random a with $\gcd(a, n) = 1$*

2. *Find the period* r *of* $f(x) = a^x \bmod n$

3. *If* r *is even and* $a^{r/2} \not\equiv -1 \pmod{n}$*:*

   - $\gcd(a^{r/2} - 1, n)$ *gives a nontrivial factor*
   - *Success probability* $\geq 1/2$ *for random* a

Proof Sketch. Since $a^r \equiv 1 \pmod{n}$, we have $(a^{r/2})^2 \equiv 1 \pmod{n}$, so:

$$(a^{r/2} - 1)(a^{r/2} + 1) \equiv 0 \pmod{n} \tag{23}$$

If $a^{r/2} \not\equiv \pm 1 \pmod{n}$, then $a^{r/2} - 1$ shares a nontrivial factor with n. □

Algorithm 5 Shor's Algorithm for Integer Factorization

Require: $n = pq$ (semiprime to factor)
Ensure: Factors p, q
Choose random $a \in \{2, ..., n-1\}$
Compute $g = \gcd(a, n)$ classically
if $g > 1$ **then**
    return g and n/g ▷ Lucky! Found factor classically
end if
Quantum Period Finding:
    Initialize quantum registers: $|0\rangle^{\otimes m}|0\rangle^{\otimes n}$
    Create superposition: $\frac{1}{\sqrt{2^m}} \sum_{x=0}^{2^m-1} |x\rangle|0\rangle$
    Compute $f(x) = a^x \bmod n$ in second register
    Measure second register (collapses to periodic subset)
    Apply QFT to first register
    Measure first register → get value y
    Use continued fractions on $y/2^m$ to find period r
if r is even and $a^{r/2} \not\equiv -1 \pmod{n}$ **then**
    return $\gcd(a^{r/2} - 1, n)$ and $\gcd(a^{r/2} + 1, n)$
else
    Retry with different a
end if

Concrete Example: Factoring 15 Let's factor $n = 15$ step by step to see the algorithm in action:

Example 18 (Factoring 15 with Shor's Algorithm)**.** ***Step 1:*** *Choose* $a = 7$ *(random choice with* $\gcd(7, 15) = 1$*)*

Step 2: *We need the period of* $f(x) = 7^x \bmod 15$*:*

$$7^0 \equiv 1 \pmod{15} \tag{24}$$
$$7^1 \equiv 7 \pmod{15} \tag{25}$$
$$7^2 \equiv 49 \equiv 4 \pmod{15} \tag{26}$$
$$7^3 \equiv 28 \equiv 13 \pmod{15} \tag{27}$$
$$7^4 \equiv 91 \equiv 1 \pmod{15} \tag{28}$$

Period $r = 4$ *(since* $7^4 \equiv 7^0 \pmod{15}$*)*

Step 3: *Quantum circuit finds* $r = 4$ *through interference:*

$$|0\rangle \xrightarrow{} \frac{1}{\sqrt{8}}\sum_{x=0}^{7}|x\rangle \xrightarrow{} \frac{1}{\sqrt{8}}\sum_{x}|x\rangle|7^x \bmod 15\rangle$$

Measure $f(x)$: *Get* $|1\rangle \rightarrow$ $\frac{1}{2}(|0\rangle + |4\rangle)$

After QFT: *Measure* $\rightarrow$ *Get* 0 *or* 2 *Period* $= \frac{8}{2} = 4$

Step 4: *Classical post-processing:*

- $r = 4$ *is even* ✓
- $a^{r/2} = 7^2 = 49 \equiv 4 \pmod{15}$
- $4 \not\equiv -1 \pmod{15}$ ✓
- $\gcd(7^2 - 1, 15) = \gcd(48, 15) = 3$
- $\gcd(7^2 + 1, 15) = \gcd(50, 15) = 5$

Result: $15 = 3 \times 5$ ✓

The Quantum Circuit Architecture The actual quantum circuit for Shor's algorithm requires precise resource accounting:

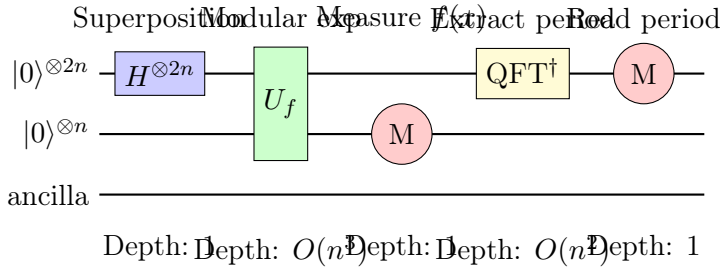


Figure 12: High-level quantum circuit for Shor's algorithm with depth annotations

Remark 9 (Logical vs Physical Qubits)**.** *Critical distinction for realistic assessment:*

- ***Logical qubits:*** *Error-free qubits in the algorithm*
- ***Physical qubits:*** *Noisy qubits in actual hardware*
- ***Error correction overhead:*** *∼1,000-10,000 physical per logical*
- ***Current state (2025):*** *∼1,000 physical qubits, 0 logical qubits*

For factoring an n-bit number:

- **Logical qubits:** $3n + O(\log n)$
- **Quantum gates:** $O(n^3)$ (dominated by modular exponentiation)
- **Circuit depth:** $O(n^3)$
- **Classical runtime:** $O(n^3)$ for polynomial-time quantum circuits
- **Physical qubits (with error correction):**

- Conservative: ∼10,000 physical per logical
- Optimistic: ∼1,000 physical per logical
- For RSA-2048: 20-200 million physical qubits

Remark 10 (Breaking RSA-2048: Reality Check)**.** *The "8 hours" claim assumes:*

- *Logical qubits with perfect gates (not physical qubits)*
- *Clock speed of ∼1 MHz for quantum operations*
- *No decoherence or error correction overhead*
- *Efficient classical control systems*

With realistic error correction and current parameters, breaking RSA-2048 would require:

- *∼4,000 logical qubits*
- *∼20 million physical qubits (conservative)*
- *∼8 hours of coherent quantum computation*
- *Technology gap: 4-5 orders of magnitude from current systems*

Impact Analysis: The Dual Catastrophe Here's what many miss: Shor's algorithm doesn't just break encryption—it equally devastates digital signatures:

Algorithm	**Purpose**	**Broken?**	**Impact**
RSA-2048	Encryption	✗	No confidentiality
RSA-2048	Signatures	✗	No authentication
ECDH-P256	Key exchange	✗	No secure channels
ECDSA-P256	Signatures	✗	No identity verification
DSA	Signatures	✗	No integrity protection
AES-128	Encryption	↓	Reduced to 64-bit security
SHA-256	Hashing	↓	Reduced to 128-bit security

Table 6: Shor's algorithm breaks all asymmetric cryptography, not just encryption

Example 19 (Breaking Digital Signatures with Shor)**.** *Given ECDSA public key $Q = dG$ on curve with order n:*

1. *Use Shor to solve discrete log: find d from $Q = dG$*
2. *Now you have the private key d*
3. *Forge any signature: $s = k^{-1}(h + rd) \bmod n$*
4. *Result: Perfect impersonation of key owner*

Time required: Same as breaking encryption (polynomial in key size)

Timeline and the "Harvest Now, Decrypt Later" Threat The timeline creates an unusual situation in cryptography:

- **Today (2025):** No quantum computer can run Shor's algorithm at scale
- **2033:** IBM roadmap targets 100,000 physical qubits
- **2035-2040:** Possible threshold for logical qubits via error correction
- **Critical insight:** Data encrypted today can be stored and decrypted later

Remark 11 (The Retrospective Threat)**.** *If your encrypted data has value for 10+ years, it's already at risk:*

- *Government secrets (25-50 year classification)*
- *Medical records (lifetime + 10 years)*
- *Financial records (7-10 years regulatory)*
- *Industrial IP (20+ year patents)*
- *Personal communications (lifetime privacy)*

Adversaries are likely harvesting encrypted data now for future decryption. This isn't paranoia—it's documented in national security assessments.

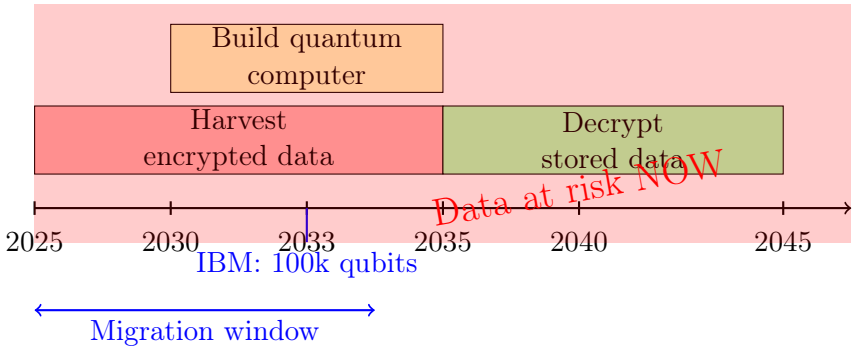


Figure 13: The harvest now, decrypt later timeline aligned with quantum roadmaps

Why No Classical Algorithm Can Compete The quantum advantage for factoring is absolute, not incremental:

$$\text{Best classical (GNFS):} \quad L_n[1/3, (64/9)^{1/3}] \approx \exp(1.9 \cdot n^{1/3} \cdot (\log n)^{2/3}) \tag{29}$$

$$\text{Shor's algorithm (quantum):} \quad O(n^3) \text{ quantum operations} \tag{30}$$

$$\text{For RSA-2048:} \quad \text{Classical: } 10^{20} \text{ years vs. Quantum: 8 hours}^* \tag{31}$$

*With perfect logical qubits; see reality check above for physical requirements.

This is not a speedup that Moore's Law can address. If classical computers became 10^{10} times faster, factoring RSA-2048 would still take a billion years. The quantum computer doesn't work harder—it works differently.

Remark 12 (A Physicist's Perspective)**.** *Think of it this way: classical algorithms must explore an exponentially large solution space, like a particle diffusing through a vast energy landscape. Quantum algorithms use interference to "tunnel" directly to the solution, bypassing the exponential search entirely. It's not about speed—it's about taking a fundamentally different path through the computation.*

The implications are stark: every RSA key, every ECDSA signature, every Diffie-Hellman exchange will become breakable. Not weakened—completely broken. This isn't a distant theoretical threat; it's an engineering challenge that billions of dollars are actively solving. The only question is when, not if.

With this existential threat to asymmetric cryptography established, we now turn to the more subtle challenge that quantum computers pose to symmetric systems.

3.1.2 Grover's Algorithm: The Symmetric Threat

While Shor's algorithm gets the headlines—"Quantum computers will break all encryption!"—Grover's algorithm poses a more subtle threat. It doesn't obliterate symmetric cryptography the way Shor destroys RSA, but it does weaken it significantly. The good news: this weakening is predictable, bounded, and manageable. The bad news: many people either ignore it entirely or wildly overestimate what it can do.

The Unstructured Search Problem Grover's algorithm solves a deceptively simple problem:

Definition 11 (Unstructured Search)**.** *Given a function $f : \{0,1\}^n \to \{0,1\}$ (a "black box" oracle) where exactly one input x^* satisfies $f(x^*) = 1$, find x^*.*

Classically, this is the ultimate needle-in-a-haystack problem. With no structure to exploit, you must check inputs one by one:

- **Classical:** Expected 2^{n-1} evaluations (half the search space)
- **Quantum:** Only $\frac{\pi}{4}\sqrt{2^n} \approx 0.785 \cdot 2^{n/2}$ evaluations
- **Speedup:** Quadratic (square root)

Example 20 (Concrete Impact on Search)**.** *Breaking a 128-bit key:*

- *Classical:* $2^{127} \approx 10^{38}$ *operations (impossible)*
- *Quantum:* $2^{64} \approx 10^{19}$ *operations (difficult but feasible)*
- *Mitigation: Use 256-bit key* $\to 2^{128}$ *quantum operations (safe again)*

Remark 13 (Why Grover Doesn't Threaten Asymmetric Crypto)**.** *Students often ask: "Can't Grover search for RSA private keys?" Let's address this immediately:*

- *RSA-2048 private key space:* $\sim 2^{2048}$
- *Grover speedup:* 2^{1024} *quantum operations*
- *Universe lifetime:* $\sim 2^{60}$ *nanoseconds*
- *Still need* 2^{964} *universe lifetimes!*

Grover provides only quadratic speedup—insufficient to overcome exponential key spaces in asymmetric cryptography. This is why we focus on Shor for asymmetric and Grover for symmetric systems.

How Grover's Algorithm Works Here's the critical insight: Grover's algorithm does NOT check all possibilities in parallel. This misconception leads to confusion about its capabilities and limitations.

Amplitude Evolution:

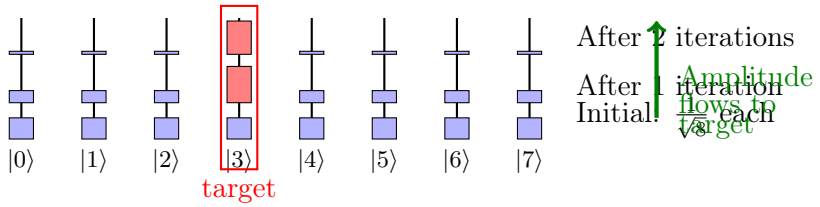


Figure 14: Grover's algorithm amplifies the target state's amplitude through interference

Instead of parallel search, Grover uses **amplitude amplification**:

1. Start with equal superposition: $|\psi\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle$
2. Mark the target state by flipping its phase: $|x^*\rangle \to -|x^*\rangle$
3. Perform "inversion about average" operation
4. Repeat $\approx \frac{\pi}{4}\sqrt{N}$ times
5. Measure to get x^* with high probability

The physics analogy: think of a resonant cavity where constructive interference gradually builds up amplitude at the target frequency while destructive interference suppresses others.

Algorithm 6 Grover's Algorithm

Require: Oracle O_f where $O_f|x\rangle = (-1)^{f(x)}|x\rangle$
Require: Number of items $N = 2^n$
Ensure: Solution x^* where $f(x^*) = 1$
Initialize: $|\psi\rangle = H^{\otimes n}|0\rangle^{\otimes n} = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle$
Set iterations: $k = \lfloor \frac{\pi}{4}\sqrt{N} \rfloor$
for $i = 1$ to k **do**
    Apply oracle: $|\psi\rangle \leftarrow O_f|\psi\rangle$ ▷ Flip phase of target
    Apply diffusion: $|\psi\rangle \leftarrow D|\psi\rangle$ ▷ Inversion about average
        where $D = 2|s\rangle\langle s| - I$ and $|s\rangle = \frac{1}{\sqrt{N}} \sum_x |x\rangle$
end for
Measure $|\psi\rangle$ to get x
Verify: if $f(x) = 1$ return x, else repeat

The Algorithm in Detail The Grover operator $G = D \cdot O_f$ has elegant geometric interpretation:

- O_f: Reflection through hyperplane orthogonal to target
- D: Reflection through average state

- G: Rotation in 2D subspace spanned by target and non-target states
- Angle of rotation: $\theta = 2\arcsin(1/\sqrt{N})$

After k iterations, the amplitude of the target state is:

$$\alpha_k = \sin\left((2k+1)\arcsin(1/\sqrt{N})\right) \tag{32}$$

Optimal number of iterations: $k_{\text{opt}} = \lfloor \frac{\pi}{4}\sqrt{N} - \frac{1}{2} \rfloor$

Impact on Symmetric Cryptography Let's quantify the concrete impact on real cryptographic primitives:

Algorithm	**Classical Security**	**Quantum Attack**	**Effective Security**	**Safe?**
AES-128	2^{128}	2^{64}	64 bits	No
AES-192	2^{192}	2^{96}	96 bits	Marginal
AES-256	2^{256}	2^{128}	128 bits	Yes
SHA-256 (collision)	2^{128}	$2^{85.3}$	85 bits	No
SHA-384 (collision)	2^{192}	2^{128}	128 bits	Yes
SHA-512 (collision)	2^{256}	$2^{170.7}$	171 bits	Yes
SHA-256 (preimage)	2^{256}	2^{128}	128 bits	Yes
SHA-512 (preimage)	2^{512}	2^{256}	256 bits	Yes

Table 7: Grover's impact on symmetric primitives (Note: collision finding gets $2^{n/3}$ due to BHT algorithm)

Example 21 (Breaking AES-128 with Grover)**.** *Given ciphertext $C = AES_k(P)$ where P is known:*

- *Define oracle:* $f(k') = \begin{cases} 1 & \text{if } AES_{k'}(P) = C \\ 0 & \text{otherwise} \end{cases}$
- *Search space:* 2^{128} *possible keys*
- *Grover iterations needed:* $\frac{\pi}{4} \cdot 2^{64} \approx 1.25 \times 10^{19}$
- *Each iteration: 1 AES encryption + quantum gates*
- *Total time:* $\sim 2^{64}$ *AES operations*

This is feasible with a large quantum computer, unlike the classical 2^{128}*.*

Why This is Manageable Unlike Shor's algorithm, Grover's threat has a simple solution:

Theorem 4 (Grover's Bounded Impact)**.** *For any function with n-bit security against classical attack:*

- *Quantum security: $n/2$ bits*
- *Mitigation: Use $2n$-bit keys*
- *Cost: Linear increase in key size*

Compare the mitigation strategies:

- **Shor's algorithm on RSA:**
  - Going from 2048 to 4096 bits: Still broken instantly
  - Going to 1 million bits: Still polynomial time for quantum
  - Only solution: Abandon RSA entirely
- **Grover's algorithm on AES:**
  - Going from 128 to 256 bits: Full security restored
  - Cost: 2× key storage, identical performance
  - Implementation: Change one parameter

Practical Implications and Recommendations Current guidance for quantum-resistant symmetric cryptography:

1. **Use AES-256** for long-term security
   - Already standard in many applications
   - No performance penalty on modern hardware (AES-NI)
   - Provides 128-bit post-quantum security
2. **Use SHA-384 or SHA-512** for hashing
   - SHA-256 remains safe for preimage resistance
   - But use larger variants for collision resistance
   - SHA-3 family also quantum-resistant with appropriate sizes
3. **Double MAC key sizes** where feasible
   - HMAC-SHA256 with 256-bit keys
   - Poly1305 remains secure (not searchable)

Current Practice	Quantum-Safe	Migration Effort
AES-128	AES-256	Change parameter
3DES	AES-256	Algorithm replacement
SHA-1 (broken anyway)	SHA-384	Algorithm replacement
SHA-256	SHA-256 (preimage)	None needed
	SHA-384 (collision)	Change parameter
HMAC-SHA256	HMAC-SHA256	None (key size ok)
ChaCha20	ChaCha20	Double key size

Table 8: Symmetric crypto migration is straightforward

The Bottom Line on Grover Grover's algorithm is the "good news" story of quantum cryptanalysis:

- **Predictable impact:** Exactly square root speedup
- **Simple mitigation:** Double key sizes
- **Already implemented:** AES-256 is standard

- **No algorithm changes:** Same AES, just bigger keys
- **Performance neutral:** Hardware acceleration handles it
- **Future-proof:** No known quantum algorithm beats Grover for unstructured search

Example 22 (Cost of Quantum-Proofing Symmetric Crypto)**.** *Large cloud provider migrating to quantum-safe symmetric crypto:*

- *Current: 10 million AES-128 keys in HSMs*
- *Migration: Update to AES-256*
- *Storage cost: 128 bits × 10M = 160 MB additional*
- *Performance impact: 0% (AES-NI handles both identically)*
- *Development effort: Update configuration files*
- *Timeline: Can be done incrementally*

Compare to RSA → PQC migration: New algorithms, protocols, libraries, testing...

Remark 14 (A Reassuring Perspective)**.** *If Grover's algorithm were the only quantum threat to cryptography, the transition would be trivial. Update some configuration files, double some parameters, and we're done. The real challenge isn't symmetric crypto—it's the complete algorithmic replacement required for public key systems. Grover is quantum computing's warning shot, not its killing blow.*

This manageable threat stands in stark contrast to the existential crisis Shor's algorithm poses for public key cryptography. While we must completely rebuild our asymmetric cryptographic infrastructure, symmetric cryptography needs only minor adjustments. This is why post-quantum cryptography focuses almost exclusively on replacing RSA, Diffie-Hellman, and elliptic curves—the symmetric primitives will survive with minimal changes.

Having examined how quantum algorithms threaten current cryptography, we now turn to the broader landscape of quantum-safe solutions and the crucial differences between physical and mathematical approaches to security.

3.2 PQC vs QKD: Complementary Not Competing

3.2.1 Fundamental Differences

A dangerous misconception pervades discussions of quantum-safe security: that we must choose between Quantum Key Distribution (QKD) and Post-Quantum Cryptography (PQC). This false dichotomy misunderstands both technologies. QKD and PQC solve different problems, operate on different principles, and have different failure modes. Most critically, QKD **cannot** provide digital signatures—meaning PQC is mandatory regardless of QKD deployment. Understanding these fundamental differences is crucial for any serious quantum security strategy.

Security Paradigms: Unconditional vs Computational The core distinction lies in what provides the security guarantee:

Definition 12 (Information-Theoretic Security (QKD))**.** *Security guaranteed by the laws of physics, independent of computational power:*

- *Based on: Quantum mechanics (no-cloning theorem, measurement disturbance)*

- *Broken by: Violating physics or compromising the physical channel*
- *Computational power: Irrelevant—infinite computation doesn't help*

Definition 13 (Computational Security (PQC))**.** *Security guaranteed by computational hardness, assuming bounded resources:*

- *Based on: Mathematical problems believed hard for quantum computers*
- *Broken by: New algorithms or sufficient computational power*
- *Computational power: Central—more computation weakens security*

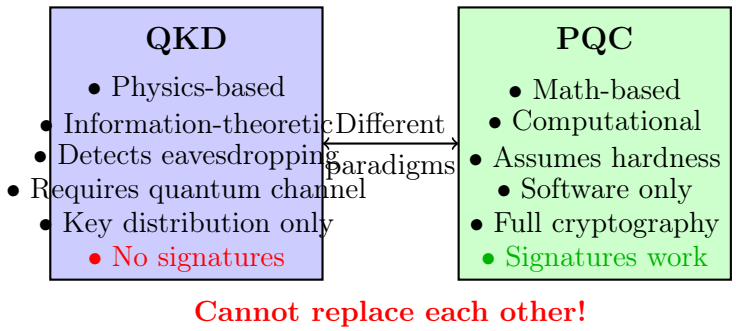


Figure 15: QKD and PQC: Fundamentally different approaches to quantum-safe security

Neither paradigm is inherently superior—they protect against different threat models:

- **QKD wins when:** Adversary has unlimited computation but must obey physics
- **PQC wins when:** You need signatures, internet-scale deployment, or software-only solutions
- **Both needed when:** Protecting ultra-high-value secrets for decades

Physical vs Mathematical Protection The infrastructure requirements create a stark divide:

Requirement	QKD	PQC
Physical channel	Dedicated fiber/free-space	Any channel
Distance limit	∼100-200 km (fiber)	Unlimited
Trusted repeaters	Required for long distance	Not needed
Quantum hardware	Single-photon sources/detectors	None
Classical hardware	Control electronics	Standard CPU
Cost per link	$100,000+	$0 (software)
Deployment time	Months (fiber installation)	Minutes (software update)
Maintenance	Specialized technicians	Standard IT

Table 9: Infrastructure requirements: QKD needs physics, PQC needs mathematics

Example 23 (Real QKD Deployment)**.** *Swiss government QKD network protecting Geneva elections:*

- *Distance: 29 km dedicated fiber*
- *Cost: $10 million+*

- *Speed: 1.4 Mbps key generation*
- *Availability: 99.9% (weather affects free-space backup)*
- *Limitation: Only connects specific government buildings*
- *Still needs: Classical crypto for authentication and signatures!*

The Scalability Divide The physical nature of QKD creates fundamental scalability limits:

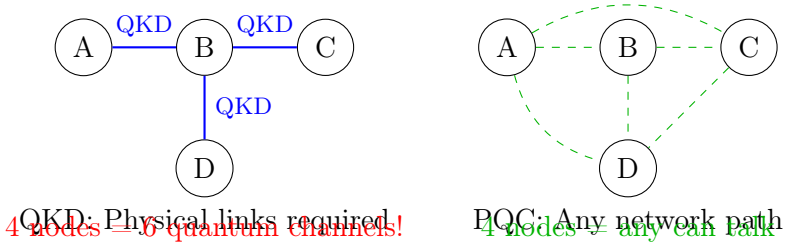


Figure 16: QKD requires dedicated point-to-point links; PQC works over any network

For n parties to communicate with QKD:

- Required quantum channels: $\binom{n}{2} = \frac{n(n-1)}{2}$
- For 1,000 users: 499,500 quantum channels needed!
- Alternative: Trusted relay nodes (but this weakens security model)
- Reality: QKD works for specific high-value point-to-point links

The Authentication Paradox Here's the dirty secret of QKD that vendors don't emphasize:

Theorem 5 (QKD Authentication Requirement)**.** *QKD is* ***unconditionally secure*** *against eavesdropping but* ***completely insecure*** *against man-in-the-middle attacks without pre-existing authentication.*

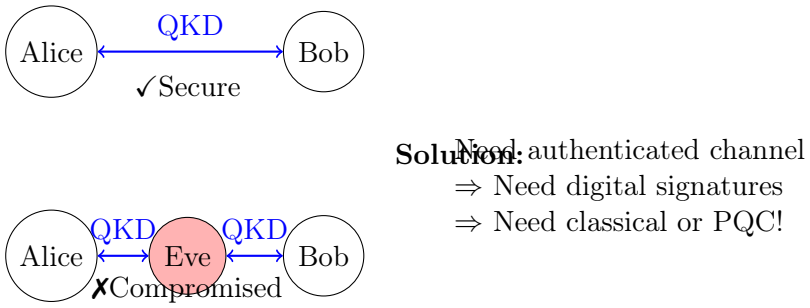


Figure 17: QKD's bootstrapping problem: Secure key exchange requires prior authentication

The authentication options:

1. **Pre-shared keys:** Defeats the purpose (if you can share keys, why QKD?)
2. **Classical signatures:** Vulnerable to quantum computers
3. **PQC signatures:** Quantum-safe but admits QKD isn't sufficient alone

The Missing Piece: Digital Signatures This is the killer limitation that many overlook:

Remark 15 (Critical Limitation)**.** ***QKD cannot, even in principle, provide digital signatures.*** *Signatures require:*

- *Public verifiability (anyone can verify)*
- *Non-repudiation (signer cannot deny)*
- *Transferability (signature remains valid when forwarded)*

QKD provides none of these. It only distributes secret keys between two parties.

Consider what needs signatures in practice:

- **Software updates:** How do you verify legitimate code?
- **Certificates:** How do you authenticate websites?
- **Financial transactions:** How do you prove authorization?
- **Legal documents:** How do you create binding agreements?
- **Email:** How do you verify sender identity?

Example 24 (The Signature Gap)**.** *Scenario: Bank wants quantum-safe communication with customers*

- *QKD can: Distribute keys to encrypt account data*
- *QKD cannot: Sign transaction authorizations*
- *QKD cannot: Authenticate the bank's website*
- *QKD cannot: Verify customer's identity*
- *Result: Must use PQC for signatures regardless*

Practical Deployment Reality Let's be brutally honest about deployment challenges:

Factor	QKD Reality	PQC Reality
Cost	\$50k-500k per link	Free (software update)
Distance	100km without repeaters	Global (Internet)
Speed	kbps to Mbps key rate	Gbps data rate
Integration	Requires new infrastructure	Uses existing networks
Maintenance	Quantum physicists on call	Standard IT staff
Reliability	99-99.9% (environmental)	99.999%+ (software)
Scalability	Point-to-point only	Billions of endpoints

Table 10: Deployment reality: QKD for specific links, PQC for everything

Example 25 (Cost Analysis: Securing a Corporate Network)**.** *Medium enterprise with 10 offices wanting quantum-safe security:*

QKD Approach:

- *45 quantum channels needed (full mesh)*
- *Cost: 45 × \$200k = \$9 million*

- *Time: 6-12 months fiber installation*
- *Limitation: Still need PQC for signatures, remote workers, cloud*

PQC Approach:

- *Software update to existing VPN infrastructure*
- *Cost: $0 (open source) to $100k (commercial)*
- *Time: 1-2 months testing and deployment*
- *Coverage: All employees, including remote*

The Bottom Line The fundamental differences make QKD and PQC complementary, not competing:

1. **QKD provides:** Information-theoretic security for key distribution between fixed points
2. **QKD requires:** Dedicated infrastructure, authentication, and distance limits
3. **PQC provides:** Computational security for all cryptographic primitives at Internet scale
4. **PQC requires:** Faith in mathematical hardness assumptions

Most crucially: **You cannot deploy QKD without also deploying PQC** (for signatures and authentication). But you can deploy PQC without QKD.

Remark 16 (For the QKD Enthusiast)**.** *As physicists, we love QKD—it's beautiful physics with elegant security proofs. But engineering reality is harsh. QKD is like a Ferrari: amazing for specific routes but you can't drive it to the grocery store, it doesn't work in bad weather, and you still need a regular car for daily life. PQC is like upgrading your engine—less exotic but solves the actual problem for 99.9% of use cases.*

These fundamental differences shape where each technology excels—our next topic.

3.2.2 Use Case Analysis

Now that we understand the fundamental differences, let's be practical: when should you actually deploy QKD versus PQC? The answer isn't about which technology is "better"—it's about matching capabilities to requirements. QKD excels in specific, high-value scenarios where its unique properties justify the cost. PQC handles everything else, which turns out to be 99.9% of cryptographic applications.

Where QKD Excels: The Crown Jewels QKD makes sense when all of the following are true: the data value is extreme, the link is point-to-point, distance is limited, and you can afford dedicated infrastructure. These scenarios exist but are rare.

1. Government/Diplomatic Communications

Example 26 (Real Deployment: Chinese Quantum Network)**.** *Beijing to Shanghai quantum backbone (2,000 km):*

- *Purpose: Government and military communications*
- *Architecture: 32 trusted relay nodes*
- *Cost: $100+ million*

- *Key rate: 1 kbps end-to-end*
- *Justification: Nation-state adversaries with unknown capabilities*
- *Still uses: PQC for authentication at every node*

Why QKD makes sense here:

- Adversary capability: Assume computational breakthroughs
- Data lifetime: Decades of classification
- Fixed endpoints: Government facilities don't move
- Budget: National security justifies cost
- Failure impact: Could compromise nation

2. Financial System Backbone

Example 27 (Swiss Banking Network)**.** *Major Swiss banks quantum-secured backbone:*

- *Links: Zurich - Geneva - Basel*
- *Purpose: Inter-bank settlement systems*
- *Daily volume: $500+ billion*
- *QKD benefit: Detects tampering attempts*
- *Backup: Conventional encrypted links*
- *Reality: Marketing value exceeds security benefit*

The financial case is marginal:

- Pro: Enormous transaction values
- Pro: Fixed, known endpoints
- Con: Transactions need signatures (must use PQC anyway)
- Con: Most attacks are insider/software, not cryptographic
- Reality: Often deployed for reputation, not security

3. Critical Infrastructure Control

Example 28 (Power Grid Control)**.** *Utility company protecting SCADA systems:*

- *Link: Control center - Nuclear plant*
- *Distance: 50 km dedicated fiber*
- *Threat: State-sponsored infrastructure attack*
- *QKD advantage: Physical tamper detection*
- *Integration: QKD keys for SCADA encryption*
- *Challenge: Industrial systems rarely support QKD*

4. Ultra Long-Term Secrets

Secret Type	Protection Period	QKD?	Rationale
Nuclear weapons designs	50+ years	Yes	Existential threat
Intelligence sources	25-50 years	Yes	Human lives at risk
Diplomatic cables	25 years	Maybe	Historical precedent
Medical records	100 years	No	PQC sufficient
Financial records	7-10 years	No	PQC sufficient
Personal messages	Lifetime	No	PQC sufficient

Table 11: QKD justified only for extreme protection periods with extreme consequences

Where PQC Dominates: Everything Else For the vast majority of cryptographic applications, PQC isn't just adequate—it's the only viable option:

1. Internet-Scale Applications

- **HTTPS/TLS:** 100+ billion connections daily
  - QKD impossible: Can't have quantum channel to every server
  - PQC solution: Drop-in replacement for RSA/ECDH
- **Email Security:** 300+ billion emails daily
  - Need: End-to-end encryption and signatures
  - QKD limitation: No quantum path between arbitrary users
  - PQC solution: Quantum-safe S/MIME, PGP
- **Software Updates:** Every device, every app
  - Need: Verify authentic, unmodified code
  - QKD impossible: Can't distribute signing keys
  - PQC requirement: Digital signatures mandatory

2. Mobile and IoT Devices

Example 29 (Smartphone Cryptography)**.** *Your phone performs daily:*

- *1,000+ TLS connections*
- *100+ app signature verifications*
- *50+ authentication operations*
- *Multiple VPN tunnels*

QKD feasibility: 0% (no quantum hardware) PQC feasibility: 100% (software update)

3. Cloud Services

Modern cloud architecture makes QKD impossible:

- Dynamic endpoints (containers, lambdas)
- Global distribution (CDNs, edge computing)
- Multi-tenancy (shared infrastructure)
- API-driven (need signatures for every call)

4. Blockchain and Cryptocurrencies

Remark 17 (The Ultimate PQC Use Case)**.** *Cryptocurrencies face unique challenges:*

- *Signatures are permanent on-chain*
- *Breaking old signatures breaks current ownership*
- *Must protect against future quantum computers*
- *QKD impossible: Decentralized, global network*
- *Only solution: Migrate to PQC signatures before quantum computers arrive*

Decision Framework Use this practical framework to choose technologies:

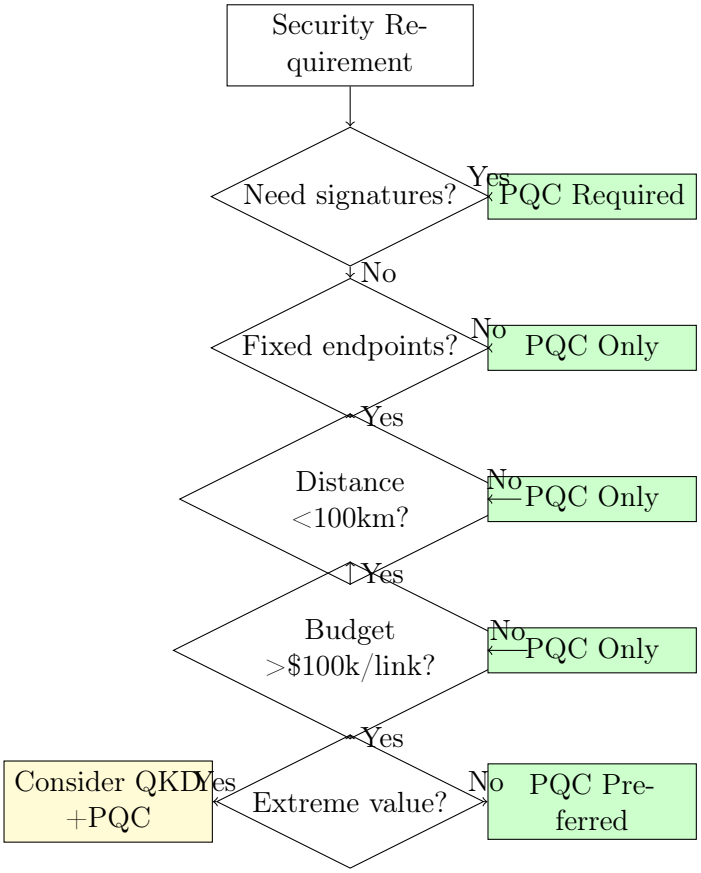


Figure 18: Decision tree: QKD only viable in very specific circumstances

Real-World Lessons Learning from actual deployments:
Successful QKD Deployments:

- **Geneva election system:** Fixed endpoints, extreme integrity requirements
- **Tokyo QKD network:** R&D testbed, not production
- **Chinese backbone:** Government resources, national priority
- Common factors: Fixed infrastructure, unlimited budget, specific threats

Failed QKD Projects:

- **European SECOQC:** Too expensive to maintain after research funding ended

- **Various bank trials:** No advantage over PQC for actual operations
- **Corporate deployments:** Complexity exceeded IT capabilities
- Common factors: Underestimated operational costs, overestimated threats

PQC Migration Experiences:

- **Google's CECPQ2:** Successfully tested PQC in Chrome with 0.3% overhead
- **Cloudflare:** Deployed PQC to millions of websites transparently
- **Signal Protocol:** Added PQC with no user-visible changes
- Common factors: Software-only deployment, incremental rollout, backward compatible

The Practical Reality Let's summarize with brutal honesty:

Application Category	QKD Viable?	PQC Required?
Web browsing (HTTPS)	No	Yes
Email encryption	No	Yes
Software signing	No	Yes
VPN/Remote access	No	Yes
Cloud services	No	Yes
IoT devices	No	Yes
Blockchain/Crypto	No	Yes
Mobile apps	No	Yes
Government backbone	Maybe	Yes
Banking backbone	Maybe	Yes
Critical infrastructure	Maybe	Yes
Diplomatic cables	Yes	Yes
Percentage of use cases	**<0.1%**	**100%**

Table 12: The stark reality: PQC is mandatory everywhere, QKD is optional in rare cases

Remark 18 (The Bottom Line)**.** *If you're asking "Should I use QKD or PQC?"—you need PQC. The question is whether you* also *need QKD for specific high-value links. For 99.9% of organizations, PQC alone provides excellent quantum-safe security. QKD is for the 0.1% with nation-state adversaries, unlimited budgets, and fixed infrastructure. Even then, you still need PQC for signatures.*

This analysis leads naturally to our next topic: how to combine both technologies when extreme security demands it.

3.2.3 Hybrid Approaches

Security professionals have a saying: "Defense in depth is the only defense." When protecting against quantum threats, combining QKD and PQC creates multiple independent security layers. If one fails—whether through algorithmic breakthrough, implementation flaw, or operational error—the other maintains protection. This belt-and-suspenders approach makes particular sense during the transition period when we're uncertain about both quantum computer timelines and PQC algorithm security.

Defense in Depth Strategy The hybrid philosophy is simple: force an attacker to break multiple, fundamentally different security mechanisms:

Definition 14 (Hybrid Quantum-Safe Security)**.** *Combine information-theoretic (QKD) and computational (PQC) security such that:*

- *Breaking the system requires breaking BOTH QKD AND PQC*
- *Different expertise needed: quantum physics vs. mathematics/algorithms*
- *Different attack vectors: physical channel vs. computational resources*
- *Failure modes are independent*

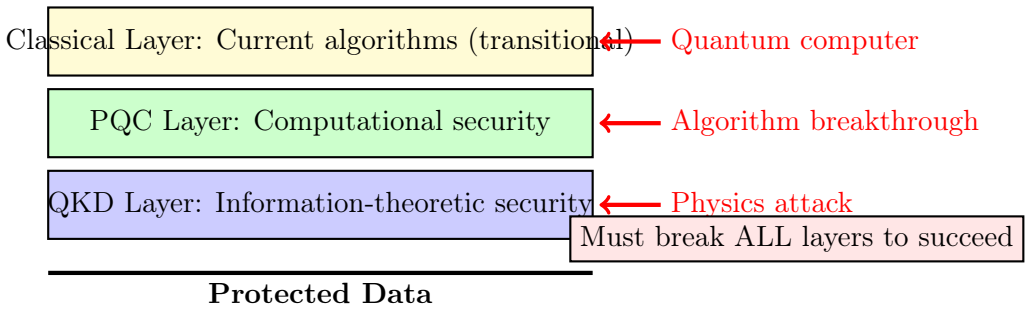


Figure 19: Defense in depth: Multiple independent security layers

Why this works:

- **QKD break requires:** Violating quantum mechanics or compromising every relay node
- **PQC break requires:** New mathematical algorithm or massive computation
- **Joint break probability:** $P(\text{QKD break}) \times P(\text{PQC break}) \ll$ either alone
- **Diverse expertise:** Few adversaries excel at both quantum physics and advanced mathematics

Hybrid Key Establishment The core protocol combines QKD and PQC key exchange:
Key combination methods:

1. **XOR Combination:** $K_{\text{final}} = K_{\text{QKD}} \oplus K_{\text{PQC}}$
   - Pro: Provably secure if either key is random
   - Con: Requires equal-length keys
2. **KDF Combination:** $K_{\text{final}} = \text{HKDF}(K_{\text{QKD}} || K_{\text{PQC}})$
   - Pro: Flexible output length
   - Pro: Includes context binding
   - Standard: NIST SP 800-56C
3. **Dual-Key Usage:** Use different keys for different purposes
   - K_{QKD} for bulk data encryption
   - K_{PQC} for authentication/integrity

Algorithm 7 Hybrid QKD-PQC Key Establishment

Require: QKD link between Alice and Bob
Require: PQC algorithms (e.g., Kyber for KEM, Dilithium for signatures)
Ensure: Shared secret key K_{final} secure against both quantum and algorithmic attacks

Phase 1: Authentication
Alice $\rightarrow$ Bob: Cert_A with PQC public key
Bob $\rightarrow$ Alice: Cert_B with PQC public key
Both verify certificates using PQC signatures
Phase 2: PQC Key Exchange
Bob generates $(pk_{\text{kem}}, sk_{\text{kem}}) \leftarrow \text{Kyber.KeyGen}()$
Bob $\rightarrow$ Alice: pk_{kem}, $\text{Sign}_{\text{Dilithium}}(pk_{\text{kem}}, sk_B)$
Alice: $(c, K_{\text{PQC}}) \leftarrow \text{Kyber.Encaps}(pk_{\text{kem}})$
Alice $\rightarrow$ Bob: c, $\text{Sign}_{\text{Dilithium}}(c, sk_A)$
Bob: $K_{\text{PQC}} \leftarrow \text{Kyber.Decaps}(c, sk_{\text{kem}})$
Phase 3: QKD Key Distribution
Run QKD protocol authenticated with K_{PQC}
Obtain K_{QKD} with information-theoretic security
Phase 4: Key Combination
$K_{\text{final}} = \text{KDF}(K_{\text{QKD}} || K_{\text{PQC}} || \text{transcript})$
Both parties have K_{final} secure against all known attacks

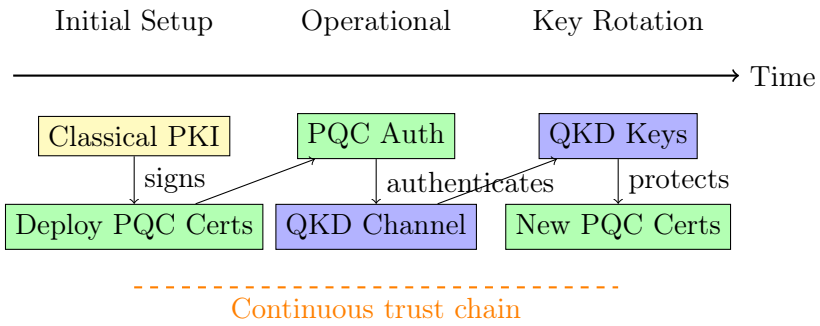


Figure 20: Bootstrapping trust: From classical to hybrid quantum-safe

Authentication Chains The circular dependency problem has an elegant solution:
This creates a trust ratchet:

1. Classical PKI bootstraps initial PQC certificates
2. PQC certificates authenticate QKD channels
3. QKD channels protect PQC key distribution
4. New PQC keys distributed via QKD-secured channels
5. System never depends solely on classical crypto after bootstrap

Practical Deployment Architectures Real-world hybrid deployments use tiered architectures:

Example 30 (Financial Sector Hybrid Architecture)**.** *Major bank implementing quantum-safe infrastructure:*

Tier 1 - Critical Core (QKD + PQC):

- *Links: Between main data centers*

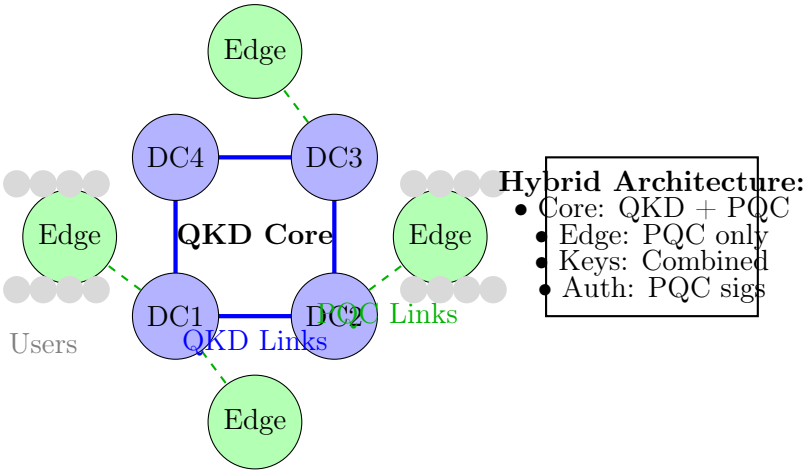


Figure 21: Typical hybrid deployment: QKD backbone with PQC everywhere

- *Purpose: Core banking systems, key management*
- *Protection: QKD for key distribution + PQC for authentication*
- *Cost: $5M for 4 sites*

Tier 2 - Extended Core (PQC only):

- *Links: Branch offices, disaster recovery sites*
- *Purpose: Transaction processing, backups*
- *Protection: PQC algorithms (Kyber + Dilithium)*
- *Cost: $100k for software upgrades*

Tier 3 - Customer Access (PQC only):

- *Links: Mobile apps, web banking*
- *Purpose: Customer transactions*
- *Protection: Hybrid classical-PQC during transition*
- *Cost: Included in normal development*

Migration Strategies The path to quantum-safe security isn't binary:

Future-Proofing Through Diversity Hybrid approaches provide insurance against multiple failure modes:

Remark 19 (Crypto-Agility is Key)**.** *The hybrid approach's greatest benefit isn't the current security—it's the* agility *to respond to future threats:*

- *New quantum algorithm breaks PQC? QKD maintains security*
- *Physical attack on QKD? PQC maintains security*
- *Better PQC algorithm developed? Deploy without disrupting QKD*
- *Improved QKD technology? Upgrade without changing PQC*

This agility is worth more than any specific algorithm choice.

Algorithm 8 Phased Hybrid Migration

Phase 1: Inventory and Planning (Now - 2025)
  Identify all cryptographic dependencies
  Classify by risk and lifetime
  Design target architecture
Phase 2: PQC Foundation (2025-2027)
  Deploy PQC alongside classical algorithms
  $K = \text{KDF}(K_{\text{classical}} || K_{\text{PQC}})$
  Test performance and compatibility
Phase 3: QKD Pilots (2026-2028)
  Install QKD for highest-value links
  Integrate with PQC authentication
  Measure operational costs
Phase 4: Full Hybrid (2027-2030)
  QKD backbone operational
  PQC mandatory for all systems
  Classical crypto in compatibility mode only
Phase 5: Quantum-Safe Only (2030+)
  Disable classical algorithms
  QKD + PQC for critical systems
  PQC only for general use

Threat Scenario	**Classical**	**PQC Only**	**Hybrid**
Quantum computer arrives	Broken	Secure	Secure
PQC algorithm broken	Broken*	Broken	Secure
Implementation flaw	Vulnerable	Vulnerable	Partially secure
Physical compromise	Secure	Secure	Degraded
Zero-day in protocol	Vulnerable	Vulnerable	Partially secure

*During transition period where both are used

Table 13: Hybrid approach provides maximum resilience

The Realistic Bottom Line Hybrid QKD-PQC deployments make sense for:

- Organizations with nation-state adversaries
- Infrastructure protecting millions of people
- Secrets requiring 30+ year protection
- Scenarios where trust in any single approach is insufficient

For everyone else, PQC alone provides excellent quantum resistance at reasonable cost.

Example 31 (Decision Matrix)**.**

Organization Type	***Recommendation***	***Rationale***
Intelligence agency	*QKD + PQC hybrid*	*Maximum adversary capability*
Major bank (core)	*QKD + PQC hybrid*	*Systemic risk, fixed infrastructure*
Major bank (branches)	*PQC only*	*Cost-benefit doesn't justify QKD*
Healthcare system	*PQC only*	*Long-term privacy, but not extreme*
Enterprise	*PQC only*	*Standard threat model*
Small business	*PQC only*	*Resource constraints*

The hybrid approach represents the ultimate in quantum-safe security, but comes with corresponding complexity and cost. For the few organizations that need it, the investment provides unparalleled protection. For everyone else, well-implemented PQC offers quantum safety without the quantum physics.

3.3 PQC Algorithm Families Overview

3.3.1 Lattice-Based Cryptography

Here's a stunning fact: when the dust settled on NIST's post-quantum competition, three of the four selected algorithms were based on lattices. Not codes, not hashes, not multivariate polynomials—lattices. This mathematical structure, familiar to every physicist from crystallography and solid-state physics, has emerged as the dominant foundation for post-quantum cryptography. The reasons are compelling: lattices provide efficient algorithms for both encryption and signatures, enable advanced protocols like fully homomorphic encryption, and have resisted cryptanalysis for over 25 years.

Basic Concept: Physics Meets Cryptography A lattice is simply all integer linear combinations of basis vectors—a concept physicists encounter daily:

Definition 15 (Lattice (Informal))**.** *Given linearly independent vectors* $\mathbf{b}_1, ..., \mathbf{b}_n \in \mathbb{R}^n$*, the lattice is:*

$$\mathcal{L} = \left\{ \sum_{i=1}^{n} a_i \mathbf{b}_i : a_i \in \mathbb{Z} \right\}$$

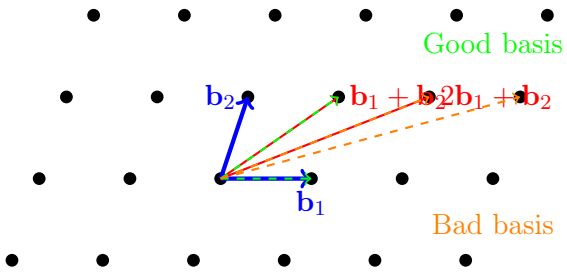


Figure 22: Same lattice, different bases: Finding the "good" basis from a "bad" one is cryptographically hard

For physicists, think of:

- **Crystal structure:** Atoms at lattice points
- **Reciprocal lattice:** Fourier transform preserves lattice structure
- **Brillouin zones:** Fundamental domains in lattice
- **Basis reduction:** Finding primitive unit cell

The cryptographic magic: While you understand lattices in 2D or 3D, cryptography uses lattices in 500+ dimensions where geometric intuition fails and problems become exponentially hard.

Capability	Lattice	Code	Hash	Multivariate
Encryption	✓	✓	✗	✓
Signatures	✓	✗	✓	✓
Key Exchange	✓	✓	✗	✗
Homomorphic Encryption	✓	✗	✗	✗
Zero-Knowledge Proofs	✓	✗	✗	Limited
Identity-Based Crypto	✓	✗	✗	✗
NIST Selections	**3**	**0**	**1**	**0**

Table 14: Lattices uniquely provide all cryptographic primitives efficiently

The Versatility Advantage Unlike other post-quantum approaches, lattices enable the complete cryptographic toolkit:

This versatility isn't accidental—it stems from the rich algebraic structure:

- **Additive structure:** Enables homomorphic properties
- **Noise tolerance:** Small errors don't break the scheme
- **Worst-case hardness:** Average instances as hard as worst instances
- **Quantum resistance:** No periodic structure to exploit

Why Lattices Dominate PQC The dominance isn't just about capability—it's about practical engineering:

1. **Security Foundation: 25+ Years Strong**
   - NTRU proposed in 1996—still unbroken
   - LWE introduced in 2005—security only strengthened
   - Extensive cryptanalysis by worldwide community
   - No quantum algorithm better than classical
2. **Efficiency: Reasonable Trade-offs**
   - Key sizes: 1-10 KB (vs. MB for codes)
   - Speed: Often faster than RSA at same security
   - Simple operations: Just integer arithmetic
   - Parallelizable: Suits modern hardware
3. **Flexibility: Multiple Constructions**
   - Standard lattices: Maximum security confidence
   - Ideal lattices: Smaller keys via algebra
   - Module lattices: Balance between security and efficiency
   - Each suited for different use cases

Remark 20 (The Goldilocks Zone)**.** *Lattices occupy a "Goldilocks zone" in PQC:*

- *Codes: Ultra-conservative but keys too large (MB)*
- *Multivariate: Efficient but keeps getting broken*
- *Isogenies: Elegant but catastrophically broken (SIKE)*
- *Lattices: Just right—secure, efficient, and versatile*

NIST's Lattice Winners Three different lattice approaches won for good reasons:

1. **CRYSTALS-Kyber (Now ML-KEM): The Encryption Champion**

- **Based on:** Module-LWE (Module Learning With Errors)
- **Strengths:** Fast, reasonable key sizes, simple implementation
- **Key sizes:** 800-1,568 bytes (public), 1,632-3,168 bytes (ciphertext)
- **Use case:** General-purpose encryption, TLS key exchange
- **Why chosen:** Best balance of security and performance

2. **CRYSTALS-Dilithium (Now ML-DSA): The Signature Workhorse**

- **Based on:** Module-LWE with Fiat-Shamir
- **Strengths:** Fast signing and verification, reasonable sizes
- **Signature size:** 2,420-4,595 bytes
- **Use case:** General-purpose signatures, certificates
- **Why chosen:** Best overall signature performance

3. **Falcon: The Compact Alternative**

- **Based on:** NTRU lattices with GPV framework
- **Strengths:** Smallest signatures (666-1,280 bytes)
- **Weakness:** Complex implementation, needs floating-point
- **Use case:** Constrained environments, blockchain
- **Why chosen:** Diversity and compact signatures

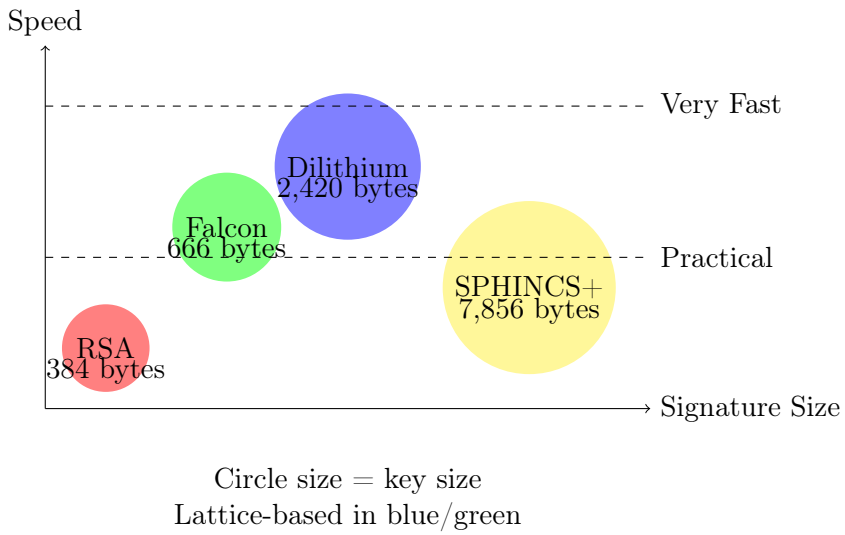


Figure 23: Performance vs. size trade-offs: Lattice algorithms dominate the practical region

Why Three Different Approaches? NIST's selection philosophy: diversity protects against catastrophic breaks:

1. **Kyber + Dilithium:** Share Module-LWE foundation
   - Pro: Consistent security analysis
   - Pro: Similar implementation techniques
   - Risk: Single breakthrough affects both
2. **Falcon:** Different lattice type (NTRU)
   - Pro: Independent security foundation
   - Pro: Optimal signature sizes
   - Con: Implementation complexity
3. **Together:** Cover all use cases
   - Need speed? → Dilithium
   - Need small signatures? → Falcon
   - Need encryption? → Kyber
   - Need maximum confidence? → Use hybrid modes

Setting the Stage This overview barely scratches the surface. In Section 3, we'll dive deep into:

- The mathematics: Why are lattice problems hard?
- LWE construction: How noise creates security
- Implementation details: From theory to practice
- Security analysis: Concrete parameters and attacks

Remark 21 (For the Physics-Minded)**.** *Your crystallography background provides perfect intuition:*

- *Lattice reduction -> Finding primitive unit cell*
- *LWE -> Crystal with thermal noise*
- *Ring-LWE -> Exploiting crystal symmetries*
- *Attacks -> Finding order in apparent disorder*

The deep dive will make these analogies precise.

Lattice-based cryptography represents our best hope for a quantum-safe future: versatile enough to replace all current systems, efficient enough for practical deployment, and secure enough to have survived decades of attack. The fact that it connects to physics makes it particularly appealing for this audience—you already have the intuition, we just need to build the cryptographic layer on top.

3.3.2 Code-Based Cryptography

If you want the most conservative possible approach to post-quantum cryptography, look no further than code-based systems. The McEliece cryptosystem, proposed in 1978, has survived 45 years of cryptanalysis—longer than RSA, longer than elliptic curves, longer than any other public-key system. It's based on error-correcting codes, a cornerstone of information theory that physicists know from quantum error correction, communication theory, and statistical mechanics. There's just one problem: the keys are enormous.

From Communications to Cryptography Error-correcting codes were designed to protect data from noise, not adversaries. The cryptographic insight: decoding a random linear code is NP-complete.

Definition 16 (The Syndrome Decoding Problem)**.** *Given:*

- *A random binary matrix* $H \in \{0,1\}^{r \times n}$ *(parity check matrix)*
- *A syndrome* $s \in \{0,1\}^r$
- *An integer* w *(error weight)*

Find: An error vector $e \in \{0,1\}^n$ *with exactly* w *ones such that* $He^T = s$*.*

For physicists, this connects to:

- **Statistical mechanics:** Finding low-energy configurations
- **Quantum error correction:** Syndrome measurements and recovery
- **Information theory:** Channel capacity and decoding limits
- **Compressed sensing:** Sparse signal recovery

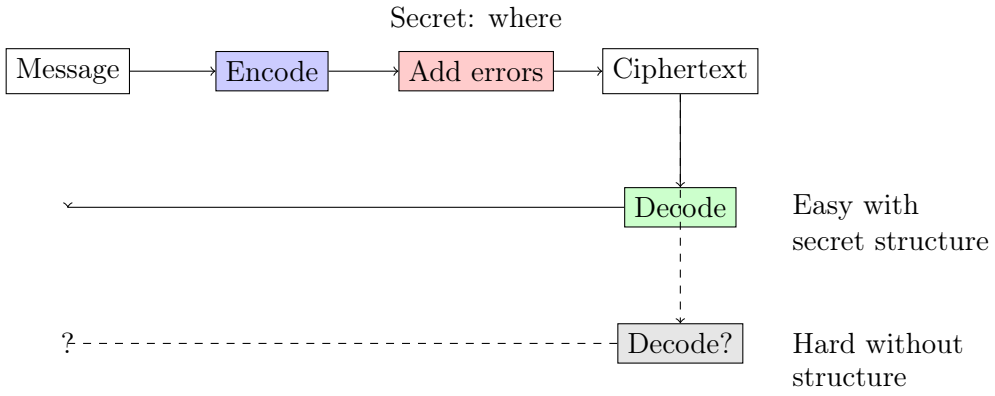


Figure 24: Code-based encryption: Hide efficient decoding structure in random-looking code

The cryptographic magic: Use a code with hidden structure that allows efficient decoding, but looks random to attackers.

The McEliece Cryptosystem McEliece's 1978 construction remains unbroken—a record unmatched in cryptography:

Security relies on two problems:

1. **Generic decoding:** Removing unknown errors from random code

Algorithm 9 McEliece Encryption (Simplified)

Key Generation:
  Choose a Goppa code with generator matrix G
  Choose random invertible matrices S and P
  Public key: $G' = SGP$ (looks random)
  Private key: (S, G, P) and Goppa polynomial
Encryption:
  Message: $m \in \{0,1\}^k$
  Random error: $e \in \{0,1\}^n$ with weight t
  Ciphertext: $c = mG' + e$
Decryption:
  Compute $cP^{-1} = mSG + eP^{-1}$
  Use Goppa decoding to remove error
  Recover m by inverting S

2. **Distinguishing:** Goppa codes look random after transformation

Example 32 (Concrete Parameters)**.** *Classic McEliece for 128-bit quantum security:*

- *Code parameters:* $[3488, 2720, 129]$ *Goppa code*
- *Public key size: 261,120 bytes (255 KB)*
- *Ciphertext expansion: 128 bytes*
- *Private key: 6,492 bytes*
- *Compare RSA-2048: 256 bytes public key*

Quantum Resistance: Information-Theoretic Near-Optimality Code-based cryptography resists quantum attack for fundamental reasons:

1. **No Hidden Structure:** Generic codes have no periodicity or algebra to exploit
2. **Grover is Optimal:** Best quantum algorithm is Grover's search
3. **Well-Understood Bounds:** Information theory provides tight limits

Theorem 6 (Quantum Complexity of Syndrome Decoding)**.** *For a random (n, k) code with t errors:*

- *Classical:* $\mathcal{O}(2^{kt/n})$ *operations*
- *Quantum:* $\mathcal{O}(2^{kt/2n})$ *operations (Grover speedup)*
- *No exponential quantum advantage known*

The Achilles' Heel: Key Size The fatal flaw that prevented NIST selection:
Why are keys so large?

- Need large random matrix to hide structure
- Matrix must be dense (not sparse) for security
- No algebraic compression possible

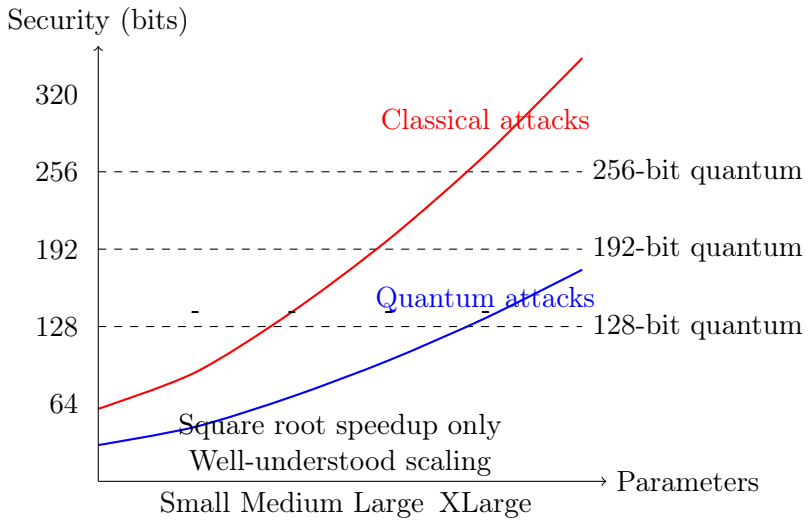


Figure 25: Code-based security: Quantum provides only Grover speedup

Algorithm	Public Key	Ciphertext	Security
RSA-2048	256 B	256 B	112-bit classical
ECDH-P256	64 B	64 B	128-bit classical
Kyber-768	1,184 B	1,088 B	128-bit quantum
Dilithium-3	1,952 B	3,293 B (sig)	128-bit quantum
Classic McEliece	**261,120 B**	128 B	128-bit quantum

Table 15: The key size catastrophe: 1000× larger than current systems

- Linear in security parameter

Example 33 (Bandwidth Impact)**.** *TLS handshake with Classic McEliece:*

- *Current (ECDH): 1.5 KB total*
- *With McEliece: 262 KB total*
- *175× increase in handshake size*
- *For Google: 30+ petabytes/day extra*

NIST Competition Results Classic McEliece made it to the final round but wasn't selected:

Strengths recognized by NIST:

- Longest security track record
- Simple security assumption
- Fast operations
- Small ciphertext expansion

Why not selected:

- Public keys too large for most applications
- Bandwidth costs prohibitive
- Storage requirements unrealistic

- Lattice schemes offer better trade-offs

NIST's comment: "Classic McEliece has unmatched security confidence but deployment challenges make it unsuitable as a general-purpose standard."

The Trade-off Dilemma Code-based cryptography presents a stark choice:

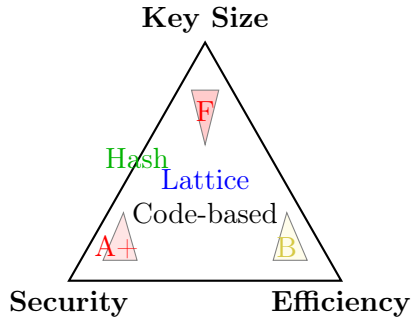


Figure 26: The impossible triangle: Code-based crypto excels at security and speed but fails at key size

Where code-based might still make sense:

- **High-bandwidth scenarios:** Backbone links with fiber capacity
- **Paranoid security:** When you absolutely cannot risk algorithmic breaks
- **Future storage:** If storage becomes essentially free
- **Hybrid modes:** Combine with lattice for diversity

Remark 22 (The Conservation Law of Cryptography)**.** *Code-based cryptography illustrates a fundamental trade-off: you can have any two of small keys, high security, computational efficiency, but not all three. McEliece chose security and efficiency, sacrificing key size. Lattices found a better balance for most applications, but at the cost of newer, less-tested mathematical assumptions. There's no free lunch in post-quantum cryptography.*

Example 34 (Potential Future: Compressed Codes)**.** *Research continues on reducing key sizes:*

- *Quasi-cyclic codes: Some structure, smaller keys*
- *List decoding: Higher error rates, smaller codes*
- *New code families: Moderate improvements*
- *Reality check: Still 10-100× larger than lattices*

Code-based cryptography remains the "gold standard" for conservative security—if you can afford the keys. For special applications where bandwidth and storage aren't constraints, it provides unmatched confidence. For everyone else, the key sizes remain prohibitive. As NIST concluded, it's a specialized tool rather than a general solution, but one that provides an important existence proof: we have at least one approach that has survived everything cryptanalysts have thrown at it for nearly half a century.

3.3.3 Hash-Based Signatures

Here's a beautiful fact: you can build quantum-secure digital signatures using nothing but hash functions. No number theory, no lattices, no algebraic structures—just the humble one-way function. This minimalist approach offers ultimate confidence: if SHA-256 remains one-way against quantum computers (with appropriate parameter increases), then hash-based signatures remain secure. There's just one catch: this approach **only** works for signatures, never encryption. SPHINCS+, the hash-based winner in NIST's competition, represents the culmination of 40 years of research into turning this theoretical possibility into practical reality.

Building from One-Way Functions The journey starts with Lamport's elegant one-time signature from 1979:

Algorithm 10 Lamport One-Time Signature (Simplified)

Key Generation:
  For each bit position $i = 1$ to n:
    Choose random values $x_i^0, x_i^1 \in \{0,1\}^{256}$
    Compute $y_i^0 = H(x_i^0)$, $y_i^1 = H(x_i^1)$
  Private key: $\{x_i^0, x_i^1\}_{i=1}^n$
  Public key: $\{y_i^0, y_i^1\}_{i=1}^n$
Sign message $m = m_1 m_2 ... m_n$:
  For each bit m_i: reveal $x_i^{m_i}$
  Signature: $(x_1^{m_1}, x_2^{m_2}, ..., x_n^{m_n})$
Verify signature $(s_1, ..., s_n)$ on m:
  Check: $H(s_i) = y_i^{m_i}$ for all i

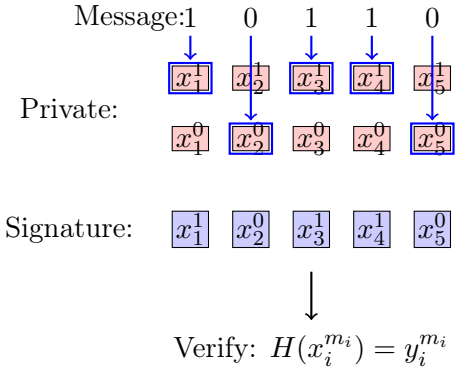


Figure 27: Lamport signatures: Reveal half of the secrets based on message bits

Why "one-time"? Signing two different messages reveals different private key pieces, allowing forgery:

- Sign $m = 1010...$: Reveals $(x_1^1, x_2^0, x_3^1, x_4^0, ...)$
- Sign $m' = 1110...$: Reveals $(x_1^1, x_2^1, x_3^1, x_4^0, ...)$
- Attacker knows both x_2^0 and x_2^1: Can forge bit 2 as either value

From One-Time to Many-Time: The Merkle Tree Merkle (1979) showed how to sign many messages by organizing one-time keys in a tree:

To sign the i-th message:

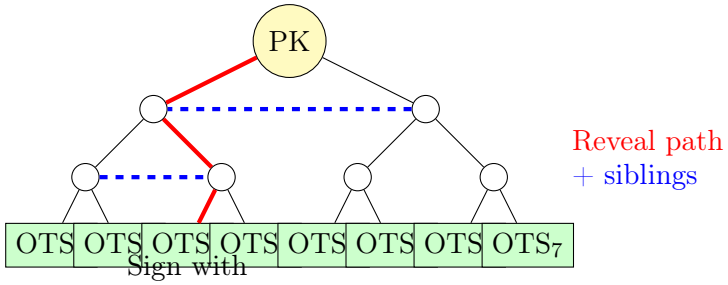


Figure 28: Merkle tree: One public key authenticates many one-time signatures

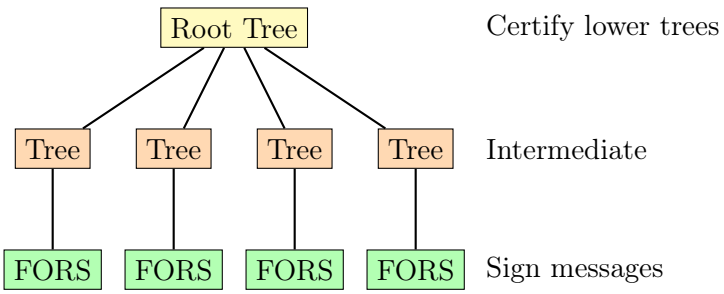
1. Use OTS_i to sign the message
2. Reveal authentication path from leaf to root
3. Verifier reconstructs root, checks against public key

The state management nightmare:

- **Critical:** Never reuse an OTS key
- **Problem:** Must track which keys are used
- **Distributed systems:** State synchronization across nodes
- **Backup/restore:** State loss = security catastrophe
- **Result:** Beautiful in theory, dangerous in practice

The Stateless Revolution: SPHINCS+ SPHINCS+ (2019) eliminates state by using "few-time" signatures and hypertrees:

Definition 17 (Few-Time Signature)**.** *Instead of one-time signatures that break after two uses, use schemes that tolerate hundreds of uses before security degrades. Combined with massive trees, collision probability becomes negligible.*



Total signatures: 2^{64} (effectively unlimited)

Figure 29: SPHINCS+ hypertree: Stateless through massive tree and few-time signatures

Key innovations:

- **FORS:** Few-time signatures tolerating many uses
- **Hypertree:** Multiple tree layers for efficiency
- **Randomized signing:** Pseudo-random index selection
- **No state needed:** Deterministic from message + key

NIST Selection: The Conservative Backup SPHINCS+ was selected as the fourth algorithm for specific reasons:

Algorithm	Type	Signature Size	Speed
Dilithium3	Lattice	3,293 B	2,000 sig/sec
Falcon-512	Lattice	666 B	5,000 sig/sec
SPHINCS+-128f	Hash	17,088 B	100 sig/sec
SPHINCS+-128s	Hash	7,856 B	15 sig/sec

Table 16: SPHINCS+ trades size and speed for conservative security

Why NIST included SPHINCS+:

- **Diversity:** Only non-lattice signature scheme
- **Conservative:** Based solely on hash functions
- **Backup plan:** If lattices break, we have an alternative
- **Special use cases:** Firmware updates, root certificates

NIST's comment: "SPHINCS+ provides an important hedge against advances in lattice cryptanalysis, despite larger signatures and slower performance."

The Signature-Only Limitation Critical point often misunderstood: hash-based techniques **cannot** provide encryption.

Why no encryption?

- **One-way only:** Hashes can't be inverted for decryption
- **No trapdoor:** Missing the asymmetry needed for public-key encryption
- **Information theoretic:** Signatures destroy information, encryption preserves it

Remark 23 (A Fundamental Divide)**.** *The physics analogy is thermodynamics:*

- *Signatures = Irreversible process (increase entropy)*
- *Encryption = Reversible process (preserve information)*

Hash functions are computational "heat engines"—great for irreversible signatures, useless for reversible encryption.

Use cases where hash-based excels:

1. **Firmware signing:** Size less critical, conservative security vital
2. **Root certificates:** Sign once, verify millions of times
3. **Long-term archives:** Documents valid for 50+ years
4. **Blockchain:** Where signature size is less critical than security
5. **Emergency backup:** If all lattice schemes catastrophically fail

Example 35 (SPHINCS+ in Practice)**.** *Software update signing:*

- *Current (RSA-2048): 256-byte signature*
- *SPHINCS+-128s: 7,856-byte signature*

- *Impact: 30× larger, but...*
- *Update size: 100 MB typical*
- *Overhead: 0.008% (negligible)*
- *Benefit: Quantum-proof based only on SHA-256*

The Bottom Line Hash-based signatures occupy a unique niche:

- **Most conservative:** Security = existence of one-way functions
- **Limited scope:** Signatures only, no encryption
- **Poor performance:** Large and slow
- **Critical role:** Ultimate backup plan

Think of SPHINCS+ as cryptographic insurance: you hope never to need it, but if lattice cryptanalysis achieves a breakthrough, hash-based signatures ensure digital signatures don't disappear entirely. For daily use, stick with Dilithium or Falcon. For ultra-critical, long-term signatures where conservative security trumps efficiency, SPHINCS+ provides unmatched confidence.

Remark 24 (Historical Note)**.** *Lamport and Merkle invented these techniques before RSA! They were considered impractical curiosities for decades. The quantum threat transformed them from academic exercises into our safety net—a reminder that in cryptography, no idea should be dismissed entirely.*

3.3.4 Other Approaches

Cryptography has a graveyard filled with brilliant ideas that didn't survive contact with cryptanalysis. For every RSA or AES that stands the test of time, dozens of schemes lie broken and abandoned. The post-quantum landscape is particularly littered with casualties—schemes that looked promising for years before catastrophic breaks emerged. Understanding these failures isn't just historical curiosity; it's essential for recognizing what makes a cryptographic foundation trustworthy. The spectacular collapse of SIKE in 2022 serves as the most recent and dramatic reminder.

Multivariate Polynomials: The Eternal Optimist The idea is seductively simple: solving systems of multivariate quadratic equations over finite fields is NP-complete. Why not build cryptography on this?

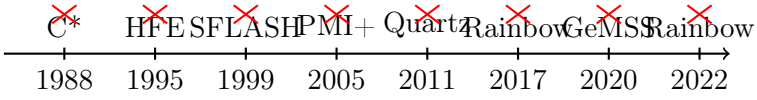
Definition 18 (Multivariate Quadratic (MQ) Problem)**.** *Given m quadratic polynomials in n variables over $\mathbb{F}_q$:*

$$p_i(x_1, ..., x_n) = \sum_{j \leq k} a_{ijk} x_j x_k + \sum_j b_{ij} x_j + c_i = 0$$

Find: $(x_1, ..., x_n) \in \mathbb{F}_q^n$ satisfying all equations.

The cryptographic construction:

- **Private key:** Hidden structure making system easy to solve
- **Public key:** Transformed system looking random
- **Problem:** The hidden structure keeps getting exposed



The Multivariate Graveyard: 35 years of broken schemes

Figure 30: Multivariate schemes: A consistent pattern of initial promise followed by cryptanalytic breaks

Why they keep failing:

1. **Structure leakage:** Hidden structure needed for efficiency is hard to hide completely
2. **Algebraic attacks:** Gröbner basis algorithms keep improving
3. **Parameter trap:** Make it secure → huge keys; make keys reasonable → breakable
4. **Differential attacks:** Structure often visible through derivatives

Example 36 (Rainbow's Fall)**.** *Rainbow was a NIST finalist until 2022:*

- *Promise: Small signatures (66 bytes!)*
- *Structure: Layers of oil-vinegar equations*
- *Break: Beullens found a way to peel off layers*
- *Result: All proposed parameters broken*
- *Lesson: 20+ years of study wasn't enough*

The Isogeny Saga: Beauty and Catastrophe Isogeny-based cryptography was the newest, most mathematically sophisticated approach—until it spectacularly imploded.

Definition 19 (Isogenies (Simplified))**.** *An isogeny is a structure-preserving map between elliptic curves. The cryptographic idea:*

- *Start with elliptic curve* E_0
- *Secret: A path through the isogeny graph*
- *Public: The ending curve* E_A
- *Hard problem: Find the path from* E_0 *to* E_A

SIKE (Supersingular Isogeny Key Encapsulation):

- Submitted to NIST in 2017
- Made it to Round 4 (finalist)
- Smallest keys among all candidates (335 bytes!)
- Based on 10+ years of research
- July 2022: Destroyed in 1 hour on a laptop

The Devastating Attack:
Castryck and Decru's breakthrough:

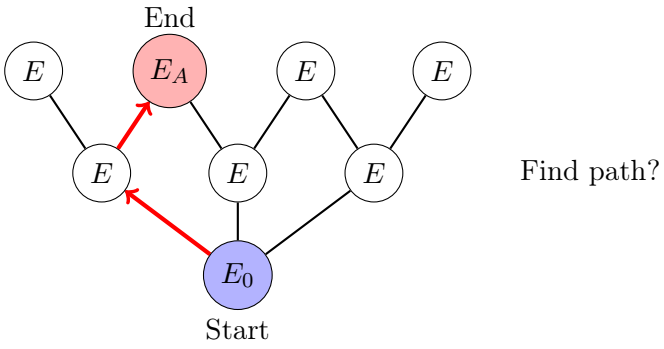


Figure 31: Isogeny graph: Beautiful mathematics, fatal flaw

1. Found auxiliary points with special properties
2. Used Richelot isogenies and abelian surfaces
3. Converted path-finding to linear algebra
4. Complexity: Polynomial instead of exponential!

Remark 25 (The Shock)**.** *The attack was so unexpected that the SIKE team's initial response was disbelief. The attack used mathematics from the 1990s—nothing quantum, nothing fancy. It had been sitting there, waiting to be discovered, the entire time. The cryptographic community was stunned.*

Example 37 (Timeline of Doom)**.**

- *July 30, 2022: Attack posted on ePrint*
- *August 1: Attack verified independently*
- *August 5: All SIKE parameters broken*
- *August 8: Attack extended to SIDH*
- *August 15: Microsoft removes SIKE from libraries*
- *Result: 10 years of work destroyed in 2 weeks*

Other Failed Attempts The graveyard extends beyond these:

1. Knapsack Cryptography (1978-1982)

- Based on subset sum (NP-complete!)
- Broken by Shamir using lattice reduction
- Lesson: Worst-case hardness $\neq$ average-case hardness

2. Braid Group Cryptography (1999-2007)

- Based on word problem in braid groups
- Elegant non-commutative structure
- Broken by various heuristic attacks
- Lesson: Infinite groups don't guarantee security

3. Polynomial HFE variants (ongoing)

- Hidden Field Equations and variants
- Repeatedly patched and re-broken
- Current status: Mostly abandoned

What Makes a Survivor? Patterns emerge from the carnage:

Common failure modes:

1. **Hidden structure too visible:** Cryptanalysis finds the trapdoor
2. **Parameter trap:** Secure parameters make scheme impractical
3. **Mathematical brittleness:** One insight breaks everything
4. **Limited cryptanalysis:** Not enough eyes on the problem
5. **Narrow security margin:** Small parameter increases don't help

Properties of survivors:

1. **Simple underlying problem:** Lattices, codes, hashes
2. **Multiple independent constructions:** Different ways to use same hardness
3. **Graceful degradation:** Attacks reduce security, don't destroy it
4. **20+ years of study:** Time tests all things
5. **Connection to other areas:** More researchers, more confidence

Approach	Years Studied	Status	Why
Lattices	25+	Selected	Robust theory
Codes	45+	Finalist	Key size only
Hash	40+	Selected	Conservative
Multivariate	35+	Broken	Structure leaks
Isogeny	10	Broken	Auxiliary attack
Knapsack	4	Broken	Lattice reduction

Table 17: Time is the ultimate cryptanalytic test

The Innovation Paradox Post-quantum cryptography faces a fundamental tension:

- **Need innovation:** Must replace all current public-key crypto
- **Need conservatism:** Can't afford failures in deployment
- **Result:** Stick with well-studied approaches

Remark 26 (The 20-Year Rule)**.** *An informal principle has emerged: don't trust any cryptographic approach with less than 20 years of serious cryptanalysis. This rules out exciting new mathematics but prevents SIKE-style catastrophes. In cryptography, boring is beautiful.*

Lessons from the graveyard:

1. **Mathematical beauty $\neq$ security:** SIKE was gorgeous mathematics

2. **NP-completeness is not enough:** Average case matters
3. **Structure for efficiency is dangerous:** It's also structure for attacks
4. **Novel mathematics needs decades:** No shortcuts to confidence
5. **Multiple approaches essential:** Don't put all eggs in one basket

The spectacular failure of SIKE just as NIST was finalizing standards reinforced the conservative choices. Lattices, codes, and hashes may lack the mathematical excitement of isogenies, but they've survived decades of attack. In cryptography, survival is the only metric that matters.

Example 38 (The SIKE Postmortem)**.** *What went wrong?*

- *Too much focus on quantum attacks*
- *Not enough on classical cryptanalysis*
- *Auxiliary information considered "harmless"*
- *Small community = fewer eyes*
- *Lesson: Quantum-safe must mean classically-safe first!*

This tour through the cryptographic graveyard explains why NIST chose conservative, well-studied approaches. The next time someone proposes a brilliant new mathematical foundation for post-quantum cryptography, remember SIKE: mathematical elegance is no defense against a clever graduate student with a laptop.

4 Deep Dive: Lattice Cryptography

4.1 Mathematical Foundations

4.1.1 What is a Lattice?

Let's start with what you already know. In crystallography, a lattice describes the periodic arrangement of atoms in a crystal—a discrete set of points with translational symmetry. Cryptographic lattices are exactly the same mathematical object, just in hundreds of dimensions where your geometric intuition breaks down and computational problems become exponentially hard.

Definition 20 (Lattice)**.** *A lattice $\mathcal{L} \subseteq \mathbb{R}^n$ is the set of all integer linear combinations of linearly independent basis vectors $\mathbf{b}_1, ..., \mathbf{b}_n \in \mathbb{R}^n$:*

$$\mathcal{L}(\mathbf{B}) = \left\{ \sum_{i=1}^{n} a_i \mathbf{b}_i : a_i \in \mathbb{Z} \right\} = \{\mathbf{Bx} : \mathbf{x} \in \mathbb{Z}^n\}$$

where $\mathbf{B} = [\mathbf{b}_1|...|\mathbf{b}_n]$ is the basis matrix.

The Physics Connection You Already Have Your crystallography background provides perfect intuition:

Example 39 (From NaCl to Cryptography)**.** *Consider the face-centered cubic lattice of NaCl:*

- *Basis vectors:* $\mathbf{a}_1 = (a, 0, 0)$, $\mathbf{a}_2 = (0, a, 0)$, $\mathbf{a}_3 = (0, 0, a)$
- *You can instantly find the nearest atom to any point*
- *Now imagine this in 700 dimensions—suddenly finding nearest points is exponentially hard*
- *This dimension-induced hardness is the foundation of lattice cryptography*

Crystallography	Cryptographic Lattices	Key Difference
3D lattice	n-dimensional lattice	$n \approx 500 - 1000$
Primitive unit cell	Good basis	Many possible bases
Reciprocal lattice	Dual lattice	Same hardness properties
Brillouin zone	Fundamental domain	Voronoi cell
Miller indices	Coefficient vector	Integer constraints
Bragg peaks	Lattice points	Discrete structure

Table 18: Your physics intuition translates directly—until dimension explodes

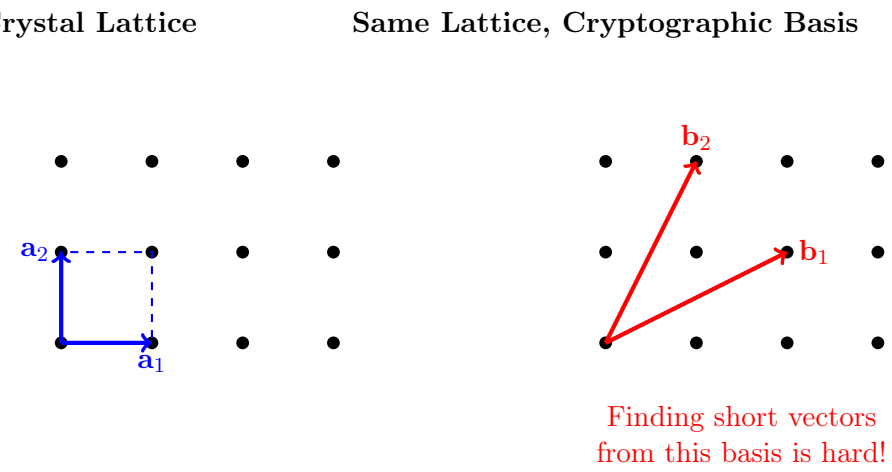


Figure 32: Same lattice points, different bases: The cryptographic challenge

Good Basis vs Bad Basis: The Cryptographic Gap The same lattice can have exponentially different bases—and this difference creates cryptographic security.

Example 40 (Concrete Basis Comparison)**.** *Consider a 2D lattice with two different bases:*

Good basis: $\mathbf{B}_{good} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$

- *Shortest vector:* $(1, 0)$ *with length 1*
- *Orthogonality defect: 1 (perfect)*
- *Finding short vectors: Trivial*

Bad basis: $\mathbf{B}_{bad} = \begin{pmatrix} 97 & 89 \\ 89 & 81 \end{pmatrix}$

- *Same lattice! (determinant = 1)*
- *Basis vectors have length > 100*
- *Shortest vector still* $(1, 0)$ *but now* $= 97\mathbf{b}_1 - 89\mathbf{b}_2$
- *Finding this combination: Requires sophisticated algorithms*

Definition 21 (Basis Quality Measures)**.** *For a basis* $\mathbf{B} = [\mathbf{b}_1|...|\mathbf{b}_n]$*:*

- ***Orthogonality defect:*** $\delta(\mathbf{B}) = \frac{\prod_{i=1}^{n} \|\mathbf{b}_i\|}{\det(\mathcal{L})}$ *(ideal = 1)*
- ***Hermite factor:*** $\gamma = \left(\frac{\|\mathbf{b}_1\|}{\det(\mathcal{L})^{1/n}}\right)^{1/n}$ *(smaller = better)*
- ***Condition number:*** $\kappa(\mathbf{B}) = \|\mathbf{B}\| \cdot \|\mathbf{B}^{-1}\|$ *(near 1 = good)*

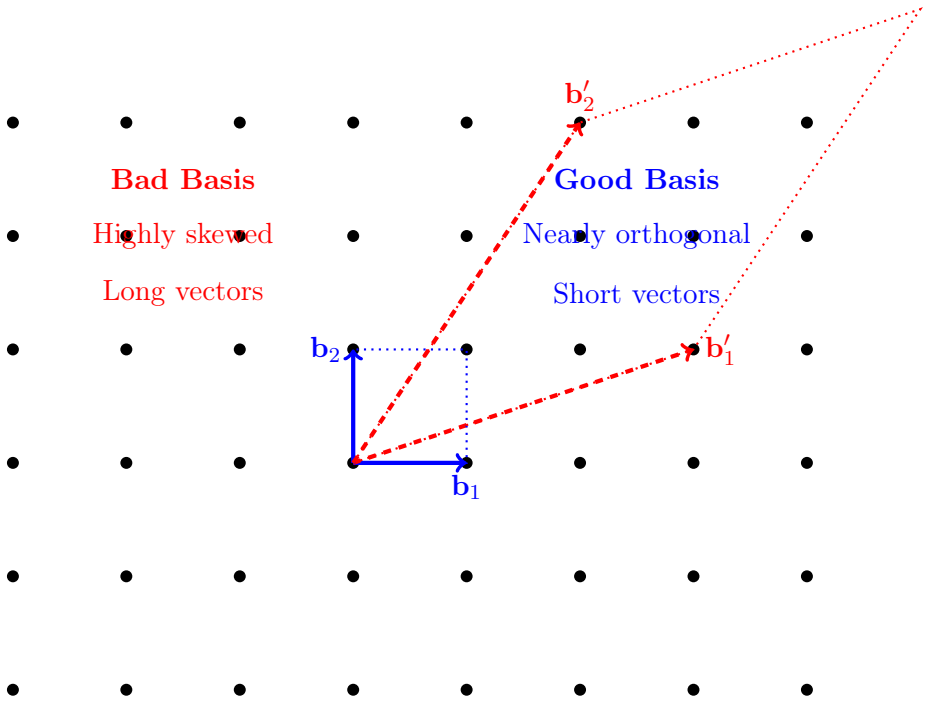


Figure 33: Same lattice, vastly different bases: Finding the good basis from the bad is cryptographically hard

The High-Dimensional Reality Check Here's where your 3D intuition fails catastrophically:

Theorem 7 (Dimension Changes Everything)**.** *In dimension n:*

1. *Volume of unit ball:* $V_n = \frac{\pi^{n/2}}{\Gamma(n/2+1)} \to 0$ *as* $n \to \infty$
2. *Most volume is near the surface:* $\frac{V_n(r)}{V_n(R)} = \left(\frac{r}{R}\right)^n$
3. *Random vectors are nearly orthogonal:* $\mathbb{E}[|\langle \mathbf{x}, \mathbf{y} \rangle|] = O(1/\sqrt{n})$
4. *Number of "almost shortest" vectors:* $\exp(\Theta(n))$

Remark 27 (Why Physicists Should Care)**.** *In condensed matter, you solve lattice problems in 3D constantly—finding ground states, computing band structures, analyzing defects. In 3D, these are polynomial-time problems. In 500D, they become exponentially hard. This dimension-induced phase transition from easy to hard is what protects cryptographic secrets.*

Example 41 (Concrete Dimension Impact)**.** *Finding shortest vector in random lattice:*

Dimension	***Time (best known)***	***Feasible?***
10	*Seconds*	*Trivial*
50	*Hours*	*Yes*
100	*Years*	*Borderline*
300	10^{30} *years*	*No*
700	10^{100} *years*	*Never*

Cryptography uses $n \geq 500$ for post-quantum security.

4.1.2 Fundamental Lattice Problems

Two problems form the foundation of lattice cryptography. Unlike factoring or discrete log, these problems have resisted efficient algorithms for decades—even with quantum computers.

The Shortest Vector Problem (SVP)

Definition 22 (Shortest Vector Problem). *Given a basis* $\mathbf{B}$ *of lattice* $\mathcal{L}$*, find a shortest non-zero vector:*

$$\mathbf{v} \in \mathcal{L} \setminus \{\mathbf{0}\} \text{ such that } \|\mathbf{v}\| = \lambda_1(\mathcal{L}) = \min_{\mathbf{x} \in \mathcal{L} \setminus \{\mathbf{0}\}} \|\mathbf{x}\|$$

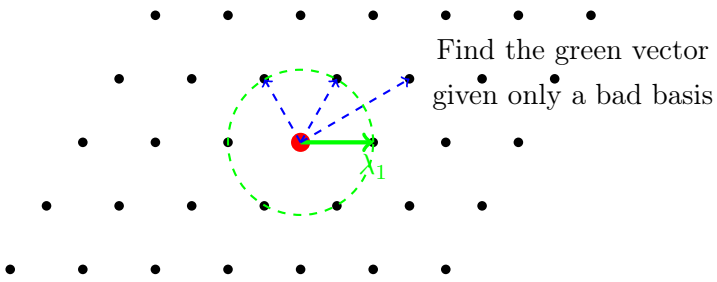


Figure 34: SVP: Find the shortest non-zero lattice vector

Why SVP is hard:

- **Exponentially many candidates:** Ball of radius R contains $\approx R^n$ lattice points
- **No local structure:** Can't use gradient descent or similar
- **Discrete geometry:** Continuous relaxations don't help
- **Basis dependence:** Good basis makes it trivial, bad basis makes it hard

The Closest Vector Problem (CVP)

Definition 23 (Closest Vector Problem). *Given a basis* $\mathbf{B}$ *of lattice* $\mathcal{L}$ *and target* $\mathbf{t} \in \mathbb{R}^n$*, find:*

$$\mathbf{v} \in \mathcal{L} \text{ such that } \|\mathbf{v} - \mathbf{t}\| = \min_{\mathbf{x} \in \mathcal{L}} \|\mathbf{x} - \mathbf{t}\|$$

CVP is the decoding problem: given a noisy codeword, find the original.

Example 42 (CVP in Cryptography). *In LWE decryption:*

- *Target:* $\mathbf{t} = \mathbf{s} + \mathbf{e}$ *(secret + small error)*
- *Lattice: Generated by public key*
- *If you can solve CVP: Recover* $\mathbf{s}$*, break encryption*
- *Reality: CVP is NP-hard even to approximate*

The Deep Connection: SVP and CVP These problems are intimately related through duality:

Theorem 8 (SVP-CVP Connection).

1. *CVP in lattice* $\mathcal{L} \leftrightarrow$ *SVP in modified lattice* $\mathcal{L}'$
2. *Solving* γ*-CVP gives algorithm for* (2γ)*-SVP*
3. *Both problems have same quantum complexity (up to polynomial factors)*

This connection means that progress on one problem immediately impacts the other—a crucial consideration for cryptographic security.

Approximation Factor	Complexity	Best Algorithm	Quantum?
$\gamma = 1$ (exact)	NP-hard	Exponential	No speedup
$\gamma = O(1)$	NP-hard	Exponential	No speedup
$\gamma = 2^{O(\sqrt{\log n})}$	Unknown	Sub-exponential	No speedup
$\gamma = \text{poly}(n)$	P (not useful)	Polynomial	Already easy

Table 19: Hardness depends critically on approximation factor

Approximation Variants Perfect solutions are often unnecessary. Approximation versions are more relevant:

Definition 24 (Approximation SVP/CVP)**.** *γ-SVP: Find $\mathbf{v} \in \mathcal{L}$ with $0 < \|\mathbf{v}\| \leq \gamma \cdot \lambda_1(\mathcal{L})$*
γ-CVP: Find $\mathbf{v} \in \mathcal{L}$ with $\|\mathbf{v} - \mathbf{t}\| \leq \gamma \cdot dist(\mathbf{t}, \mathcal{L})$

The Unique SVP Gap Unlike many problems, lattice problems have a "hardness gap":

Theorem 9 (Hardness Gap for SVP)**.** *For a random lattice $\mathcal{L}$ of dimension n:*

- $\lambda_1(\mathcal{L}) \approx \sqrt{n} \cdot \det(\mathcal{L})^{1/n}$ *(shortest vector)*
- $\lambda_2(\mathcal{L}) \approx \lambda_1(\mathcal{L})$ *(second shortest)*
- *But in "planted" lattices: $\lambda_1 \ll \lambda_2$*

This gap enables cryptographic constructions.

4.1.3 Computational Complexity

Here's the critical fact that makes lattice cryptography quantum-safe: the best known quantum algorithms for lattice problems offer only polynomial speedups, not exponential ones.

Algorithm 11 LLL Algorithm (Simplified)

Require: Basis $\mathbf{B} = [\mathbf{b}_1, ..., \mathbf{b}_n]$
Ensure: Reduced basis with $\|\mathbf{b}_1\| \leq 2^{(n-1)/2}\lambda_1(\mathcal{L})$

```
B* ← GramSchmidt(B)                          ▷ Orthogonalization
repeat
    for i = 2 to n do
        for j = i − 1 down to 1 do
            b_i ← b_i − ⌊μ_{i,j}⌉ b_j        ▷ Size reduction
    for i = 1 to n − 1 do
        if Lovász condition violated then
            Swap b_i and b_{i+1}
until no more swaps
return B
```

Classical Algorithms: The State of the Art **Algorithm landscape:**

- **LLL (1982):** Polynomial time, exponential approximation
- **BKZ (Block Korkine-Zolotarev):** Time-quality tradeoff
- **Sieving algorithms:** Best for exact SVP
- **Enumeration:** Good for small dimensions

Algorithm	Time	Approximation	Memory
LLL	$O(n^5 \log B)$	$2^{O(n)}$	$O(n^2)$
BKZ-β	$2^{O(\beta)}$	γ^n, $\gamma = O(\beta/n)$	$O(n^2)$
Sieving	$2^{0.292n+o(n)}$	Exact	$2^{0.208n}$
Enumeration	$2^{O(n \log n)}$	Exact	poly(n)

Table 20: Classical algorithms: Trade time for approximation quality

Quantum (Non-)Advantages The quantum story for lattices is remarkably different from factoring:

Theorem 10 (Quantum Speedup for Lattice Problems)**.** *Best known quantum algorithms for n-dimensional lattice problems:*

- *SVP:* $2^{0.265n+o(n)}$ *(vs classical* $2^{0.292n+o(n)}$*)*
- *CVP: Similar modest improvement*
- *Speedup:* $2^{0.027n} \approx 1.019^n$ *(not exponential!)*

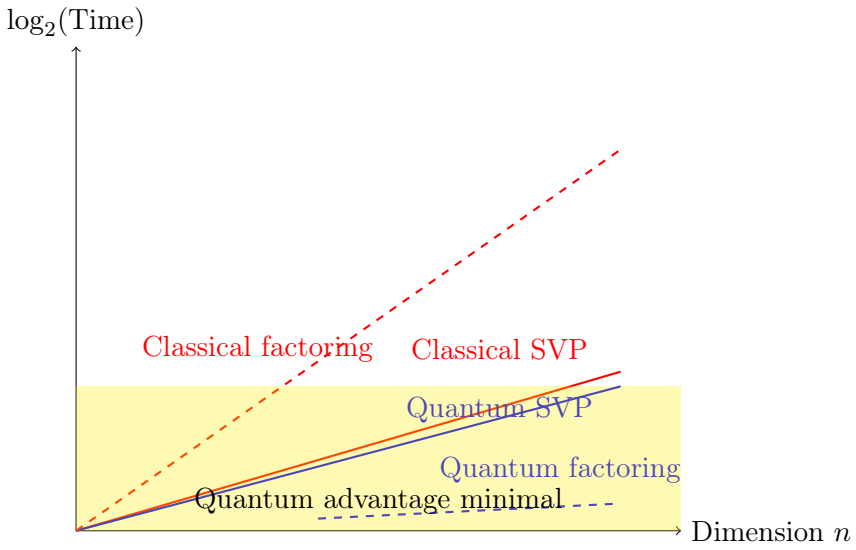


Figure 35: Quantum speedup for lattices is polynomial, not exponential like factoring

Why no quantum breakthrough?

1. **No periodic structure:** Shor's algorithm needs periodicity
2. **No hidden subgroup:** Lattice problems don't fit HSP framework
3. **Grover gives little:** Search space too large
4. **Geometric nature:** Quantum computers don't naturally handle geometry

Remark 28 (The Quantum Surprise)**.** *Most physicists expect quantum computers to revolutionize optimization. For lattice problems, they barely help. This isn't because we haven't found the right algorithm—there are information-theoretic arguments suggesting no exponential speedup exists. Lattice geometry seems fundamentally resistant to quantum attack.*

Concrete Security Estimates Translating complexity to real-world security:

Definition 25 (Core-SVP Hardness)**.** *The standard security metric: time to solve SVP in dimension n on a classical computer.*

- *Model: $T = 2^{0.292n}$ operations*
- *Operation = one lattice vector operation*
- *Assume 10^9 operations/second/core*
- *Parallelization: Limited by memory requirements*

Dimension	Classical Security	Quantum Security	NIST Level
400	2^{117}	2^{106}	Below I
500	2^{146}	2^{133}	I (AES-128)
600	2^{175}	2^{159}	III (AES-192)
700	2^{204}	2^{186}	V (AES-256)

Table 21: Lattice dimensions for standard security levels

Example 43 (Breaking Kyber-768)**.** *Kyber-768 uses dimension 768 lattices:*

- *Classical time: 2^{224} operations*
- *With Earth's computing power: 10^{50} years*
- *Quantum time: 2^{203} operations*
- *With mythical quantum computer: 10^{44} years*
- *Compare RSA-2048: 8 hours on same quantum computer*

The Bottom Line Lattice problems occupy a unique position in the computational complexity landscape:

1. **Worst-case hardness:** Breaking average instances as hard as worst instances
2. **NP-hard core:** Exact versions are NP-hard
3. **Quantum resistance:** No exponential quantum speedup known
4. **25+ years of cryptanalysis:** No breakthrough despite intense effort
5. **Simple to state:** Find short vectors—how hard could it be?

This combination of properties—especially quantum resistance—makes lattices the foundation for post-quantum cryptography. But having hard problems isn't enough; we need to build cryptosystems. That's where Learning With Errors enters the picture.

Remark 29 (For the Skeptical Physicist)**.** *You might think: "Surely quantum computers must help with geometric optimization." The surprise is they don't—at least not exponentially. This isn't like factoring where quantum computers find hidden structure. Lattice problems appear to be "genuinely hard" in a way that resists both classical and quantum optimization. After 25 years of trying, the quantum speedup remains stubbornly polynomial.*

4.2 Learning With Errors (LWE)

If lattices are the foundation of post-quantum cryptography, then Learning With Errors is the master key that unlocks their potential. LWE transforms abstract geometric problems into practical cryptographic tools. The brilliant insight: add carefully controlled noise to linear algebra, and suddenly you have encryption, signatures, and even fully homomorphic encryption. For physicists, think of it as adding thermal noise to a crystal lattice—except here the noise is our friend, not our enemy.

4.2.1 The LWE Problem

Start with something every physicist knows: solving linear equations. Now add one twist that changes everything.

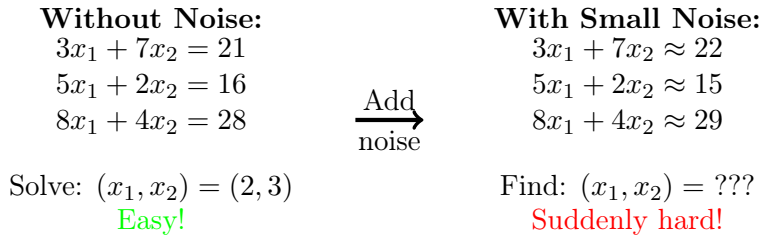


Figure 36: LWE in one picture: Noise transforms easy linear algebra into hard problems

The Beautiful Idea: Linear Algebra Plus Noise

Definition 26 (Learning With Errors (Search Version))**.** *Let $n, q \in \mathbb{N}$, and χ be an error distribution over $\mathbb{Z}_q$.*

- ***Secret:*** $\mathbf{s} \in \mathbb{Z}_q^n$ *(chosen uniformly)*
- ***Samples:*** *Many pairs $(\mathbf{a}_i, b_i)$ where:*
  - $\mathbf{a}_i \in \mathbb{Z}_q^n$ *uniform random*
  - $e_i \leftarrow \chi$ *(small error)*
  - $b_i = \langle \mathbf{a}_i, \mathbf{s} \rangle + e_i \bmod q$
- ***Goal:*** *Recover* $\mathbf{s}$

Example 44 (Concrete LWE Instance)**.** *Parameters: $n = 4$, $q = 97$, errors in $\{-1, 0, 1\}$*
Secret: $\mathbf{s} = (13, 25, 17, 41)$
Sample equations:

$$23 \cdot 13 + 67 \cdot 25 + 43 \cdot 17 + 91 \cdot 41 \equiv 51 \pmod{97} \quad \textit{(error = 0)} \tag{33}$$

$$71 \cdot 13 + 19 \cdot 25 + 84 \cdot 17 + 52 \cdot 41 \equiv 9 \pmod{97} \quad \textit{(error = 1)} \tag{34}$$

$$45 \cdot 13 + 88 \cdot 25 + 31 \cdot 17 + 76 \cdot 41 \equiv 74 \pmod{97} \quad \textit{(error = -1)} \tag{35}$$

Without errors: Gaussian elimination in milliseconds. With errors: Best known algorithm takes exponential time!

The Critical Parameters LWE security depends on careful parameter balance:

Definition 27 (LWE Parameters)**.**

- n*: Dimension (security parameter), typically 500-1000*
- q*: Modulus (polynomial in* n*), typically* n^2 *to* n^3
- χ*: Error distribution, typically discrete Gaussian*
- $\alpha = \sigma/q$*: Noise rate, typically* $1/\sqrt{n}$ *to* $1/poly(n)$

Parameter	Typical Value	Effect
Larger n	$512 \to 1024$	More security, larger keys
Larger q	$2^{13} \to 2^{15}$	Easier implementation, weaker security
Larger σ	$3.2 \to 6.4$	More security, higher failure rate
More samples m	$n \log q$	More information for attacker

Table 22: LWE parameter tradeoffs: Balance security, efficiency, and correctness

The Error Distribution: Your Noise Model For physicists, the error distribution is like choosing your thermal noise model:

Definition 28 (Discrete Gaussian Distribution)**.** *The discrete Gaussian* $D_{\mathbb{Z},\sigma}$ *over integers with parameter* σ*:*

$$\Pr[X = x] \propto \exp\left(-\frac{x^2}{2\sigma^2}\right)$$

Truncated to $[-B, B]$ *where* $B = \Theta(\sigma\sqrt{\log n})$ *for negligible tail probability.*

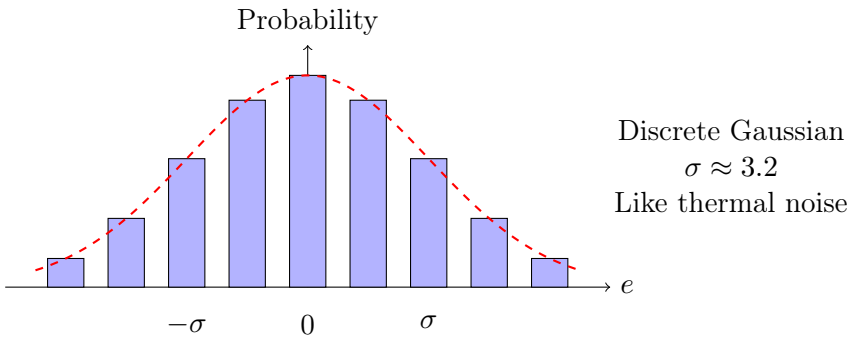


Figure 37: LWE errors: Discrete Gaussian noise hides the secret

Search vs Decision: Two Faces of LWE

Definition 29 (Decision-LWE)**.** *Distinguish between:*

- *LWE samples:* $(\mathbf{a}_i, \langle \mathbf{a}_i, \mathbf{s} \rangle + e_i)$
- *Uniform random:* $(\mathbf{a}_i, u_i)$ *where* $u_i \in \mathbb{Z}_q$ *uniform*

Theorem 11 (Search-Decision Equivalence)**.** *For polynomial* q *and Gaussian errors, Search-LWE and Decision-LWE are polynomial-time equivalent.*

This equivalence is crucial: cryptographic security often relies on indistinguishability.

4.2.2 Why LWE is Hard

LWE's security rests on one of the most beautiful results in cryptography: average-case LWE is as hard as worst-case lattice problems. This is extraordinary—most cryptographic assumptions are average-case with no worst-case guarantees.

The Worst-Case to Average-Case Connection

Theorem 12 (Regev's Reduction (Simplified))**.** *If you can solve LWE with parameters (n, q, χ) in polynomial time, then you can solve γ-SVP on any n-dimensional lattice in polynomial time, where:*

$$\gamma = \tilde{O}(n\sqrt{q}/\sigma)$$

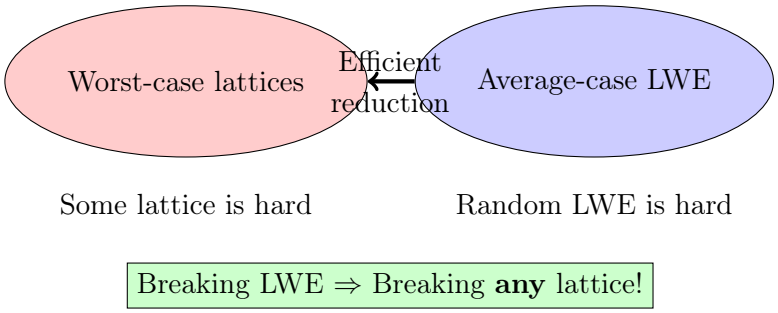


Figure 38: LWE's extraordinary guarantee: Average equals worst-case hardness

Why this matters:

- **RSA/DH:** No worst-case guarantees (some numbers might be easy to factor)
- **LWE:** If any lattice problem is hard, then random LWE is hard
- **Confidence:** Don't need to avoid "weak instances"

The Reduction Intuition: Quantum Fourier Transform Connection Here's the brilliant idea behind Regev's reduction, explained for physicists:

1. **Start with any hard lattice $\mathcal{L}$**
2. **Use LWE oracle** to learn about $\mathcal{L}$'s dual lattice $\mathcal{L}^*$
3. **Apply Quantum Fourier Transform** to connect primal and dual
4. **Reconstruct short vectors** in $\mathcal{L}$ from dual information
5. **Works for ANY lattice**, not just special ones

Remark 30 (The Physics of the Quantum Fourier Transform in Regev's Reduction)**.** *For physicists, this is deeply familiar: the quantum Fourier transform (QFT) connects position and momentum representations. In crystallography:*

- *Real space lattice ↔ Reciprocal (momentum) space lattice*
- *Fourier transform relates the two*
- *Small features in real space ↔ Large features in reciprocal space*

Regev's reduction exploits exactly this duality:

- *LWE samples give noisy information about the dual lattice $\mathcal{L}^*$*
- *QFT converts dual lattice info back to primal lattice*
- *The noise level determines the approximation factor γ*
- *This is why the original reduction required quantum computation*

Classical reductions now exist but are more complex and less intuitive.

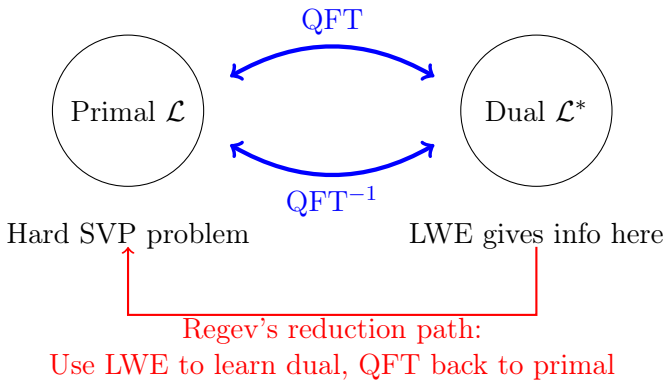


Figure 39: The quantum Fourier transform bridges primal and dual lattices in Regev's reduction

Quantum Resistance: Why Shor Doesn't Apply LWE resists quantum attack for fundamental reasons:

Theorem 13 (No Quantum Advantage for LWE)**.** *Best known quantum algorithms for LWE:*

- *Algebraic attacks: No better than classical*
- *Lattice reduction: Polynomial speedup only*
- *Grover search: $\sqrt{q^n}$ time (infeasible for large n)*
- *BKZ variants: $2^{0.265n}$ quantum vs $2^{0.292n}$ classical*

No exponential quantum speedup known after 15+ years.

Why Shor's algorithm fails:

- **No periodicity:** LWE samples don't have hidden periods
- **Continuous noise:** Errors destroy any algebraic structure
- **High dimension:** Quantum amplitude spread too thin
- **No group structure:** Can't use hidden subgroup framework

Example 45 (Concrete Quantum Attack on LWE)**.** *Kyber-768 parameters: $n = 256$, $q = 3329$, module rank $k = 3$*

- *Effective lattice dimension:* 768
- *Classical security:* $\approx 2^{180}$
- *Quantum security:* $\approx 2^{164}$
- *Grover on key:* 2^{128} *(dominates)*
- *Conclusion: 128-bit quantum security achieved*

Attack Type	Complexity	Kyber-512	Kyber-768	Kyber-1024
Primal lattice	$2^{0.292d}$	2^{118}	2^{180}	2^{257}
Dual lattice	$2^{0.292d}$	2^{119}	2^{182}	2^{260}
Algebraic	Unknown	—	—	—
Grover key search	$2^{n/2}$	2^{128}	2^{128}	2^{128}
Overall (quantum)		2^{107}	2^{164}	2^{236}

Table 23: LWE security: Multiple attacks, none are efficient

Concrete Hardness: From Theory to Practice

4.2.3 LWE-Based Encryption

Now for the payoff: how to build encryption from LWE. Regev's 2005 scheme remains the template for all modern constructions.

The Geometric Intuition Think of LWE encryption as hiding a message in a high-dimensional noise cloud:

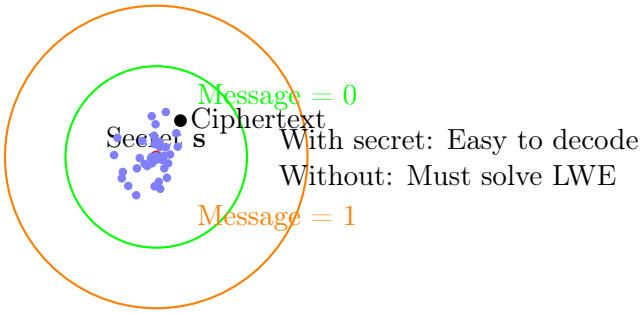


Figure 40: LWE encryption geometry: Hide messages in noise clouds around secret points

Algorithm 12 Regev's LWE Encryption Scheme

Setup: Parameters n, q, χ, message space $\{0,1\}$
KeyGen():
  $\mathbf{s} \leftarrow \mathbb{Z}_q^n$ ▷ Secret key
  For $i = 1$ to $m = n \log q$:
    $\mathbf{a}_i \leftarrow \mathbb{Z}_q^n$, $e_i \leftarrow \chi$
    $b_i = \langle \mathbf{a}_i, \mathbf{s} \rangle + e_i \bmod q$
  $pk = \{(\mathbf{a}_i, b_i)\}_{i=1}^m$ ▷ Public key
  Return $(pk, sk = \mathbf{s})$
Encrypt$(pk, \mu \in \{0,1\})$:
  $S \subseteq [m]$ random subset
  $\mathbf{a}' = \sum_{i \in S} \mathbf{a}_i \bmod q$
  $b' = \sum_{i \in S} b_i + \mu \cdot \lfloor q/2 \rfloor \bmod q$
  Return $(\mathbf{a}', b')$
Decrypt$(sk = \mathbf{s}, (\mathbf{a}', b'))$:
  $v = b' - \langle \mathbf{a}', \mathbf{s} \rangle \bmod q$
  If $|v| < q/4$: return 0
  Else: return 1

Correctness Analysis Why does decryption work? Let's trace through the algebra:

Theorem 14 (Decryption Correctness)**.** *For ciphertext* $(\mathbf{a}', b')$ *encrypting* μ*:*

$$v = b' - \langle \mathbf{a}', \mathbf{s} \rangle \tag{36}$$

$$= \sum_{i \in S} b_i + \mu \cdot \lfloor q/2 \rfloor - \langle \sum_{i \in S} \mathbf{a}_i, \mathbf{s} \rangle \tag{37}$$

$$= \sum_{i \in S} (\langle \mathbf{a}_i, \mathbf{s} \rangle + e_i) + \mu \cdot \lfloor q/2 \rfloor - \sum_{i \in S} \langle \mathbf{a}_i, \mathbf{s} \rangle \tag{38}$$

$$= \sum_{i \in S} e_i + \mu \cdot \lfloor q/2 \rfloor \tag{39}$$

$$= e_{total} + \mu \cdot \lfloor q/2 \rfloor \tag{40}$$

If $|e_{total}| < q/4$*, decryption recovers* μ *correctly.*

Example 46 (Concrete Encryption)**.** *Parameters:* $n = 4$*,* $q = 97$*, message* $\mu = 1$
Public key sample: $(\mathbf{a}_1, b_1) = ((23, 67, 43, 91), 51)$
Encryption with subset $S = \{1, 3, 5\}$*:*

- $\mathbf{a}' = \mathbf{a}_1 + \mathbf{a}_3 + \mathbf{a}_5 = (45, 23, 78, 12)$
- $b' = b_1 + b_3 + b_5 + 48 = 51 + 29 + 83 + 48 = 17 \bmod 97$
- *Ciphertext:* $((45, 23, 78, 12), 17)$

Decryption:

- $v = 17 - \langle (45, 23, 78, 12), (13, 25, 17, 41) \rangle = 17 - 66 = 48 \bmod 97$
- *Since* $48 \approx q/2$*, decrypt as* $\mu = 1$ ✓

Security Analysis The security proof is remarkably clean:

Theorem 15 (LWE Encryption Security)**.** *If Decision-LWE is hard, then the encryption scheme is IND-CPA secure.*

Proof sketch:

1. Public key: LWE samples $\{(\mathbf{a}_i, b_i)\}$
2. By Decision-LWE: Indistinguishable from uniform
3. Encryption uses random subset sum
4. Result looks uniform to attacker without $\mathbf{s}$
5. Message bit hidden by $\lfloor q/2 \rfloor$ offset

From Basic Scheme to Modern Cryptography Regev's scheme demonstrates feasibility but isn't practical:

- Public keys: $O(n^2 \log q)$ bits
- Ciphertexts: $O(n \log q)$ bits
- Only encrypts single bits

Modern improvements:

- **Ring-LWE:** Use polynomial structure for compression
- **Module-LWE:** Balance between security and efficiency
- **Better encoding:** Encrypt multiple bits per ciphertext
- **CCA security:** Add authentication via Fujisaki-Okamoto

This evolution leads directly to Kyber/ML-KEM, the NIST standard.

Remark 31 (The Path Forward)**.** *We've shown that:*

- *Lattice problems resist quantum attack*
- *LWE reduces average-case to worst-case hardness*
- *The quantum Fourier transform elegantly connects primal and dual in the reduction*
- *Practical encryption emerges from noisy linear algebra*

The remaining challenge: efficiency. Ring-LWE and Module-LWE solve this by adding algebraic structure—our next topic.

Example 47 (LWE Parameters in Practice)**.**

Scheme	n	q	σ	***Security***
Toy example	*64*	*1,024*	*2.0*	*None*
Learning	*256*	*4,093*	*2.8*	2^{80}
Kyber-512	*256×2*	*3,329*	*1.6*	2^{128} *quantum*
Kyber-768	*256×3*	*3,329*	*1.6*	2^{192} *quantum*
Kyber-1024	*256×4*	*3,329*	*1.6*	2^{256} *quantum*

The module dimensions (256×k) show how Ring-LWE improves efficiency.

The beauty of LWE: it transforms the geometric hardness of lattices into the algebraic simplicity of linear equations with noise. This bridge from theory to practice, with the quantum Fourier transform providing the theoretical guarantee, makes lattice-based cryptography not just secure, but implementable at Internet scale.

4.3 Ring-LWE and Module-LWE

Plain LWE gives us security but at a devastating cost: public keys of megabytes and ciphertexts of kilobytes. For Internet-scale deployment, we need a 100× improvement in efficiency without sacrificing security. The solution comes from an unexpected source: polynomial rings. By adding algebraic structure to our lattices, we achieve dramatic compression while maintaining quantum resistance. This isn't just an optimization—it's the breakthrough that makes post-quantum cryptography practical.

4.3.1 Efficiency Through Structure

Let's confront the brutal reality of plain LWE efficiency:

The Key Size Crisis

Example 48 (Impact on TLS Handshake)**.** *With plain LWE for 128-bit quantum security:*

- *Each handshake: 1.6 MB public key transmission*
- *Google scale (100B/day): 160 petabytes/day extra*

Scheme	Public Key	Ciphertext	Usable?
RSA-2048	256 B	256 B	Yes
ECDH-P256	64 B	64 B	Yes
Plain LWE-512	1.6 MB	2 KB	No
Plain LWE-768	3.6 MB	3 KB	No
Plain LWE-1024	6.4 MB	4 KB	No
Ring-LWE-512	800 B	768 B	Yes
Ring-LWE-768	1,184 B	1,088 B	Yes
Ring-LWE-1024	1,568 B	1,568 B	Yes

Table 24: The efficiency revolution: Ring structure enables 1000× key compression

- *Mobile users: Minutes to establish connection*
- *Certificate chains: 5+ MB with intermediate CAs*
- *Verdict: Completely impractical*

The Polynomial Ring Solution The breakthrough: represent vectors as polynomials and exploit algebraic structure.

Definition 30 (The Ring R_q)**.** *Let n be a power of 2. Define:*

$$R = \mathbb{Z}[X]/(X^n + 1) \quad \textit{and} \quad R_q = \mathbb{Z}_q[X]/(X^n + 1)$$

Elements are polynomials of degree $< n$ with coefficients in $\mathbb{Z}_q$.

Why polynomials help:

- **Vector:** $(a_0, a_1, ..., a_{n-1}) \in \mathbb{Z}_q^n$ requires n elements
- **Polynomial:** $a(X) = a_0 + a_1X + ... + a_{n-1}X^{n-1}$ requires n coefficients
- **Magic:** Multiplication by $a(X)$ represents $n \times n$ circulant matrix!

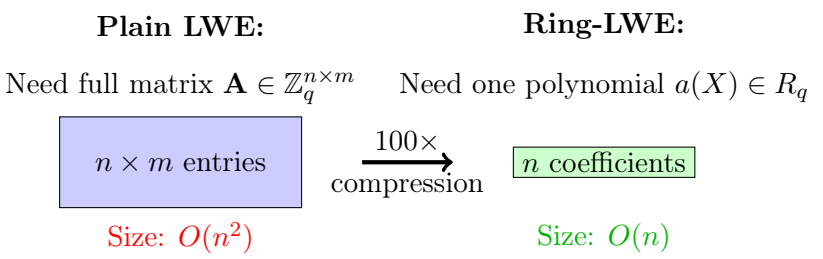


Figure 41: Ring structure enables massive compression without losing information

Why $X^n + 1$? The Cyclotomic Choice The choice of $X^n + 1$ (where $n = 2^k$) is carefully motivated by both mathematical and practical considerations:

Theorem 16 (Why Cyclotomic Polynomials Are Optimal)**.** *The polynomial $X^n + 1$ for $n = 2^k$ provides:*

1. ***Irreducibility:*** *$X^n + 1$ is the $2n$-th cyclotomic polynomial, irreducible over $\mathbb{Z}$*
2. ***Fast arithmetic:*** *Enables Number Theoretic Transform (NTT) in $O(n \log n)$*

3. ***Security:*** *Corresponds to ideal lattices with provable worst-case hardness*

4. ***Implementation:*** *Powers of 2 optimize memory access and bit operations*

Remark 32 (Why Not Other Polynomials?)**.** *You might wonder about alternatives:*

- $X^n - 1$*: Factors as* $(X-1)(X^{n-1}+...+1)$*, not irreducible*
- $X^p - 1$ *(p prime): Gives cyclic convolution, less secure structure*
- *Random irreducible: No fast multiplication algorithm*
- *Other cyclotomics: Less efficient NTT, harder error sampling*

The choice $X^{2^k}+1$ *uniquely balances all requirements.*

Example 49 (Multiplication in R_q)**.** *Let* $n=4$*,* $q=17$*. In* $R_q = \mathbb{Z}_{17}[X]/(X^4+1)$*:*
$a(X) = 3+2X+5X^2+X^3$ $b(X) = 1+4X+2X^2+3X^3$
Computing $a(X)\cdot b(X)$*:*

$$\begin{aligned} a \cdot b &= (3+2X+5X^2+X^3)(1+4X+2X^2+3X^3) && (41)\\ &= 3+14X+19X^2+34X^3+21X^4+17X^5+3X^6 && (42)\\ &\text{Reduce using } X^4 \equiv -1: && (43)\\ &= 3+14X+19X^2+34X^3-21-17X-3X^2 && (44)\\ &= -18+-3X+16X^2+34X^3 && (45)\\ &\equiv 16+14X+16X^2+0X^3 \pmod{17} && (46) \end{aligned}$$

The Negacyclic Magic Polynomial multiplication in R_q corresponds to matrix-vector multiplication with a special structure:

$$a(X)\cdot b(X) \leftrightarrow \begin{pmatrix} a_0 & -a_{n-1} & -a_{n-2} & \cdots & -a_1 \\ a_1 & a_0 & -a_{n-1} & \cdots & -a_2 \\ a_2 & a_1 & a_0 & \cdots & -a_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n-1} & a_{n-2} & a_{n-3} & \cdots & a_0 \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ \vdots \\ b_{n-1} \end{pmatrix} \tag{47}$$

One polynomial encodes an entire structured matrix!

4.3.2 Ring-LWE Definition

Now we can define the structured version of LWE:

Definition 31 (Ring Learning With Errors)**.** *Let* $R_q = \mathbb{Z}_q[X]/(X^n+1)$ *and* χ *be an error distribution over* R*.*

- ***Secret:*** $s \in R_q$ *(polynomial with small coefficients)*
- ***Samples:*** *Pairs* (a_i, b_i) *where:*
  - $a_i \in R_q$ *uniformly random*
  - $e_i \leftarrow \chi$ *(polynomial with small coefficients)*
  - $b_i = a_i \cdot s + e_i \in R_q$
- ***Goal:*** *Recover* s

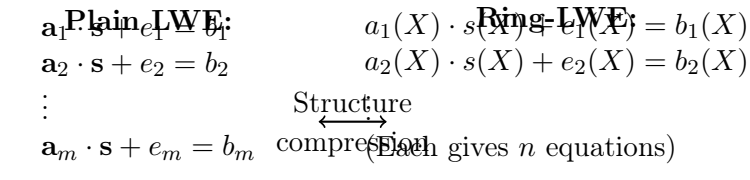


Figure 42: Ring-LWE: Each polynomial sample provides n scalar equations

Operation	Plain LWE	Ring-LWE	Speedup
Key generation	$O(n^2)$	$O(n \log n)$	$100\times$
Encryption	$O(n^2)$	$O(n \log n)$	$100\times$
Public key size	$O(n^2 \log q)$	$O(n \log q)$	$n\times$
Ciphertext size	$O(n \log q)$	$O(n \log q)$	Same

Table 25: Ring structure improves both time and space complexity

Concrete Efficiency Gains The improvements are dramatic:

Example 50 (Number Theoretic Transform in Practice)**.** *Fast polynomial multiplication using NTT (FFT over finite fields):*

- *Direct multiplication:* $O(n^2)$ *operations*
- *NTT method:* $O(n \log n)$ *operations*
- *For* $n = 1024$*: 100× speedup*
- *Hardware friendly: No floating point needed*
- *Example: Kyber uses* $q = 3329 = 13 \cdot 256 + 1$ *(NTT-friendly)*

4.3.3 Security Considerations

Adding algebraic structure always raises security concerns. Here's why Ring-LWE remains secure:

The Additional Structure Ring-LWE adds structure at multiple levels:

1. **Ideal lattices:** Ring elements correspond to ideals
2. **Algebraic relations:** Polynomial multiplication constraints
3. **Symmetries:** Automorphisms of the ring
4. **Subfield structure:** For cyclotomic polynomials

The security question: Do these structures enable new attacks?

Why Ring-LWE Remains Secure

Theorem 17 (Ring-LWE Hardness (Informal))**.** *Breaking Ring-LWE for cyclotomic rings reduces to solving worst-case ideal lattice problems. After 15+ years:*

- *No polynomial-time quantum algorithm known*

- *Best attacks only marginally better than plain LWE*
- *Concrete security estimates nearly identical*

Failed attack attempts:

- **2010-2013:** Subfield attacks → Only work for poor parameters
- **2014-2016:** Ideal lattice exploitation → No advantage found
- **2017-2019:** Algebraic number theory attacks → Marginal improvements
- **2020-2023:** Quantum algebraic attacks → No breakthrough

Remark 33 (The Conservative Response)**.** *The cryptographic community's response to potential algebraic attacks: Module-LWE. This hybrid approach balances efficiency and conservative security.*

Module-LWE: The Best of Both Worlds

Definition 32 (Module Learning With Errors)**.** *Work over R_q^k for small k (typically 2-4):*

- *Elements: Vectors of polynomials $(f_1(X), ..., f_k(X))$*
- *Operations: Matrix-vector multiplication over R_q*
- *Structure: Between plain LWE ($k = n$) and Ring-LWE ($k = 1$)*

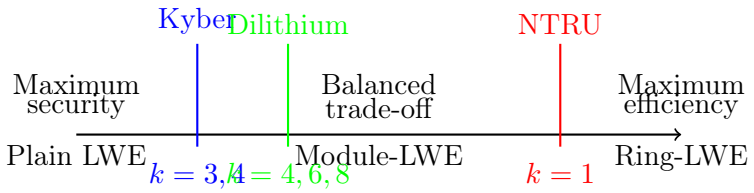


Figure 43: Module-LWE provides flexible security-efficiency trade-offs

Example 51 (Concrete Module-LWE Operations)**.** *Let's see Module-LWE in action with Kyber-768 parameters:*

- *Ring: $R_q = \mathbb{Z}_{3329}[X]/(X^{256} + 1)$*
- *Module dimension: $k = 3$*
- *Secret: $\mathbf{s} = (s_1(X), s_2(X), s_3(X)) \in R_q^3$*
- *Public key matrix: $\mathbf{A} \in R_q^{3\times 3}$*

Key generation:

$$\mathbf{t} = \mathbf{A}\mathbf{s} + \mathbf{e} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \begin{pmatrix} s_1 \\ s_2 \\ s_3 \end{pmatrix} + \begin{pmatrix} e_1 \\ e_2 \\ e_3 \end{pmatrix}$$

Each a_{ij}, s_i, e_i is a polynomial in R_q. The matrix multiplication involves:

- *9 polynomial multiplications (NTT-accelerated)*

- *6 polynomial additions*
- *Total:* $9 \times 256 \times \log(256) \approx 18,000$ *operations*
- *Compare plain LWE:* $768^2 \approx 590,000$ *operations*

Why NIST chose Module-LWE:

- **Conservative:** Less algebraic structure than pure Ring-LWE
- **Efficient:** Much better than plain LWE
- **Flexible:** Adjust k for different security levels
- **Proven:** Worst-case hardness still holds

Parameter Selection Rationale Choosing parameters requires balancing multiple constraints:

Parameter	Constraint	Typical Choice
n (ring dimension)	Power of 2 for NTT	256, 512, 1024
q (modulus)	NTT-friendly, small	3329, 7681, 12289
k (module rank)	Security/efficiency balance	2, 3, 4
η (error bound)	Correctness vs security	2, 3

Table 26: Parameter choices balance security, efficiency, and implementation

Example 52 (Kyber's Parameter Choices)**.** *Why Kyber chose* $n = 256$*,* $q = 3329$*:*

- $n = 256 = 2^8$*: Optimal for AVX2 (8×32-bit registers)*
- $q = 3329 = 13 \cdot 256 + 1$*:*
  - *Smallest prime* $> 12 \cdot \eta \cdot \sqrt{n}$ *(correctness)*
  - $= 1 \bmod 2n$ *(NTT exists)*
  - *Fits in 12 bits (efficient storage)*
- *Module dimensions:* $k \in \{2, 3, 4\}$ *for NIST levels I, III, V*
- *Total dimension:* $256k$ *matches security requirements*

Performance in Practice Real-world benchmarks show the dramatic improvement:

Operation	RSA-2048	Plain LWE	Kyber-768
KeyGen	100 ms	50 ms	0.1 ms
Encapsulation	0.05 ms	20 ms	0.12 ms
Decapsulation	2 ms	15 ms	0.11 ms
Public key	256 B	1.6 MB	1,184 B
Ciphertext	256 B	2 KB	1,088 B

Table 27: Module-LWE (Kyber) is faster than RSA with reasonable key sizes

The Bottom Line on Structure After 15+ years of intensive cryptanalysis:

- Ring structure provides 100-1000× efficiency gain
- No significant security loss discovered
- Module structure offers conservative middle ground
- Enables practical post-quantum deployment

Remark 34 (For the Physics-Minded)**.** *Think of this as symmetry in physics: adding symmetry (ring structure) constrains the system but often makes problems easier to solve. In cryptography, we've found a sweet spot where the efficiency gains are massive but the security loss is negligible. It's like finding a crystal structure that's both stable and useful.*

Example 53 (Migration Impact)**.** *With Ring/Module-LWE, post-quantum transition becomes feasible:*

- *TLS handshake overhead: 2× instead of 1000×*
- *Certificate sizes: 5 KB instead of 5 MB*
- *IoT devices: Possible with constrained memory*
- *Performance: Often faster than current algorithms*

Without this structure, PQC would remain theoretical.

The journey from abstract lattices to practical cryptography required two breakthroughs: LWE showed how to use lattice hardness, and Ring-LWE made it efficient. Module-LWE provides the conservative balance that convinced NIST. This combination gives us quantum-resistant cryptography that works at Internet scale. Now we can examine how NIST transformed these ideas into concrete standards.

4.4 NIST Lattice-Based Standards

After five years of intense global competition, NIST selected three lattice-based algorithms as standards for post-quantum cryptography. This wasn't just an academic exercise—these algorithms will protect trillions of dollars in commerce, secure government communications for decades, and safeguard personal privacy in the quantum era. Understanding why NIST chose these specific algorithms, and how they differ, is crucial for anyone deploying post-quantum security.

4.4.1 NIST Competition Process

The NIST Post-Quantum Cryptography Standardization Process represents the largest coordinated cryptographic evaluation in history.

Figure 44: NIST PQC timeline: 9 years from recognition to deployment

Timeline: From Crisis to Standards

The Competition Details The process was unprecedented in scope and transparency:

- **82 initial submissions:** From academia, industry, and governments worldwide
- **69 complete packages:** Meeting all requirements
- **Public evaluation:** All submissions, analyses, and attacks published
- **3 PQC conferences:** Community feedback and cryptanalysis
- **100+ researchers:** Active participants in evaluation
- **1000+ papers:** Published analyses and improvements

Evaluation Criteria: Beyond Just Security NIST evaluated candidates on multiple dimensions:

Criterion	Description	Weight
Security	Resistance to classical/quantum attack	40%
Performance	Speed on various platforms	20%
Key/signature sizes	Bandwidth and storage impact	20%
Implementation	Side-channel resistance, simplicity	10%
Flexibility	Parameter choices, crypto-agility	5%
Maturity	Years of analysis, confidence level	5%

Table 28: NIST's holistic evaluation went far beyond theoretical security

Example 54 (Real Evaluation Trade-offs)**.** *Rainbow (multivariate signatures):*

- ✓*Tiny signatures (66 bytes!)*
- ✓*Fast verification*
- ✗*Huge public keys (158 KB)*
- ✗*Broken in 2022 by new attack*

Lesson: Multiple criteria prevent single-point optimization.

Algorithm Naming Conventions Understanding the naming helps navigate the standards:

- **CRYSTALS:** Cryptographic Suite for Algebraic Lattices
  - Kyber → ML-KEM (Module Lattice-Based KEM)
  - Dilithium → ML-DSA (Module Lattice-Based Digital Signature Algorithm)
- **Falcon:** Fast Fourier Lattice-based Compact Signatures over NTRU
- **SPHINCS+:** Not lattice-based (hash-based signatures)

Why Multiple Winners? NIST's philosophy: diversification protects against catastrophic failure.

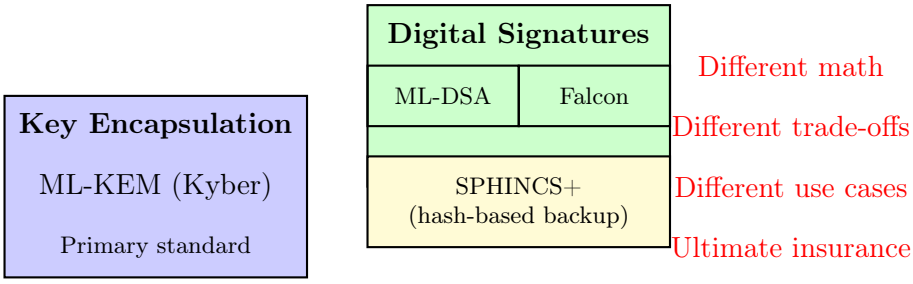


Figure 45: NIST's portfolio approach: Multiple algorithms for resilience

Why SPHINCS+ Wasn't Covered SPHINCS+ is NIST's insurance policy—a hash-based signature scheme:

- **Security:** Based only on hash function security (SHA-256)
- **Quantum-safe:** Even if all lattice problems fall
- **Trade-off:** Large signatures (8-50 KB)
- **Use case:** Emergency backup, firmware signing
- **Not lattice-based:** Outside our current scope

Patent Considerations NIST required all submissions to be available royalty-free:

- All selected algorithms: Patent-free worldwide
- Some candidates withdrawn due to IP issues
- NTRU patents expired before selection
- Clear legal status critical for global adoption

4.4.2 CRYSTALS-Kyber (ML-KEM)

Kyber, now standardized as ML-KEM (Module Lattice-Based Key Encapsulation Mechanism), is NIST's primary choice for key establishment.

Why Kyber Won Kyber achieved the best overall balance:

- **Security:** Conservative Module-LWE foundation
- **Efficiency:** Competitive with elliptic curves
- **Simplicity:** Clean implementation without footguns
- **Flexibility:** Three parameter sets for different needs

Technical Foundation **Key innovations:**

- **Compression:** Reduce ciphertext size by dropping LSBs
- **Error reconciliation:** Ensure both parties get same key
- **CCA security:** Fujisaki-Okamoto transform built-in
- **Deterministic noise:** Reduces randomness requirements

Algorithm 13 ML-KEM (Kyber) Overview

Parameters: $(n, k, q, \eta_1, \eta_2, d_u, d_v)$
Ring: $R_q = \mathbb{Z}_q[X]/(X^n + 1)$ where $n = 256$, $q = 3329$
KeyGen():
  $\mathbf{A} \leftarrow R_q^{k \times k}$ (seed expansion)
  $\mathbf{s}, \mathbf{e} \leftarrow \beta_\eta^k$ (small secrets)
  $\mathbf{t} = \mathbf{A}\mathbf{s} + \mathbf{e}$
  $pk = (\mathbf{t}, \text{seed}(\mathbf{A}))$, $sk = \mathbf{s}$
Encapsulate(pk):
  $\mathbf{r}, \mathbf{e}_1, e_2 \leftarrow \beta_\eta$ (small noise)
  $\mathbf{u} = \mathbf{A}^T\mathbf{r} + \mathbf{e}_1$
  $v = \mathbf{t}^T\mathbf{r} + e_2 + \lfloor q/2 \rceil \cdot m$
  $c = (\text{Compress}(\mathbf{u}), \text{Compress}(v))$
  $K = \text{KDF}(m, H(c))$
Decapsulate(sk, c):
  Decompress and compute $v - \mathbf{s}^T\mathbf{u}$
  Recover m and recompute K

Parameter Set	k	Security	PK (bytes)	CT (bytes)	SS (bytes)
ML-KEM-512	2	128-bit quantum	800	768	32
ML-KEM-768	3	192-bit quantum	1,184	1,088	32
ML-KEM-1024	4	256-bit quantum	1,568	1,568	32
For comparison:					
X25519	—	128-bit classical	32	32	32
RSA-2048	—	112-bit classical	256	256	256

Table 29: ML-KEM parameters: 30-40× larger than ECC but still practical

Parameter Sets and Security

Performance Characteristics Real-world benchmarks on Intel Core i7-8700K @ 3.7GHz, AVX2 enabled, GCC 11.2:

Operation	ML-KEM-768	X25519	Ratio
KeyGen	35 µs	45 µs	0.8×
Encapsulate	44 µs	45 µs	1.0×
Decapsulate	40 µs	45 µs	0.9×

Table 30: ML-KEM matches or beats elliptic curve performance!

Remark 35 (The Surprise)**.** *Lattice operations (integer arithmetic + NTT) are so efficient that ML-KEM often outperforms elliptic curves despite working in dimension 768 vs. 256-bit scalars. This wasn't expected when the competition began.*

Hybrid Mode Deployment For transitional security, NIST recommends hybrid modes:

Definition 33 (Hybrid Key Exchange)**.** *Combine classical and post-quantum algorithms:*

$$\textit{SharedSecret} = \textit{KDF}(\textit{ECDH\_Secret}||\textit{ML-KEM\_Secret})$$

Why hybrid modes matter:

- **Belt and suspenders:** Security if either algorithm holds
- **Transition period:** Maintain compatibility
- **Standards:** X25519+ML-KEM-768 for TLS 1.3
- **Overhead:** Only 1.3 KB extra vs pure ML-KEM

Example 55 (TLS 1.3 Hybrid Handshake)**.** *Client and server negotiate:*

- *Classical: X25519 (32 byte public key)*
- *Post-quantum: ML-KEM-768 (1,184 byte public key)*
- *Combined: 1,216 bytes total*
- *Security: Best of both worlds*
- *Performance: Negligible impact (< 5% slower)*

4.4.3 CRYSTALS-Dilithium (ML-DSA)

Dilithium, standardized as ML-DSA (Module Lattice-Based Digital Signature Algorithm), is NIST's primary signature scheme.

The Signature Challenge Digital signatures from lattices are harder than encryption:

- Signatures reveal information about the secret key
- Must ensure no leakage across many signatures
- Size-security trade-offs more challenging
- Need rejection sampling for security

Algorithm 14 ML-DSA (Dilithium) Signing - Simplified

Input: Message M, secret key $(\mathbf{s}_1, \mathbf{s}_2)$
Output: Signature $(\mathbf{z}, \mathbf{h}, c)$
repeat
  $\mathbf{y} \leftarrow S_\gamma$ ▷ Large masking vector
  $\mathbf{w} = \mathbf{A}\mathbf{y}$ ▷ Commitment
  $c = H(\mathbf{w}_{\text{high}} || M)$ ▷ Challenge
  $\mathbf{z} = \mathbf{y} + c\mathbf{s}_1$ ▷ Response
  if $||\mathbf{z}||_\infty \geq \gamma - \beta$ **then**
    restart ▷ Abort to prevent leakage!
  endif
until signature accepted
Compute hint $\mathbf{h}$ for verification
return $(\mathbf{z}, \mathbf{h}, c)$

Dilithium's Approach: Fiat-Shamir with Aborts **Key insight:** Rejection sampling ensures the signature distribution is independent of the secret key!

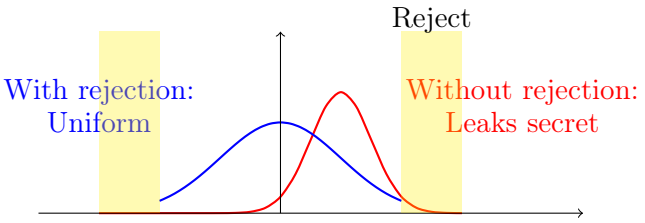


Figure 46: Rejection sampling: Sacrifice efficiency for perfect security

Level	Security	PK (bytes)	Sig (bytes)	Sign/s	Verify/s
ML-DSA-44	128-bit	1,312	2,420	3,700	9,100
ML-DSA-65	192-bit	1,952	3,309	2,500	6,200
ML-DSA-87	256-bit	2,592	4,627	1,900	4,600
For comparison:					
Ed25519	128-bit classical	32	64	24,000	8,000
RSA-2048	112-bit classical	256	256	500	17,000

Table 31: ML-DSA: Larger signatures but competitive performance. Benchmarks on Intel Core i7-8700K.

Parameter Sets and Trade-offs **Why these trade-offs are acceptable:**

- Signatures are usually transmitted once, verified many times
- 3 KB is tiny compared to typical signed data
- Verification speed matches current algorithms
- Signing speed sufficient for most applications

Example 56 (TLS Certificate Chain Impact)**.** *Current TLS with RSA-2048:*

- *3 certificates × 256 bytes = 768 bytes signatures*
- *Total certificate chain: 4 KB*

With ML-DSA-44:

- *3 certificates × 2,420 bytes = 7,260 bytes signatures*
- *Total certificate chain: 11 KB*
- *Impact: 7 KB extra per handshake (acceptable)*

4.4.4 Falcon

Falcon represents a different approach: using NTRU lattices and advanced sampling techniques to achieve the smallest signatures.

The NTRU Heritage Falcon builds on NTRU, one of the oldest lattice schemes (1996):

Definition 34 (NTRU Lattice Structure)**.** *Work in ring $R = \mathbb{Z}[X]/(X^n + 1)$ with:*

- *Public key: $h = f^{-1} \cdot g$ where f, g small*
- *Secret key: The "good" basis (f, g)*
- *Signing: Use secret basis to solve CVP efficiently*
- *Security: Finding good basis from h is hard*

Gaussian Sampling: The Technical Challenge Falcon's key operation is sampling from a discrete Gaussian over a lattice:

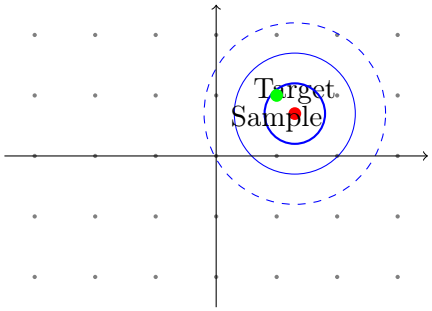


Figure 47: Falcon samples lattice points near targets with Gaussian probability

WARNING: Falcon's Implementation Complexity
Falcon is notoriously difficult to implement securely:

- **Floating-point arithmetic:** Requires IEEE 754 double precision (53+ bits)
- **Platform dependence:** Different FP behavior across CPUs
- **Side-channel vulnerability:** FP operations leak timing information
- **No constant-time guarantee:** Branch-free implementation extremely difficult
- **Memory access patterns:** Can leak secret key information

Listing 4: Falcon's Precision Requirements - DO NOT USE IN PRODUCTION WITHOUT EXPERTISE

```
// Gaussian sampling requires high precision
typedef struct {
    double real;  // MUST be IEEE 754 double
    double imag;  // Rounding errors can break security
} complex;

// Fast Fourier sampling over complex numbers
void FFT(complex *f, unsigned logn) {
    // WARNING: Not constant-time!
    // Side-channel attacks possible
    // Different results on different platforms
    // Requires expert implementation
}

// Compare to Dilithium:
// Only needs uint32_t arithmetic
// Naturally constant-time
// Platform independent
```

Falcon's Trade-offs

Example 57 (Where Falcon Shines)**.** *Blockchain transaction:*

- *Thousands of signatures stored forever*

Aspect	Advantage	Disadvantage
Signature size	Smallest (666-1,280 bytes)	Still 10× Ed25519
Verification	Very fast	—
Public key	Compact (897-1,793 bytes)	—
Signing	—	Complex implementation
Side-channels	—	Extremely hard to protect
Platforms	—	Needs FP unit
Implementation	—	Few correct implementations

Table 32: Falcon: Optimal sizes at the cost of severe implementation complexity

- *Size matters more than signing speed*
- *Falcon-512: 666 bytes vs. Dilithium: 2,420 bytes*
- *Saves 1.7 KB per transaction*
- *1 million transactions: 1.7 GB saved*
- *BUT: Requires expert implementation team*

NIST's Warning on Falcon NIST's official stance on Falcon is cautious:

> "Falcon is suitable for applications where signature size is critical and implementation expertise is available. Organizations should carefully evaluate whether they have the necessary expertise to implement Falcon securely. For most applications, ML-DSA is the recommended choice."

Real-world Falcon challenges:

- **ARM Cortex-M:** No FPU, software emulation too slow
- **Intel vs AMD:** Different FP rounding can break interoperability
- **Timing attacks:** Multiple academic papers show vulnerabilities
- **Reference implementation:** Not production-ready
- **Hardware support:** Requires custom FP units for safety

Use Case	Recommendation	Reason
TLS/HTTPS	ML-DSA	Easier implementation
Code signing	ML-DSA	Broad platform support
Email (S/MIME)	ML-DSA	Standard compliance
Blockchain	Falcon (maybe)	Size critical, but risky
IoT devices	ML-DSA	No FP needed
HSMs	Falcon (with care)	Controlled environment
General purpose	ML-DSA	Always safer choice

Table 33: Choose ML-DSA by default, Falcon only with expert teams

Deployment Recommendations

Remark 36 (The Practical Reality)**.** *Despite Falcon's elegance and efficiency, ML-DSA will dominate deployment. Implementation simplicity trumps optimal parameters for most applications. Falcon remains important for specialized uses and as a backup if Module-LWE faces unexpected attacks. However, any organization considering Falcon must budget for security audits and specialized expertise.*

Looking Forward The selection of three lattice-based algorithms (plus SPHINCS+ as backup) reflects both confidence and caution:

- **Confidence:** Lattices are ready for deployment
- **Caution:** Multiple approaches provide insurance
- **Reality:** ML-KEM and ML-DSA will handle 95% of uses
- **Falcon:** Niche applications with expert teams only
- **SPHINCS+:** Ultimate fallback if all lattices fail
- **Future:** Continued research may yield better trade-offs

These standards represent the culmination of 25+ years of lattice cryptography research. From Ajtai's worst-case/average-case connection through LWE to practical implementations, we now have quantum-resistant algorithms ready to protect the digital world. The challenge shifts from mathematics to engineering: deploying these algorithms at Internet scale while managing the complexity, especially for schemes like Falcon that demand exceptional implementation care.

5 Practical Deployment Considerations

Mathematical security means nothing if implementation flaws leak secrets. The transition to post-quantum cryptography isn't just about swapping algorithms—it's about navigating a minefield of implementation challenges that can completely undermine security. A single timing variation of nanoseconds can leak private keys. Poor randomness can make unbreakable schemes trivial to attack. And the larger keys and signatures of PQC stress every system they touch. This section covers what goes wrong in practice and how to avoid catastrophe.

5.1 Implementation Challenges

5.1.1 Side-Channel Resistance

Here's a sobering fact: more cryptographic systems are broken through implementation flaws than mathematical breakthroughs. Side-channel attacks—extracting secrets through timing, power consumption, or electromagnetic emissions—pose particular challenges for lattice-based cryptography.

The Constant-Time Imperative Modern cryptographic implementations must run in constant time: execution time cannot depend on secret data.

Example 58 (A Timing Disaster)**.** *Consider this innocent-looking modular reduction:*

```
// INSECURE: Timing depends on secret value
int16_t barrett_reduce(int32_t a) {
    int32_t t = (a * BARRETT_FACTOR) >> BARRETT_SHIFT;
    t = a - t * KYBER_Q;
    if (t >= KYBER_Q) {   // DANGER: Secret-dependent branch
```

```
        t -= KYBER_Q;
    }
    return t;
}
```

The branch takes different time depending on the value. An attacker measuring timing over millions of operations can recover secret keys!

Constant-time implementation rules:

1. No secret-dependent branches
2. No secret-dependent memory access patterns
3. No secret-dependent instruction selection
4. No early termination based on secrets

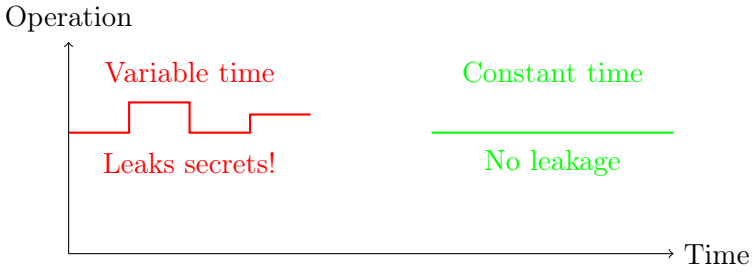


Figure 48: Constant-time execution: The same duration regardless of secret values

Lattice-Specific Side-Channel Challenges Lattice cryptography introduces unique side-channel risks:

1. Polynomial Multiplication Timing

- NTT butterflies must be constant-time
- Modular reduction after multiplication critical
- Cache timing attacks on coefficient access

Listing 5: Constant-Time NTT Butterfly

```
// SECURE: Constant-time butterfly operation
void butterfly(int16_t *a, int16_t *b, int16_t w) {
    int32_t t = montgomery_multiply(*b, w);
    // No branches, just arithmetic
    *b = *a - t;
    *a = *a + t;
    // Conditional reduction without branches
    *a -= (KYBER_Q & -((*a >= KYBER_Q)));
    *b += (KYBER_Q & -((*b < 0)));
}
```

2. Rejection Sampling in Signatures

3. Error Sampling Leakage

The discrete Gaussian sampling in lattice schemes is particularly vulnerable:

Algorithm 15 Side-Channel Resistant Rejection Sampling

INSECURE approach:
while $\|\mathbf{z}\|_\infty \geq \gamma - \beta$ **do**
    Resample ▷ Iterations leak secret!
end while
SECURE approach:
for $i = 1$ to MAX_ATTEMPTS **do**
    Sample candidate
    accept $\leftarrow$ CT_Compare($\|\mathbf{z}\|_\infty, \gamma - \beta$)
    result $\leftarrow$ CT_Select(accept, $\mathbf{z}$, result)
end for
▷ Always runs MAX_ATTEMPTS iterations

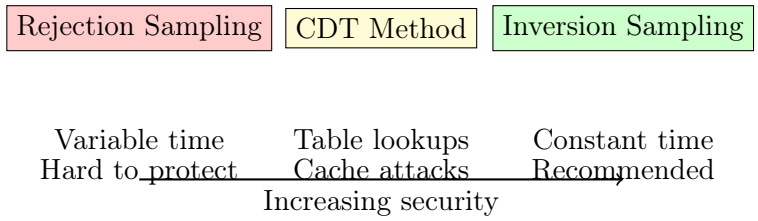


Figure 49: Gaussian sampling methods: Trade security for efficiency

The Falcon Challenge: Floating-Point Side Channels Falcon's use of floating-point arithmetic creates unique vulnerabilities:

- FP operations have variable timing on many CPUs
- Denormalized numbers cause timing variations
- Rounding modes affect results
- FP exceptions can leak information

Example 59 (FP Timing Attack)**.**

```
// Floating-point operations timing depends on values
double gaussian_sample(double sigma, double center) {
    double x = random_double();
    // DANGER: exp() timing varies with input
    double prob = exp(-(x - center)^2 / (2 * sigma^2));
    // DANGER: comparison timing may vary
    if (random_double() < prob) {
        return x;  // Variable number of iterations!
    }
}
```

Hardware vs Software Trade-offs **Recommended approaches:**

- **Hardware:** Custom ASIC/FPGA for high-security applications
- **Software:** Assembly with careful constant-time verification
- **Hybrid:** Hardware RNG + software implementation

Aspect	Hardware	Software
Timing control	Excellent	Challenging
Power analysis	Vulnerable	Less exposed
Cost per algorithm	High	Low
Flexibility	Poor	Excellent
Performance	Optimal	Good with SIMD
Certification	Easier	Harder

Table 34: Hardware provides better side-channel resistance at higher cost

5.1.2 Random Number Generation

Lattice-based cryptography is extraordinarily hungry for randomness. Poor RNG quality doesn't just weaken security—it can completely destroy it.

Operation	RSA-2048	Kyber-768
Key generation	256 bytes	3,200 bytes
Signature	32 bytes	4,096 bytes
Encryption	32 bytes	1,152 bytes

Table 35: Lattice schemes consume 10-100× more random bytes

Why Lattice Crypto Needs More Randomness **What needs randomness:**

- Error vectors (thousands of samples)
- Ephemeral secrets
- Masking values for signatures
- Rejection sampling loops

Quality Requirements: Not All Randomness is Equal

Definition 35 (Cryptographic RNG Requirements)**.** *For lattice-based cryptography, the RNG must provide:*

1. ***Statistical uniformity:*** *Indistinguishable from true random*
2. ***Unpredictability:*** *Cannot predict future outputs*
3. ***Backtracking resistance:*** *Cannot recover past outputs*
4. ***High throughput:*** *MB/s for key generation*

Example 60 (RNG Failure Impact)**.** *Debian OpenSSL disaster (2008) in PQC context:*

- *Weak RNG reduced entropy to 15 bits*
- *For RSA: 32,768 possible keys (bad)*
- *For Kyber: Error distribution destroyed*
- *Result: Linear algebra attack recovers secret instantly*
- *Lesson: Lattice crypto fails catastrophically with bad RNG*

Algorithm 16 Discrete Gaussian Sampling

Input: Standard deviation σ, center μ
Output: Sample from $D_{\mathbb{Z},\sigma,\mu}$
Method 1: Rejection Sampling
while true **do**
    $x \leftarrow \text{UniformInteger}([-\tau\sigma, \tau\sigma])$
    $p \leftarrow \exp(-(x-\mu)^2/(2\sigma^2))$
    If $\text{UniformReal}() < p$: return x
end while
Method 2: CDT (Cumulative Distribution Table)
$r \leftarrow \text{UniformInteger}([0, 2^{64}-1])$
Binary search CDT for interval containing r
Return corresponding value
Method 3: Convolution (Kyber approach)
Sum 2-3 centered binomial samples
Approximates Gaussian, constant-time

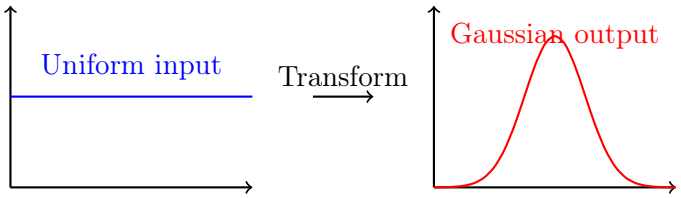


Figure 50: RNG challenge: Transform uniform bits to Gaussian distribution

Gaussian Sampling: The Distribution Challenge Most lattice schemes need Gaussian-distributed errors, not uniform:

Deterministic Variants: Removing RNG Dependence Modern PQC often uses deterministic variants to reduce RNG exposure:

Listing 6: Deterministic Key Generation

```
// Traditional: Needs continuous RNG
void keygen_random(pk, sk) {
    s = sample_random();          // RNG call
    e = sample_random();          // RNG call
    A = expand_random_seed();     // RNG call
    pk = (A, As + e);
    sk = s;
}

// Deterministic: Single RNG seed
void keygen_deterministic(pk, sk, seed) {
    // Expand single 32-byte seed deterministically
    (seedA, seedS) = SHA3-512(seed);
    A = expand_seed(seedA);
    s = sample_from_seed(seedS, 0);
    e = sample_from_seed(seedS, 1);
    pk = (A, As + e);
    sk = s;
}
```

Advantages:

- Reproducible key generation
- Easier testing and debugging
- Less RNG exposure
- Enables key recovery from seed

Risks:

- Seed compromise loses everything
- Side-channels on seed expansion
- Must protect seed lifetime

5.1.3 Key and Signature Sizes

The elephant in the room: post-quantum cryptography is big. Understanding the size impact is crucial for system design.

Algorithm	Public Key	Private Key	Signature	Ciphertext
Current Algorithms				
RSA-2048	256 B	256 B	256 B	256 B
RSA-3072	384 B	384 B	384 B	384 B
ECDSA-P256	64 B	32 B	64 B	—
ECDH-P256	64 B	32 B	—	64 B
Ed25519	32 B	32 B	64 B	—
Post-Quantum Algorithms				
ML-KEM-768	1,184 B	2,400 B	—	1,088 B
ML-DSA-65	1,952 B	4,032 B	3,309 B	—
Falcon-512	897 B	1,281 B	666 B	—
SPHINCS+-128s	32 B	64 B	7,856 B	—
Size Increase Factor				
ML-KEM vs ECDH	18.5×	75×	—	17×
ML-DSA vs Ed25519	61×	126×	52×	—
Falcon vs Ed25519	28×	40×	10×	—

Table 36: PQC size explosion: 10-100× larger than current algorithms

The Stark Reality: Size Comparison

Protocol Impact: Where Size Hurts 1. TLS Handshake Explosion

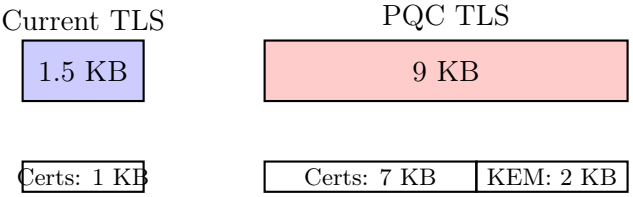


Figure 51: TLS handshake size increases 6× with PQC

2. Certificate Chain Explosion

Example 61 (Real Certificate Chain)**.** *Current (RSA-2048):*

- *Root: 1.5 KB (256 B signature)*
- *Intermediate: 1.5 KB (256 B signature)*
- *Leaf: 1.5 KB (256 B signature)*
- *Total: 4.5 KB*

With ML-DSA-65:

- *Root: 4.8 KB (3.3 KB signature)*
- *Intermediate: 4.8 KB (3.3 KB signature)*
- *Leaf: 4.8 KB (3.3 KB signature)*
- *Total: 14.4 KB (3.2× increase)*

3. IoT and Embedded Systems

Constraint	Typical IoT	ML-KEM-768	Feasible?
Flash (code)	256 KB	+40 KB	Maybe
RAM (runtime)	32 KB	8 KB	Yes
Key storage	4 KB	3.6 KB	Tight
Bandwidth	250 kbps	9 KB handshake	Challenging
Battery (per op)	1 mJ	0.5 mJ	Yes

Table 37: PQC on constrained devices: Possible but challenging

Storage and Bandwidth Considerations **Database Impact:**

- User authentication DB with 1M users
- Current (Ed25519): 32 MB public keys
- PQC (ML-DSA): 1.95 GB public keys
- 61× storage increase

Network Impact:

- Mobile app checking 100 signatures/day
- Current: 6.4 KB/day
- PQC: 331 KB/day
- Monthly: 10 MB extra data usage

Blockchain Catastrophe:

- Bitcoin transaction: 250 bytes
- With PQC signatures: 3,500 bytes
- Blockchain size growth: 14× faster
- Full node storage: TB → 14 TB

Mitigation Strategies **1. Compression Techniques**

- Compress public keys when storing
- Use seed + expansion for deterministic keys
- Domain separation for multiple keys
- Typical savings: 20-30%

2. Hybrid Modes During Transition

Listing 7: Size-Optimized Hybrid Mode

```
// Naive hybrid: RSA + PQC (size = both)
struct hybrid_sig_naive {
    uint8_t rsa_sig[256];
    uint8_t pqc_sig[3309];
}; // Total: 3,565 bytes

// Optimized: Sign hash with both
struct hybrid_sig_opt {
    uint8_t data_hash[32];
    uint8_t rsa_sig[256];
    uint8_t pqc_sig[666];  // Use Falcon
}; // Total: 954 bytes
```

3. Protocol Adaptations

- Cache certificates aggressively
- Use session resumption in TLS
- Batch signature verification
- Consider SPHINCS+ only for root CAs

Remark 37 (The Size Reality Check)**.** *The size increase is real and painful. No clever optimization makes PQC as small as ECC. Success requires:*

- *Accepting the size increase as necessary*
- *Architecting systems to minimize impact*
- *Using the right algorithm for each use case*
- *Planning infrastructure upgrades early*

The alternative—broken cryptography—is worse than larger keys.

Example 62 (Success Story: Cloudflare's Deployment)**.** *Cloudflare successfully deployed PQC to millions of websites:*

- *Hybrid mode: ECDH + Kyber*
- *Increased handshake: 1.2 KB → 2.9 KB*
- *Performance impact: <0.1 ms*
- *User experience: Unchanged*

- *Lesson: Size increase manageable with good engineering*

The implementation challenges are real but surmountable. Side-channel protection requires expertise and careful coding. Random number generation needs more attention than ever. And yes, keys and signatures are much larger. But with proper engineering, post-quantum cryptography is deployable today. The next section examines migration strategies to make this transition successful.

5.2 Migration Strategies

The post-quantum migration represents the largest cryptographic transition in history. Every encrypted connection, every digital signature, every authentication protocol must change. Unlike previous migrations that affected specific algorithms, this touches everything using public-key cryptography. The challenge isn't just technical—it's organizational, economic, and urgent. Organizations that start now will transition smoothly. Those that wait risk catastrophic exposure when quantum computers arrive.

5.2.1 Hybrid Modes

The safest path to post-quantum security runs through hybrid cryptography: using both classical and post-quantum algorithms together. This isn't overcaution—it's learning from history.

Why Hybrid? Learning from Dual EC DRBG

Example 63 (The Cautionary Tale)**.** *Dual EC DRBG (2006-2013):*

- *NSA-designed random number generator*
- *Based on elliptic curves (considered secure)*
- *Standardized by NIST*
- *Later revealed: Deliberate backdoor*
- *Lesson: Even standardized algorithms can fail*

Hybrid modes protect against:

- **Classical breaks:** If PQC algorithm broken classically
- **Implementation flaws:** Bugs in new PQC code
- **Quantum timeline uncertainty:** If quantum computers arrive early
- **Standard changes:** If NIST updates algorithms

Hybrid Construction Methods Method 1: Concatenation (Simple but Large) Method 2: Nested (Better Integration)

Security Analysis of Hybrid Modes

Theorem 18 (Hybrid Security)**.** *For hybrid scheme $H = (C, P)$ combining classical C and post-quantum P:*

- *If C is (t_c, ϵ_c)-secure and P is (t_p, ϵ_p)-secure*
- *Then H is (t, ϵ)-secure where:*
- $t = \min(t_c, t_p)$ *and* $\epsilon \leq \epsilon_c + \epsilon_p$

Real security levels:

Algorithm 17 Hybrid Key Exchange - Concatenation

Client:
  Generate $(pk_{ecc}, sk_{ecc}) \leftarrow$ ECDH.KeyGen()
  Generate $(pk_{pqc}, sk_{pqc}) \leftarrow$ Kyber.KeyGen()
  Send (pk_{ecc}, pk_{pqc}) to server
Server:
  $K_{ecc} \leftarrow$ ECDH.Encapsulate(pk_{ecc})
  $(c_{pqc}, K_{pqc}) \leftarrow$ Kyber.Encapsulate(pk_{pqc})
  $K_{final} = \text{KDF}(K_{ecc} || K_{pqc})$
  Send (c_{ecc}, c_{pqc}) to client
Security: Must break BOTH to get K_{final}

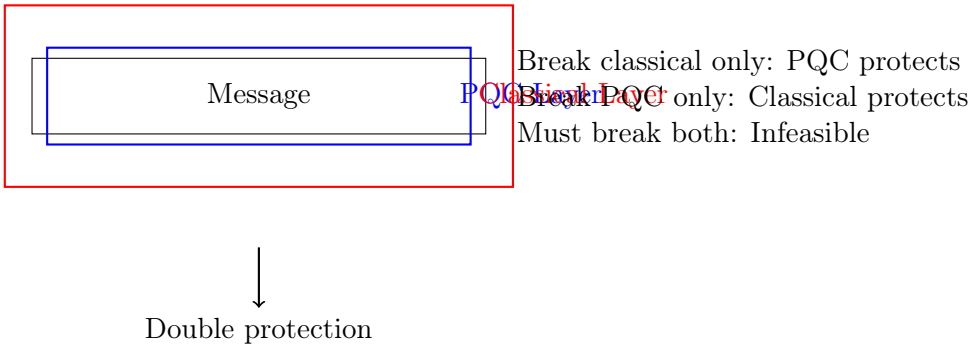


Figure 52: Nested hybrid: Each layer independently protects the data

Performance Impact The cost of hybrid modes:

Example 64 (Google's CECPQ2 Experiment)**.** *Google tested hybrid ECDH+NTRU in Chrome (2019):*

- *30% of TLS connections used hybrid*
- *Median latency increase: 1 ms*
- *95th percentile: 20 ms increase*
- *User complaints: Zero*
- *Conclusion: Hybrid deployment feasible at scale*

5.2.2 Timeline and Urgency

The quantum threat creates an unprecedented situation: we must protect against computers that don't yet exist. Understanding the timeline is crucial for appropriate urgency.

The Quantum Timeline: Best Estimates **Expert predictions (2024):**

- **Optimistic:** Cryptographically relevant QC by 2030
- **Realistic:** 2035-2040 for RSA-2048 threats
- **Conservative:** 2040-2050 for large-scale breaks
- **IBM Roadmap:** 100,000 qubits by 2033

Configuration	Classical	Quantum	Confidence
ECDH-P256 only	128-bit	0-bit	High
Kyber-768 only	180-bit	128-bit	Medium
Hybrid both	128-bit	128-bit	Highest

Table 38: Hybrid provides quantum resistance without sacrificing classical security

Operation	ECDH	Kyber	Hybrid	Overhead
Key generation	45 µs	35 µs	80 µs	1.78×
Encapsulation	45 µs	44 µs	89 µs	1.98×
Decapsulation	45 µs	40 µs	85 µs	1.89×
Handshake size	96 B	2,272 B	2,368 B	24.7×

Table 39: Hybrid overhead: 2× time, dominated by PQC size

Why Start Now? The Mosaic Theory

Definition 36 (Mosaic Theory of Migration)**.** *Cryptographic migration isn't one project—it's thousands of interconnected changes that must happen in sequence:*

- *Standards must finalize (2024)* ✓
- *Libraries must implement (2024-2025)*
- *Products must integrate (2025-2027)*
- *Deployments must complete (2027-2030)*
- *Legacy must sunset (2030-2035)*

Example 65 (The 10-Year Rule)**.** *SHA-1 deprecation timeline:*

- *2005: First collision attacks published*
- *2014: Chrome starts warning*
- *2017: First practical collision*
- *2020: Windows stops SHA-1 support*
- *2024: Still found in legacy systems*
- *Total: 19 years and counting!*

PQC migration is 100× more complex.

Lessons from Previous Migrations Success: AES Adoption

- Clear deadline: DES key too small
- Drop-in replacement: Same interface
- Performance gain: AES faster than 3DES
- Result: Smooth transition

Failure: IPv6 Adoption

- No hard deadline

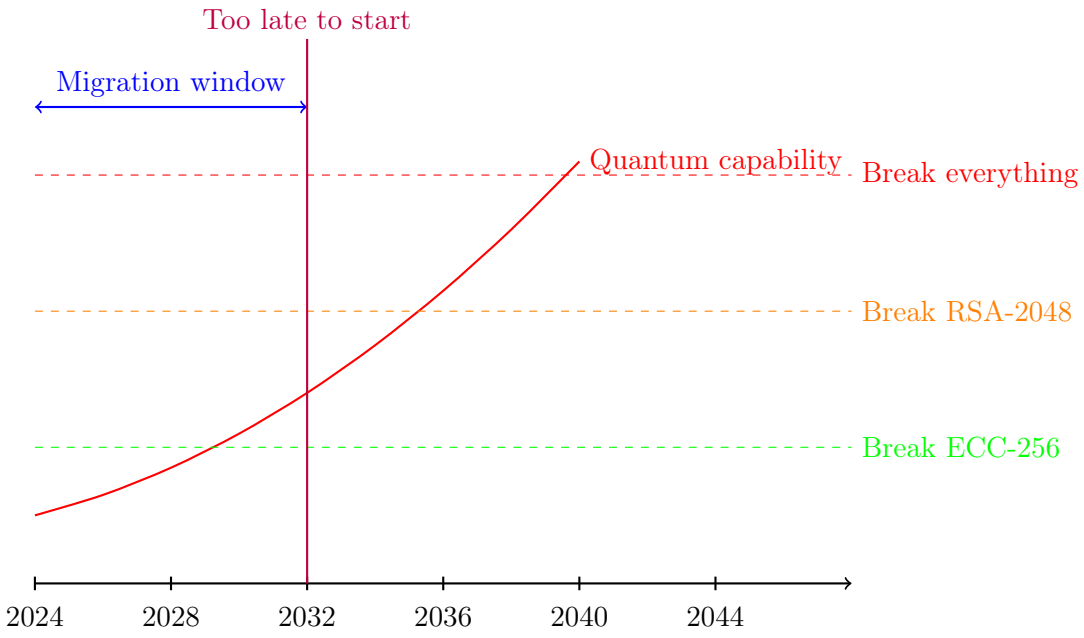


Figure 53: Quantum timeline: Migration must complete before quantum computers arrive

- Requires infrastructure changes
- No immediate benefit
- Result: 25 years, still incomplete

PQC resembles IPv6 more than AES:

- No immediate threat visible
- Requires protocol changes
- Performance degradation
- Risk: Similar 20+ year transition

Data Type	Lifetime	Risk Level	Action Needed
Government secrets	25-50 years	Critical	Immediate
Healthcare records	100 years	Critical	Immediate
Financial records	7-10 years	High	By 2025
IP/Trade secrets	20 years	High	By 2025
Personal data	Lifetime	Medium	By 2027
Session keys	Hours	Low	By 2030

Table 40: Data lifetime determines migration urgency

Risk Assessment: The "Harvest Now, Decrypt Later" Reality

Remark 38 (The Uncomfortable Truth)**.** *If your data needs protection beyond 2035, it's already at risk. Adversaries are likely harvesting encrypted data today for future decryption. Every day of delay is another day of data exposed to future quantum attack.*

5.2.3 Practical Steps

Here's your migration playbook—concrete steps every organization should take.

Step 1: Cryptographic Inventory You can't protect what you don't know exists.

Algorithm 18 Crypto Discovery Process

Phase 1: Automated Discovery
  Scan codebases for crypto libraries
  Analyze network traffic for TLS/SSH
  Check certificates and key stores
  Inventory hardware security modules
Phase 2: Manual Review
  Interview development teams
  Review security architecture
  Check third-party dependencies
  Document custom implementations
Phase 3: Risk Classification
  Map algorithms to use cases
  Identify quantum-vulnerable components
  Prioritize by data sensitivity
  Create migration roadmap

Example 66 (Typical Enterprise Discovery)**.** *Medium enterprise (5,000 employees) crypto inventory:*

- *2,847 TLS certificates (RSA/ECDSA)*
- *156 code signing certificates*
- *89 internal applications using crypto*
- *1,293 API keys (various algorithms)*
- *23 different crypto libraries*
- *7 hardware security modules*
- *Surprise: 45% unknown to security team!*

Step 2: Testing Methodology **Test Environment Setup:**

Listing 8: PQC Testing Framework

```
# Create isolated test environment
docker network create pqc-test

# Run PQC-enabled server
docker run -d --name pqc-server \
  --network pqc-test \
  -e CRYPTO_MODE=hybrid \
  -e PQC_ALGO=kyber768 \
  pqc-test-server:latest

# Test client compatibility
```

```
for client in legacy current pqc-enabled; do
  docker run --rm --network pqc-test \
    pqc-test-client:$client \
    --target pqc-server \
    --measure-performance
done
```

Key Testing Areas:

1. **Functionality:** Do connections succeed?
2. **Performance:** Latency and throughput impact
3. **Compatibility:** Legacy system interactions
4. **Scalability:** Load testing with larger keys
5. **Failure modes:** Graceful degradation

Step 3: Rollout Strategies **Strategy 1: Canary Deployment**

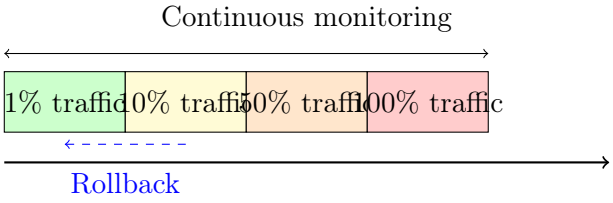


Figure 54: Canary deployment: Gradual rollout with monitoring

Strategy 2: Service-Based Migration

- **Phase 1:** Internal services (low risk)
- **Phase 2:** B2B connections (controlled)
- **Phase 3:** Customer-facing (careful)
- **Phase 4:** Legacy integration (complex)

Step 4: Building Crypto-Agility The future demands systems that can change algorithms without redesign.

Definition 37 (Crypto-Agility)**.** *The ability to rapidly transition between cryptographic algorithms without:*

- *Changing application code*
- *Disrupting service*
- *Breaking compatibility*
- *Manual intervention*

Listing 9: Crypto-Agile Design Pattern

```
// BAD: Algorithm-specific code
void encrypt_data_rsa(data, rsa_key) { ... }
void encrypt_data_kyber(data, kyber_key) { ... }

// GOOD: Algorithm-agnostic interface
typedef struct {
    enum { RSA, ECDH, KYBER, HYBRID } type;
    void *key;
    size_t key_len;
} crypto_key_t;

void encrypt_data(data, crypto_key_t *key) {
    switch(key->type) {
        case RSA: return rsa_encrypt(data, key);
        case KYBER: return kyber_encrypt(data, key);
        case HYBRID: return hybrid_encrypt(data, key);
    }
}

// Configuration-driven selection
key_type = config_get("crypto.algorithm");
```

Crypto-Agility Checklist:

- ✓Algorithms identified by OIDs, not hardcoded
- ✓Key sizes parameterized
- ✓Protocol negotiation supported
- ✓Multiple algorithms co-exist
- ✓Monitoring for algorithm usage
- ✓Automated rollback capability

Year	Milestone	Actions
2024	Foundation	Inventory, team training, pilot projects
2025	Implementation	Deploy hybrid modes, update libraries
2026	Expansion	Customer-facing systems, partner integration
2027	Acceleration	Legacy migration, full hybrid deployment
2028	Transition	Begin disabling classical-only modes
2029	Hardening	PQC-only for sensitive data
2030	Completion	Quantum-safe across organization

Table 41: Seven-year migration: Aggressive but achievable

Migration Timeline Template

Remark 39 (The Migration Imperative)**.** *Post-quantum migration isn't optional—it's existential. Organizations that complete migration will survive the quantum era. Those that don't face catastrophic exposure. The technical challenges are solvable. The timeline is tight but achievable. The only unforgivable mistake is waiting to start.*

Example 67 (Success Story: Cloudflare's Leadership)**.** *Cloudflare deployed PQC to 20+ million websites:*

- *Started: 2019 (before standards!)*
- *Hybrid deployment: 2022*
- *Full coverage: 2023*
- *Customer impact: Minimal*
- *Lesson: Early movers gain experience and confidence*

The migration challenge is real but manageable. Hybrid modes provide safety during transition. The timeline demands urgency without panic. And concrete steps turn an overwhelming challenge into executable projects. The next section examines real-world deployments to learn from early adopters.

5.3 Real-World Case Studies

Theory meets reality when post-quantum cryptography deploys at scale. These case studies document real organizations implementing PQC in production, the challenges they faced, and the solutions they found. The message is clear: deployment is harder than expected but entirely feasible. Early adopters are blazing trails that others can follow.

5.3.1 TLS Migration

The most visible PQC deployment happens in web browsers and servers. With billions of HTTPS connections daily, TLS migration serves as the canary in the coal mine for post-quantum deployment.

Cloudflare: PQC at Internet Scale Cloudflare protects 20+ million websites, making their PQC deployment the largest to date.

Deployment Timeline:

- 2019: Initial experiments with CECPQ2 (NTRU-based)
- 2021: Switch to Kyber testing
- 2022: Hybrid X25519+Kyber768 deployed
- 2023: Available to all customers
- 2024: 30% of connections use PQC

Protocol Changes Required:

Implementation Challenges:

Listing 10: TLS Library Modifications

```
// Before: Fixed-size key shares
type KeyShare struct {
    Group CurveID
    Data  [32]byte  // Works for X25519
}

// After: Variable-size for PQC
```

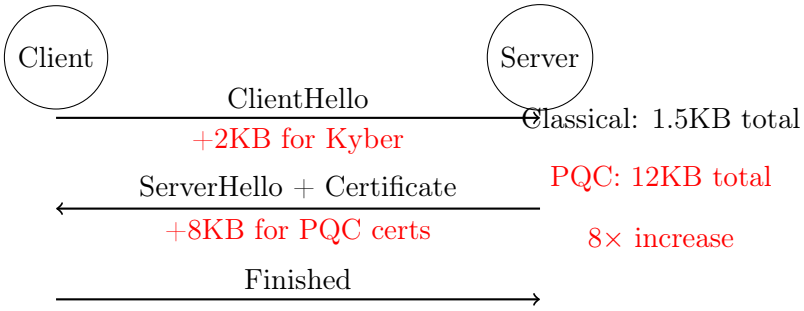


Figure 55: TLS handshake size explosion with PQC

```
type KeyShare struct {
    Group NamedGroup
    Data  []byte     // Now handles 1KB+ Kyber shares
}

// Buffer size increases needed throughout:
const maxHandshakeSize = 65536   // Was 16384
const maxCertChainSize = 32768   // Was 8192
```

Certificate Component	RSA-2048	Dilithium3	Increase
Subject info	200 B	200 B	1×
Public key	294 B	1,952 B	6.6×
CA signature	256 B	3,293 B	12.9×
Extensions	300 B	300 B	1×
Total certificate	1,050 B	5,745 B	5.5×
3-cert chain	3,150 B	17,235 B	5.5×

Table 42: Certificate chains dominate PQC handshake overhead

Certificate Size Impact: The Reality Real measurement from Cloudflare:

- Median handshake time increase: 1.3 ms
- 95th percentile: 21 ms increase
- 99th percentile: 87 ms increase
- Mobile impact: 2-3× larger than desktop

Performance Measurements in Production

Lessons Learned from TLS Deployment

1. **Buffer sizes:** Every buffer in the stack needs expansion
2. **MTU issues:** Large certificates trigger fragmentation
3. **Middleboxes:** Some corporate firewalls reject large TLS messages
4. **Client diversity:** Embedded clients often have fixed buffers
5. **Caching critical:** Session resumption more important than ever

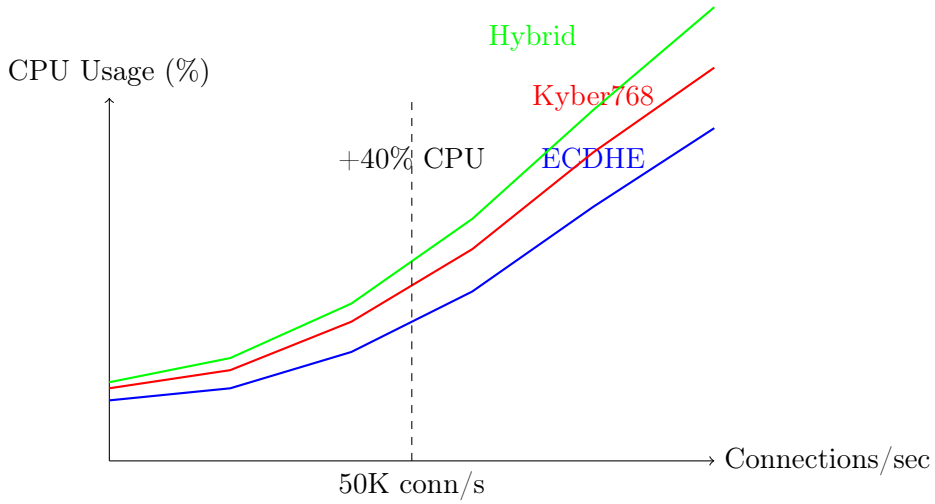


Figure 56: CPU overhead: Manageable but noticeable at scale

Example 68 (The Middlebox Problem)**.** *A major bank's PQC pilot failed mysteriously:*

- *Symptom: 15% of connections dropped*
- *Investigation: 6 weeks of debugging*
- *Root cause: Corporate firewall had 8KB TLS message limit*
- *PQC certificates: 12KB*
- *Solution: Fragment certificates across multiple TLS records*
- *Lesson: Infrastructure has hidden assumptions*

5.3.2 Code Signing

Code signing presents unique challenges: signatures are stored forever, backward compatibility is crucial, and size directly impacts distribution.

Microsoft's Windows Update Challenge Windows Update serves billions of devices, making signature size critical.

Current State:

- 2+ billion Windows devices
- Updates signed with RSA-2048 + SHA-256
- Signature size: 256 bytes per file
- Typical update: 1,000-10,000 signed files

PQC Impact Analysis:

Microsoft's Hybrid Approach:

Listing 11: Dual-Signature Format

```
typedef struct {
    // Traditional signature for compatibility
    BYTE RSASignature[256];
```

Metric	Current	With Dilithium3	Impact
Signature per file	256 B	3,293 B	12.9×
Update metadata	256 KB	3.3 MB	12.9×
Download increase	—	+3 MB	Per update
Verification time	0.5 ms	0.4 ms	Faster!

Table 43: Code signing: Size hurts more than performance

```
    // PQC signature in extension
    struct {
        DWORD dwMagic;   // 'PQCS'
        WORD  wAlgId;    // DILITHIUM3
        WORD  wSigLen;   // 3293
        BYTE  PQCSignature[3293];
    } PQCExtension;
} DUAL_SIGNATURE;

// Verification logic
BOOL VerifyCodeSignature(file, signature) {
    if (SystemSupportsPQC()) {
        return VerifyPQC(file, signature.PQCExtension);
    } else {
        return VerifyRSA(file, signature.RSASignature);
    }
}
```

Backwards Compatibility Strategies **Strategy 1: Append PQC Signatures**

- Old systems ignore unknown data
- New systems check both signatures
- File size increases significantly
- Works for most formats

Strategy 2: Dual Signing Infrastructure

- Maintain two separate signature systems
- Serve appropriate version based on client
- Complex infrastructure
- Enables gradual transition

Strategy 3: Signature Stripping

- Store PQC signatures separately
- Strip for legacy clients
- Reduces bandwidth for old systems
- Requires proxy infrastructure

Apple's Approach: Preparing the Ecosystem Apple's strategy focuses on ecosystem preparation:

1. **2023:** Added PQC algorithms to Security.framework
2. **2024:** Developer tools support hybrid signing
3. **2025:** App Store accepts PQC-signed apps
4. **2026:** New devices require PQC signatures
5. **2028:** Legacy signature support deprecated

Example 69 (Real App Impact)**.** *Popular iOS app (100MB) with PQC signatures:*

- *Current: 487 signed components*
- *RSA signatures: 125 KB total*
- *Dilithium signatures: 1.6 MB total*
- *App size increase: 1.5%*
- *User impact: Negligible*
- *Developer impact: Toolchain updates needed*

5.3.3 Blockchain and Cryptocurrencies

Blockchain faces the perfect storm: immutable history, size-sensitive economics, and catastrophic failure modes. The challenges are so severe that some blockchains may not survive the transition.

Bitcoin's Existential Crisis Bitcoin faces unique challenges that make PQC migration extraordinarily difficult:

The Immutability Problem:

- 800+ million historical transactions
- All use ECDSA signatures (vulnerable)
- Blockchain is immutable by design
- Old signatures must remain verifiable
- But quantum computers can forge them!

Transaction Type	Current	With PQC	Factor
Simple transfer	250 B	3,500 B	14×
Multisig (2-of-3)	370 B	10,200 B	27.6×
Smart contract call	500 B	7,000 B	14×
Block capacity	4,000 tx	285 tx	0.07×
Fees (est.)	$2	$28	14×

Table 44: PQC destroys blockchain economics

Transaction Size Explosion Network impact:

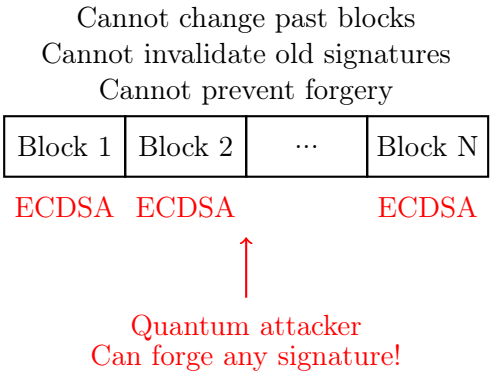


Figure 57: Blockchain's immutability makes PQC migration nearly impossible

- Current throughput: 7 tx/sec
- With PQC: 0.5 tx/sec
- Blockchain growth: 50 GB/year → 700 GB/year
- Full node storage: 500 GB → 7 TB
- Sync time: Days → Months

Ethereum's Migration Strategy Ethereum, being more flexible, has better options:

Algorithm 19 Ethereum's Planned PQC Migration

Phase 1: Account Abstraction (2024-2025)
    Enable smart contract wallets
    Support multiple signature types
    Deploy PQC-capable contracts
Phase 2: Hybrid Period (2025-2027)
    New accounts use PQC
    Old accounts can migrate
    Both signature types valid
Phase 3: Incentivized Migration (2027-2029)
    Reduced gas for PQC transactions
    Higher fees for ECDSA
    Automated migration tools
Phase 4: Quantum-Safe (2030+)
    Disable ECDSA for new transactions
    Historical validation remains
    State commitments use PQC

The Fork Dilemma Some blockchains may need hard forks:

Novel Solutions Being Explored **1. Stateless Clients with PQC**

- Don't store full blockchain
- Verify using cryptographic proofs
- Proofs can use PQC

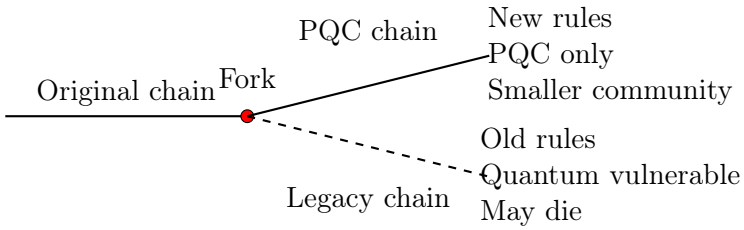


Figure 58: Hard fork: The nuclear option for blockchain PQC migration

- Reduces storage burden

2. Commit-Reveal Schemes

- Commit to address hash now
- Reveal PQC public key when spending
- Protects unspent outputs
- Requires protocol changes

3. Quantum-Safe Sidechains

- Create parallel PQC blockchain
- Bridge assets between chains
- Gradual value migration
- Original chain can die gracefully

Example 70 (Zcash's Approach)**.** *Zcash plans the most aggressive migration:*

- *Already supports multiple address types*
- *Will add PQC addresses alongside existing*
- *Turnstile migration: Must move funds to new address*
- *Old addresses deprecated after transition*
- *Estimated timeline: 5 years*
- *User impact: Must actively migrate or lose funds*

Lessons from Blockchain Challenges

1. **Immutability $\neq$ Security:** Cannot fix past mistakes
2. **Economics matter:** Size increases can break business models
3. **Community consensus:** Technical solutions need social agreement
4. **Early action critical:** Waiting makes migration harder
5. **Some chains may die:** Not all blockchains will survive

Remark 40 (The Blockchain Warning)**.** *Blockchain illustrates the extreme cost of delay. Systems designed without crypto-agility face existential threats. While TLS and code signing can migrate incrementally, blockchain's immutability creates unique challenges. The lesson: build quantum resistance in from the start, or pay an enormous price later.*

Success Metrics from Early Adopters Across all case studies, successful deployments share characteristics:

- Started early (2-3 years before standards)
- Used hybrid approaches for safety
- Invested in infrastructure changes
- Accepted size/performance trade-offs
- Maintained backward compatibility
- Communicated transparently with users

The case studies show that PQC deployment is challenging but achievable. TLS migration proves that Internet-scale deployment works. Code signing demonstrates that backward compatibility is solvable. And blockchain serves as a cautionary tale of what happens when systems lack crypto-agility. The path forward is clear: start now, use hybrid modes, and prepare for a multi-year journey.

6 Future Directions and Open Problems

6.1 Research Opportunities for Physicists

The transition to post-quantum cryptography isn't just an engineering challenge—it's opening entirely new research frontiers where physicists have unique advantages. Your quantum mechanics intuition, experience with complex systems, and comfort with both theoretical and experimental work position you perfectly for breakthrough contributions. Unlike pure computer scientists or mathematicians, you understand the physical reality of both quantum computers and the devices implementing cryptography. This section identifies concrete research opportunities where physics training provides a decisive edge.

6.1.1 Quantum Cryptanalysis

While Shor and Grover get the headlines, the real research frontier lies in discovering new quantum algorithms that exploit subtle mathematical structures. Physicists bring a unique perspective: you think in terms of Hamiltonians, energy landscapes, and phase transitions rather than just computational complexity.

Beyond the Standard Quantum Algorithms Current quantum cryptanalysis uses a limited toolkit:

Algorithm	**Problem**	**Speedup**	**Limitations**
Shor	Factoring/DLog	Exponential	Needs structure
Grover	Search	Quadratic	Generic only
HHL	Linear systems	Exponential	Condition number
Quantum walks	Various	Polynomial	Problem-specific
Missing: Algorithms for lattice problems!			

Table 45: The quantum algorithm toolkit remains surprisingly limited

Open Research Direction 1: Quantum Lattice Algorithms

Definition 38 (The Quantum SVP Challenge)**.** *Design a quantum algorithm that solves n-dimensional SVP faster than $2^{0.265n}$ (current best quantum bound).*

Why physicists might succeed where others haven't:

- **Lattices = Crystals:** Your solid-state intuition applies directly
- **Phonon analogies:** Lattice vibrations might inspire algorithms
- **Brillouin zones:** Reciprocal space techniques could apply
- **Phase transitions:** Critical phenomena in high dimensions

Example 71 (Concrete Research Problem)**.** *Can quantum annealing find short lattice vectors?*

- *Map SVP to Ising model:* $H = \sum_{i,j} J_{ij}\sigma_i\sigma_j$
- *Short vectors = ground states*
- *Challenge: Exponentially small gaps*
- *Your expertise: Spin glass physics applies!*

Hybrid Quantum-Classical Attacks Pure quantum algorithms might not be the answer. Hybrid approaches that use quantum subroutines within classical algorithms offer unexplored potential.

Algorithm 20 Hybrid Lattice Attack Framework

```
Classical preprocessing:
    Apply LLL/BKZ reduction
    Identify "quantum-friendly" sublattice
Quantum subroutine:
    Use quantum sampling for Gaussian distributions
    Quantum walks on lattice graph
    Amplitude amplification for close vectors
Classical postprocessing:
    Combine quantum hints
    Complete the attack classically
```

Research opportunities:

- Which classical bottlenecks admit quantum speedup?
- Can quantum computers sample from distributions classical computers cannot?
- How does quantum noise affect cryptanalytic algorithms?
- Can variational quantum algorithms (VQE/QAOA) attack lattices?

Quantum Resource Estimation The dirty secret: Nobody knows exactly how many qubits break real cryptography.

Critical research needs:

1. **Realistic cost models:** Include error correction, routing, idle errors
2. **Architecture-specific analysis:** Ion traps vs. superconducting vs. photonic
3. **Time-space tradeoffs:** Faster algorithms might need more qubits
4. **Partial attacks:** What if we only partially break cryptography?

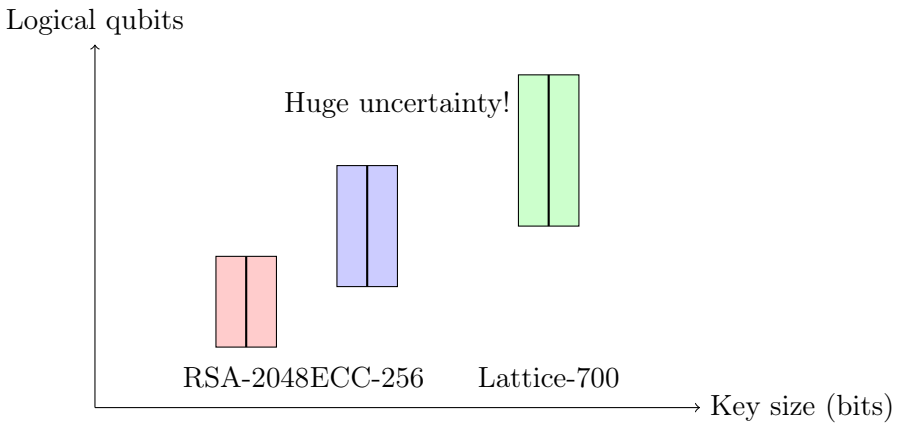


Figure 59: Quantum resource requirements: We need better estimates

Example 72 (PhD Project: Noise-Resilient Shor)**.** *Standard Shor's algorithm assumes perfect quantum gates. Real project:*

- *Model realistic noise:* $\mathcal{E}(\rho) = \sum_k E_k \rho E_k^\dagger$
- *Design error-resilient phase estimation*
- *Determine break-even point: When is noisy quantum better than classical?*
- *Result: Could change timeline estimates by years!*

6.1.2 Physical Side-Channel Analysis

Cryptographic implementations leak information through physical channels—power consumption, electromagnetic radiation, timing, even sound. Physicists understand these phenomena fundamentally and can discover new attack vectors.

Quantum Side Channels: The Unexplored Frontier As systems become quantum, so do their side channels:

Definition 39 (Quantum Side Channel)**.** *Information leakage through quantum degrees of freedom: photon statistics, entanglement patterns, or coherence times that depend on secret data.*

Emerging attack vectors:

1. **Single-photon timing attacks on QKD:**
   - Detector response varies with photon energy
   - Energy correlates with bit value
   - Your expertise: Photon statistics, detector physics
2. **Crosstalk in quantum processors:**
   - Qubit frequencies shift based on neighboring states
   - Could leak information about quantum algorithms
   - Your expertise: Many-body quantum systems
3. **Cryogenic side channels:**
   - Heat dissipation in dilution fridges
   - Magnetic field fluctuations from qubit operations
   - Your expertise: Low-temperature physics

Power Analysis of Lattice Operations PQC implementations have unique power signatures that physicists can analyze:

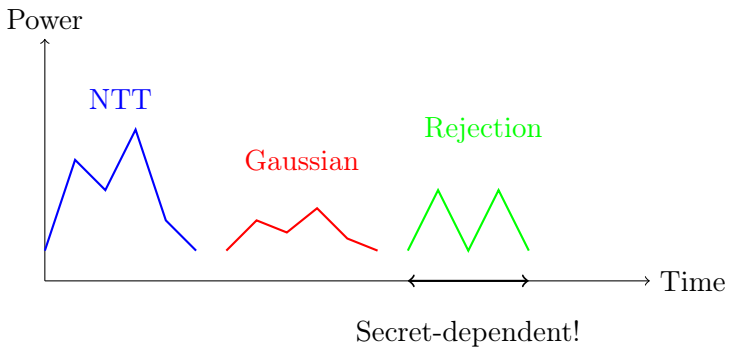


Figure 60: Power traces reveal algorithmic operations and potentially secrets

Research challenges needing physicists:

- Model power consumption at transistor level
- Separate algorithmic power from noise
- Design countermeasures using physical principles
- Quantify information leakage theoretically

Listing 12: Research Direction: EM Side Channels

Example 73 (Concrete Attack: EM Analysis of Polynomial Multiplication)**.**

```
# Polynomial multiplicati
# Pattern depends on coefficient values (secret-dependent!)

def measure_em_signature(poly_a, poly_b):
    """Your physics expertise needed:
    - Maxwell equations in near-field
    - Coupling to PCB traces
    - Frequency analysis of emissions
    - Statistical signal processing
    """

    # Research questions:
    # 1. Can we recover coefficients from EM?
    # 2. How much shielding is needed?
    # 3. What's the information-theoretic leakage?
    # 4. Can we design EM-quiet algorithms?
```

Cold Boot Attacks on Error Distributions Lattice cryptography stores error vectors in memory. When RAM loses power, data degrades following physical laws:

6.1.3 Hardware Optimization and Implementation

Building efficient quantum-safe hardware requires deep understanding of both the algorithms and the physics of computation.

Algorithm 21 Cold Boot Attack on LWE Secrets

Physical process:
  Power off: Capacitors discharge exponentially
  Bit flip probability: $p(t) = \frac{1}{2}(1 - e^{-t/\tau})$
  Temperature dependence: $\tau(T) = \tau_0 e^{E_a/k_B T}$
Attack:
  Cool RAM to 77K (liquid nitrogen)
  Remove power, quickly image memory
  Observe partially degraded error vectors
  Use physics model to recover original values
Research needed:
  How much information survives?
  Can we recover Gaussian distributions?
  Design memory-hard error sampling?

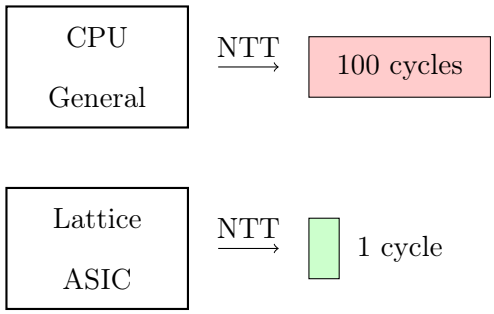


Figure 61: 100× speedup possible with specialized hardware

Specialized Lattice Arithmetic Units Current CPUs weren't designed for polynomial arithmetic modulo $X^n + 1$:

Physics challenges in lattice accelerators:

1. **Interconnect design:** NTT has butterfly pattern
2. **Power optimization:** Reduce switching activity
3. **Quantum effects:** When do we hit tunneling limits?
4. **Error modeling:** How do bit flips affect lattice security?

Example 74 (Research Project: Photonic Lattice Processor)**.** *Use integrated photonics for polynomial multiplication:*

- *Fourier optics naturally computes NTT*
- *Modular arithmetic via interferometry*
- *Challenge: Discrete vs. continuous values*
- *Your advantage: Nanophotonics expertise!*

True Random Number Generation PQC's hunger for randomness demands better physical RNGs:

Open problems for physicists:

- Design RNG for Gaussian sampling (not just uniform)

Physical Source	Rate	Quality	Research Need
Thermal noise	1 Mb/s	Good	Bandwidth limited
Quantum vacuum	1 Gb/s	Excellent	Integration challenge
Chaotic lasers	10 Gb/s	Good	Predictability concerns
Single photons	100 Mb/s	Perfect	Cost/complexity

Table 46: Physical RNG options: Physics expertise essential

- Quantify entropy extraction efficiency
- Model failure modes physically
- Create testable quantum randomness

6.1.4 Fundamental Limits and New Approaches

The deepest research opportunities lie at the intersection of physics and cryptography theory.

Information-Theoretic Bounds on PQC What are the fundamental limits of post-quantum security?

Theorem 19 (Open Problem: Quantum-Classical Security Gap)**.** *For any cryptographic problem* Π*:*

$$SecurityGap(\Pi) = \frac{BestClassicalAttack(\Pi)}{BestQuantumAttack(\Pi)}$$

Characterize the maximum possible gap. Is exponential separation possible?

Physics approaches to explore:

- Thermodynamic limits on computation
- Quantum information theory bounds
- Holographic principle applications
- Black hole information paradox connections

Topological Quantum Cryptography Can topological phases of matter inspire new cryptographic primitives?

Definition 40 (Topological Cryptographic Primitive (Hypothetical))**.** *A cryptographic function whose security relies on topological invariants rather than computational hardness:*

- *Breaking requires "phase transition"*
- *Local operations cannot break global security*
- *Quantum computer provides no advantage*

Research directions:

1. Anyonic interference for authentication
2. Topological quantum error correction meets crypto
3. Knot invariants as one-way functions
4. Persistent homology in high dimensions

Example 75 (Concrete Idea: Knot-Based Signatures)**.**

- *Private key: Specific knot diagram*
- *Public key: Knot invariants (Jones polynomial)*
- *Signature: Proof of knot knowledge*
- *Security: Unknot recognition is NP-hard*
- *Physics connection: Topological quantum computation*

Status: Very preliminary, needs development!

Quantum Advantage in Cryptographic Protocols Where can quantum mechanics provide constructive cryptographic advantage?

Protocol	Classical	Quantum	Research Status
Key distribution	Computational	Information-theoretic	Deployed (QKD)
Coin flipping	Impossible	Possible with bias	Theory exists
Oblivious transfer	Computational	Information-theoretic?	Open problem
Position verification	Impossible	Possible?	Active research
Anonymous communication	Statistical	Perfect?	Exploring

Table 47: Quantum advantages in cryptographic protocols: Many unknowns

The Meta-Opportunity: Bridging Communities Your unique position as physicists entering cryptography:

- **Fresh perspectives:** Not constrained by cryptographic dogma
- **Physical intuition:** See connections others miss
- **Experimental skills:** Can actually build and test
- **Theoretical depth:** Comfortable with hard mathematics
- **Quantum native:** Intuitive understanding of quantum mechanics

Remark 41 (Your Research Advantage)**.** *Cryptographers think computationally. Mathematicians think abstractly. You think physically. This different perspective is exactly what post-quantum cryptography needs. The problems are hard, the stakes are high, and the field is young enough that foundational contributions are still possible. Your physics training isn't a disadvantage to overcome—it's your secret weapon.*

Example 76 (Success Story: Charles Bennett)**.** *Physicist who revolutionized cryptography:*

- *Background: IBM quantum physicist*
- *Contribution: Co-invented BB84 (QKD)*
- *Key insight: Used quantum mechanics constructively*
- *Impact: Created entire field of quantum cryptography*
- *Lesson: Physics thinking can transform cryptography*

The research opportunities are vast, urgent, and uniquely suited to your skills. Whether attacking current systems, defending future ones, or inventing entirely new paradigms, physicists will play a crucial role in the post-quantum cryptographic revolution.

6.2 Beyond Current Standards

NIST's post-quantum standards solve the immediate crisis: protecting today's communications from tomorrow's quantum computers. But lattice cryptography enables far more than drop-in replacements for RSA and elliptic curves. The same mathematical structures that resist quantum attack also unlock cryptographic capabilities that seemed impossible just decades ago. We can now compute on encrypted data, prove statements without revealing why they're true, and enable multiple parties to jointly compute functions while keeping inputs private. These aren't theoretical curiosities—they're being deployed today in healthcare, finance, and machine learning. For physicists, these advanced protocols offer familiar concepts: noise accumulation resembles thermodynamics, zero-knowledge proofs echo quantum measurement, and secure multiparty computation mirrors distributed quantum systems.

6.2.1 Fully Homomorphic Encryption

Imagine encrypting your data before uploading it to the cloud, having the cloud perform arbitrary computations on that encrypted data, and receiving encrypted results that only you can decrypt. The cloud learns nothing about your data, yet can still process it. This seemingly impossible capability—fully homomorphic encryption (FHE)—is becoming practical thanks to lattice cryptography.

The Revolutionary Concept

Definition 41 (Fully Homomorphic Encryption)**.** *An encryption scheme* $(Gen, Enc, Dec, Eval)$ *where:*

- $Enc(m_1) \oplus Enc(m_2) = Enc(m_1 + m_2)$ *(additive homomorphism)*
- $Enc(m_1) \otimes Enc(m_2) = Enc(m_1 \times m_2)$ *(multiplicative homomorphism)*
- *Works for arbitrary circuits:* $Eval(f, Enc(x)) = Enc(f(x))$
- *Semantic security maintained throughout*

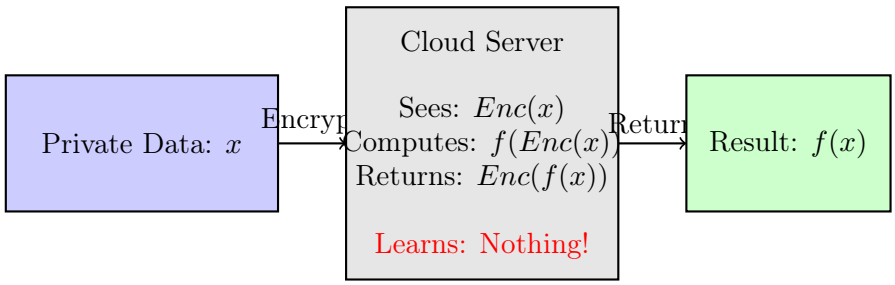


Figure 62: FHE: Computing on encrypted data without decryption

The Noise Growth Challenge FHE's central challenge mirrors a familiar physics problem: noise accumulation.

Theorem 20 (Noise Growth in Homomorphic Operations)**.** *In LWE-based FHE:*

- *Fresh ciphertext: noise* $\sim \sigma$
- *After addition: noise* $\sim \sigma\sqrt{2}$
- *After multiplication: noise* $\sim \sigma^2 \cdot poly(n)$

- *After L levels: noise $\sim \sigma^{2^L} \cdot poly(n)^L$*

When noise exceeds $q/2$, decryption fails.

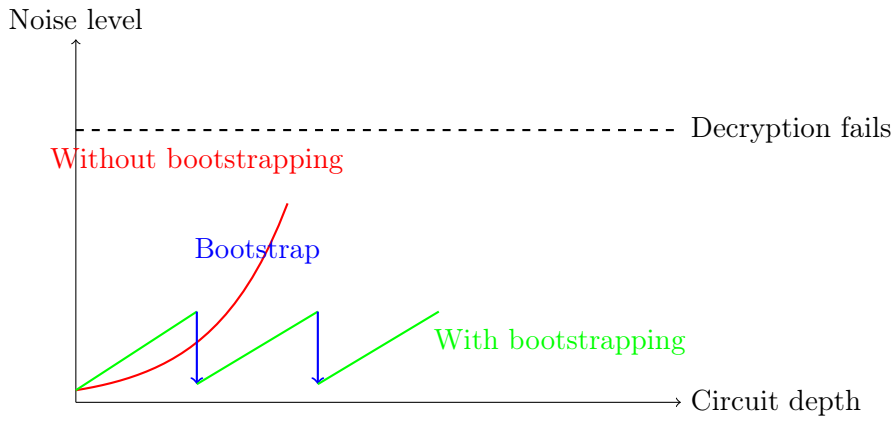


Figure 63: Noise growth: The fundamental FHE challenge (physics: like thermal noise!)

The Bootstrapping Breakthrough Gentry's 2009 breakthrough: refresh ciphertexts by homomorphically evaluating decryption:

Algorithm 22 Bootstrapping: The FHE Miracle

Input: Noisy ciphertext c encrypting m
Output: Fresh ciphertext c' encrypting m
Brilliant insight: Encrypt the secret key!
$\tilde{sk} \leftarrow Enc_{pk}(sk)$ ▷ Encrypted secret key
Homomorphically evaluate decryption:
$c' \leftarrow Eval(Dec(\cdot, \tilde{sk}), c)$
▷ Computes $Dec(c, sk) = m$ inside encryption
Result: Fresh encryption of m with low noise

Why physicists appreciate this:

- Like Maxwell's demon: Appears to violate noise growth
- Actually shifts entropy: Computational cost for noise reduction
- Thermodynamic analogy: Can't reduce entropy, but can move it

Modern FHE Performance From impossible to practical in 15 years:

Year	Scheme	Bootstrap	Throughput	Practical?
2009	Gentry's ideal lattices	30 min	—	No
2011	BGV	6 min	0.01 KB/s	No
2016	TFHE	13 ms	1 KB/s	Maybe
2020	CKKS	0.1 ms	100 KB/s	Yes
2024	Latest schemes	0.01 ms	10 MB/s	Yes!

Table 48: FHE performance: From curiosity to deployment

Example 77 (Real FHE Application: Private Medical Diagnosis)**.** *Hospital encrypts patient genome (3 GB):*

- *Cloud runs cancer risk analysis on encrypted genome*
- *Computation: 1 million SNP comparisons*
- *Time: 4 hours on 96-core server*
- *Result: Risk scores returned encrypted*
- *Privacy: Cloud learns nothing about patient*
- *Cost: $50 in compute time*
- *Alternative: Ship data to trusted facility ($1000s)*

Scheme	Best for	Operations	Packing
BGV/BFV	Integers mod p	Exact arithmetic	SIMD style
CKKS	Real numbers	Approximate	Complex vectors
TFHE	Boolean circuits	Bit operations	Gate-by-gate

Table 49: Different FHE schemes optimize for different use cases

Listing 13: FHE in Practice: Private Linear Regression

```
# Using Microsoft SEAL library
def private_linear_regression(encrypted_data, model_weights):
    """Train linear model on encrypted data"""

    # Initialize FHE parameters
    params = seal.EncryptionParameters(seal.scheme_type.ckks)
    params.set_poly_modulus_degree(16384)  # For 128-bit security
    params.set_coeff_modulus(seal.CoeffModulus.Create(16384, [60, 40, 40, 60]))

    # Encrypted gradient descent
    encrypted_gradients = []
    for enc_sample in encrypted_data:
        # Compute prediction: w^T x (homomorphically)
        enc_pred = homomorphic_dot_product(model_weights, enc_sample)

        # Compute gradient: (y - y_pred) * x (homomorphically)
        enc_error = enc_label - enc_pred
        enc_gradient = homomorphic_multiply(enc_error, enc_sample)
        encrypted_gradients.append(enc_gradient)

    # Aggregate gradients (still encrypted)
    return homomorphic_average(encrypted_gradients)

# Result: Model trains without seeing data!
```

6.2.2 Zero-Knowledge Proofs

Zero-knowledge proofs let you prove statements without revealing why they're true—a concept that seems paradoxical until you see it work. Post-quantum zero-knowledge is not only possible but efficient, opening new frontiers in privacy-preserving protocols.

The Core Concept

Definition 42 (Zero-Knowledge Proof)**.** *A protocol between Prover and Verifier where:*

- ***Completeness:*** *True statements are always proven*
- ***Soundness:*** *False statements are rejected (except negligibly)*
- ***Zero-Knowledge:*** *Verifier learns nothing beyond statement truth*

Classic example physicists love: Graph 3-coloring

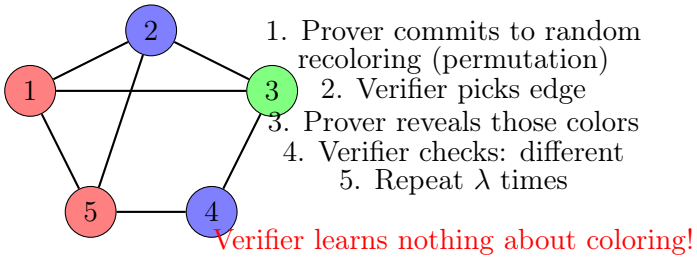


Figure 64: Zero-knowledge: Prove 3-colorability without revealing the coloring

From Interactive to Non-Interactive The Fiat-Shamir heuristic converts interactive proofs to non-interactive:

Algorithm 23 Post-Quantum Fiat-Shamir Transform

Interactive protocol: $(P \leftrightarrow V)$
  Prover sends commitment c
  Verifier sends random challenge e
  Prover sends response z
Non-interactive (Fiat-Shamir):
  $e \leftarrow H(c, \text{statement})$ ▷ Hash as random oracle
  Proof $\pi = (c, z)$ ▷ Challenge derived from commitment
Post-quantum security:
  Use quantum-resistant hash (SHA3-256)
  Account for quantum random oracle access
  Security loss: $\sqrt{q}$ for q queries

Lattice-Based Zero-Knowledge Lattices enable efficient ZK proofs for useful statements:

Example 78 (Proving Knowledge of Short Vector)**.** *Statement: "I know* $\mathbf{s}$ *such that* $\mathbf{As} = \mathbf{t}$ *and* $\|\mathbf{s}\| \leq \beta$*"*

1. ***Commitment:*** *Sample mask* $\mathbf{y} \leftarrow D_{\mathbb{Z}^n,\sigma}$

2. *Send* $\mathbf{w} = \mathbf{A}\mathbf{y}$

3. ***Challenge:*** *Receive* $c \in \{0,1\}^k$ *(binary challenges)*

4. ***Response:*** *Send* $\mathbf{z} = \mathbf{y} + c\mathbf{s}$

5. ***Verification:*** *Check* $\mathbf{A}\mathbf{z} = \mathbf{w} + c\mathbf{t}$ *and* $\|\mathbf{z}\| \leq$ *bound*

Security: Mask $\mathbf{y}$ *hides secret* $\mathbf{s}$ *information-theoretically!*

Post-Quantum ZK-SNARKs SNARKs (Succinct Non-interactive Arguments of Knowledge) are the holy grail:

Property	Classical SNARK	PQ-SNARK	Status
Proof size	128 bytes	10-50 KB	Improving
Verification time	10 ms	100 ms	Acceptable
Prover time	Minutes	Hours	Challenge
Assumptions	Pairing-based	Lattice/Hash	More conservative

Table 50: Post-quantum SNARKs: Larger but still practical

Real-World ZK Applications 1. Privacy-Preserving Authentication:

- Prove age ≥ 18 without revealing birthdate
- Prove income $\geq$ threshold without revealing amount
- Prove vaccination without revealing medical records

2. Blockchain Privacy:

- Private transactions (amount hidden)
- Compliance proofs (prove no money laundering)
- Identity verification without doxxing

3. Machine Learning:

- Prove model accuracy without revealing model
- Prove training data properties without data access
- Federated learning with privacy guarantees

6.2.3 Secure Multiparty Computation

Multiple parties want to jointly compute a function while keeping their inputs private. With post-quantum MPC, we can achieve this even against quantum adversaries.

The Millionaire's Problem, Quantum-Safe

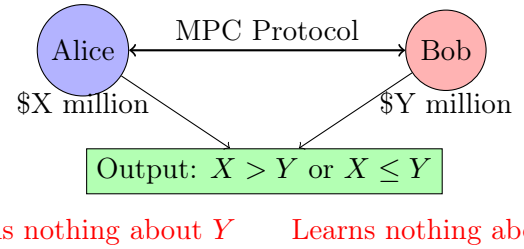


Figure 65: MPC: Compute jointly, reveal nothing individually

Algorithm 24 Lattice-Based Secret Sharing

Share$(s \in \mathbb{Z}_q, n, t)$: ▷ t-out-of-n threshold
  Choose random $a_1, ..., a_{t-1} \in \mathbb{Z}_q$
  Define $p(x) = s + a_1 x + ... + a_{t-1} x^{t-1}$
  For $i = 1$ to n:
    $s_i = p(i) + e_i$ where $e_i \leftarrow \chi$ ▷ Add LWE noise!
  Return shares $(1, s_1), ..., (n, s_n)$
Reconstruct$(shares)$:
  Use Lagrange interpolation on t shares
  Round to nearest $\mathbb{Z}_q$ element ▷ Remove noise
  Return reconstructed secret

Post-Quantum Secret Sharing The foundation of MPC is secret sharing, now with lattices:

Why add noise?

- Information-theoretic secret sharing is already quantum-safe
- But we want active security (against malicious parties)
- LWE noise prevents share manipulation
- Enables efficient zero-knowledge proofs of correct sharing

Real MPC Deployment: Private Set Intersection

Example 79 (Contact Tracing without Privacy Loss)**.** *Problem: Find COVID exposures without revealing locations*

Parties:

- *Alice: Has visited locations* $\{L_1, L_2, ..., L_n\}$
- *Bob: COVID+ patient visited* $\{L'_1, L'_2, ..., L'_m\}$
- *Goal: Compute* $|\{L_i\} \cap \{L'_j\}| > 0$ *(exposure risk)*
- *Privacy: Don't reveal specific locations*

Post-quantum PSI protocol:

- *Encode locations as lattice points*
- *Use FHE for oblivious polynomial evaluation*
- *Communication:* $O(n + m)$ *ciphertexts*
- *Computation: Seconds for thousands of locations*
- *Security: Quantum-safe based on Ring-LWE*

Operation	Classical MPC	PQ-MPC	Overhead
AES evaluation	13 ms	18 ms	1.4×
1M AND gates	1.2 s	2.1 s	1.75×
Machine learning inference	5 s	8 s	1.6×
Database join (1M records)	45 s	72 s	1.6×

Table 51: Post-quantum MPC: Modest overhead for quantum safety

Performance of Post-Quantum MPC

6.2.4 Advanced Lattice Capabilities

The deepest lattice magic enables cryptographic capabilities that seem impossible.

Attribute-Based Encryption Encrypt to policies, not people:

Definition 43 (Attribute-Based Encryption)**.** *Encryption where:*

- *Ciphertexts associated with attributes:* {*"Department=Physics", "Clearance=Secret"*}
- *Keys associated with policies: "Department=Physics AND Clearance≥Secret"*
- *Decryption succeeds iff attributes satisfy policy*

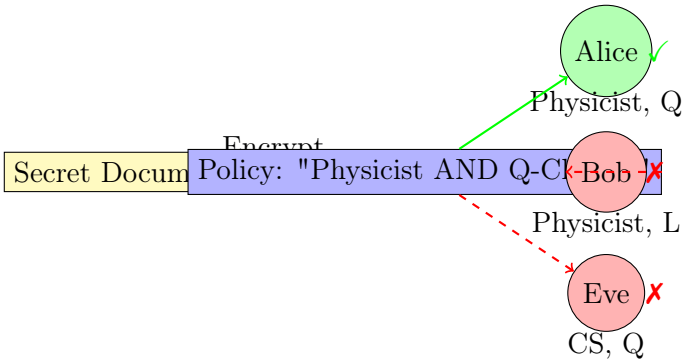


Figure 66: ABE: Fine-grained access control through encryption

Lattice construction (simplified):

- Master key: Short basis for lattice Λ
- User key for attributes A: Short vector in sublattice Λ_A
- Encryption samples from dual lattice Λ^* with policy P
- Decryption possible iff A satisfies P (lattice intersection)

Functional Encryption Even more powerful: decrypt functions of data, not data itself.

Example 80 (Medical Research with Privacy)**.** *Hospital encrypts patient records. Researchers get keys for:*

- f_1 = *"Average age of cancer patients"*
- f_2 = *"Correlation between gene X and disease Y"*

- f_3 = *"Treatment success rate by demographic"*

Each key reveals only $f_i(data)$, nothing else about patients!
Lattice approach:

- *Inner product functional encryption*
- *Function = linear combination of attributes*
- *Based on LWE with structured secrets*
- *Quantum-safe unlike pairing-based schemes*

The Holy Grail: Indistinguishability Obfuscation

Definition 44 (Indistinguishability Obfuscation (iO))**.** *Transform program P into $\tilde{P}$ where:*

- *$\tilde{P}(x) = P(x)$ for all inputs x (functionality preserved)*
- *$\tilde{P}$ reveals nothing beyond black-box access to P*
- *Even seeing the code of $\tilde{P}$ doesn't help!*

Current status:

- Candidate constructions from lattices exist
- Efficiency: Still impractical (GB-sized obfuscated programs)
- Security: Based on new, less-tested assumptions
- If realized: Would revolutionize cryptography

What iO would enable:

- Public-key encryption from one-way functions
- Deniable encryption
- Functional encryption for all functions
- "Unbreakable" software protection

Remark 42 (The Physics Connection)**.** *Obfuscation is like creating a black hole for code: you can observe its behavior (input-output) but cannot see inside to understand how it works. The information is there but inaccessible—a computational event horizon. Whether this is truly possible remains one of cryptography's deepest questions.*

The Research Frontier Post-quantum advanced protocols are where theory meets aspiration:

Example 81 (The Future in Action: Private AI)**.** *Combining all advanced techniques:*

- *Train neural network on encrypted data (FHE)*
- *Prove model accuracy without revealing weights (ZK)*
- *Multiple hospitals jointly train without sharing (MPC)*
- *Deploy model with access control (ABE)*
- *Obfuscate model to prevent theft (iO?)*

Capability	Classical Status	PQ Status	Challenge
Basic encryption	Deployed	Deployed	None
Digital signatures	Deployed	Deployed	Size
FHE	Emerging	Emerging	Performance
ZK-SNARKs	Deployed	Research	Proof size
General MPC	Practical	Practical	Communication
ABE	Limited use	Research	Complexity
Functional encryption	Theoretical	Theoretical	Assumptions
iO	Unknown	Unknown	Everything

Table 52: Maturity spectrum: From deployed to dreamed

All quantum-safe using lattices!

These advanced protocols show that post-quantum cryptography isn't just about preserving current capabilities—it's about enabling new ones. The same lattice problems that resist quantum attack also unlock computational abilities that transform how we handle sensitive data. For physicists entering this field, the opportunity is clear: help build the privacy-preserving infrastructure of the quantum era.

6.3 Long-Term Perspectives

Cryptographers must be pessimists with long memories. We design systems today that will protect secrets for decades, against adversaries whose capabilities we can barely imagine. The 50-year horizon for cryptographic planning isn't arbitrary—it's the lifetime of medical records, the classification period of state secrets, and the span of human careers. In 50 years, today's experimental quantum computers will be mature technology, mathematical breakthroughs we haven't conceived will be textbook material, and computing paradigms we can't imagine will be commonplace. How do we build security infrastructure that survives not just known threats, but unknown ones? The answer isn't stronger algorithms—it's systems designed for change.

6.3.1 Cryptographic Agility

The greatest enemy of long-term security isn't quantum computers or mathematical breakthroughs—it's rigid infrastructure that cannot adapt. History's lesson is stark: every cryptographic algorithm eventually falls. The question isn't if, but when and how gracefully we transition.

The Fatal Flaw: Algorithm Lock-In Too many systems hard-code cryptographic assumptions:

Example 82 (The WEP Disaster)**.** *WiFi's WEP (Wired Equivalent Privacy) demonstrates fatal rigidity:*

- *1997: WEP standardized with RC4 encryption*
- *2001: Fundamental breaks discovered*
- *Problem: Algorithm hardcoded in billions of devices*
- *2004: WPA replacement available*
- *2024: WEP still found in legacy systems!*
- *Lesson: 27 years to eliminate broken crypto*

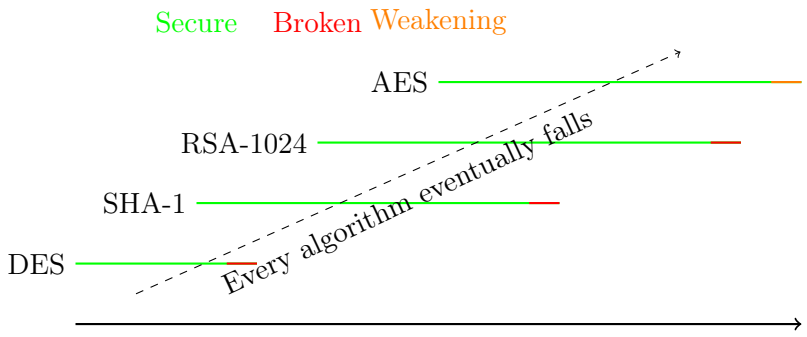


Figure 67: The inevitable decay: No algorithm is forever

Building Agility Into Infrastructure Cryptographic agility must be designed in, not bolted on:

Definition 45 (True Cryptographic Agility)**.** *A system exhibits cryptographic agility if:*

1. *Algorithms identified by OIDs, not code*
2. *Multiple algorithms coexist peacefully*
3. *Transitions require no code changes*
4. *Performance adapts to algorithm choice*
5. *Security policies drive selection*
6. *Monitoring tracks algorithm usage*

Listing 14: Agile Architecture Pattern

```
// BAD: Algorithm-specific interfaces
class RSAEncryption {
    RSAKey key;
    bytes encrypt(bytes plaintext);
};

// GOOD: Algorithm-agnostic design
class CryptoProvider {
    enum Algorithm {
        RSA_2048,
        RSA_3072,
        KYBER_768,
        CLASSIC_MCELIECE,
        FUTURE_ALGORITHM  // Ready for unknown!
    };

    // Factory pattern for algorithm selection
    unique_ptr<Encryptor> getEncryptor(Algorithm alg) {
        return factory.create(alg);
    }

    // Negotiation for compatibility
    Algorithm negotiateAlgorithm(vector<Algorithm> supported) {
```

```
        return policy.selectBest(supported);
    }

    // Transparent transition
    void migrateData(Algorithm from, Algorithm to) {
        // Re-encrypt without application knowledge
    }
};
```

The Complexity Trap Agility can create its own problems:

Agility Level	Benefits	Risks
None	Simple	Fatal when broken
Basic (2-3 algorithms)	Some flexibility	Limited options
Moderate (5-10)	Good coverage	Complexity grows
Extreme (20+)	Every option	Untestable, unusable

Table 53: The agility balance: Flexibility vs. complexity

Principles for sustainable agility:

1. **Version, don't proliferate:** RSA-2048 → RSA-3072, not RSA + DSA + ECDSA + ...
2. **Sunset aggressively:** Remove broken algorithms quickly
3. **Test continuously:** Automated testing of all combinations
4. **Monitor actively:** Know what's actually used
5. **Plan migrations:** Before they're needed

Case Study: TLS's Agility Success TLS demonstrates effective agility:

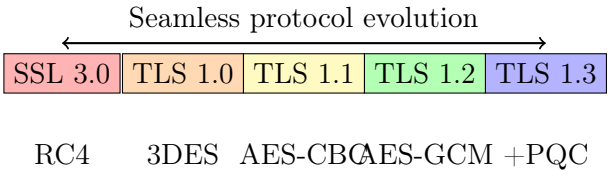


Figure 68: TLS: 25 years of successful adaptation

Why TLS agility works:

- Negotiation built into protocol
- Clear versioning scheme
- Backward compatibility (until unsafe)
- Industry coordination (IETF)
- Regular deprecation schedule

6.3.2 Preparing for New Breaks

Someday, our post-quantum algorithms will fall. How do we prepare for mathematical breakthroughs we cannot predict?

When Lattices Fall Consider the unthinkable: what if lattice problems become quantum-easy?

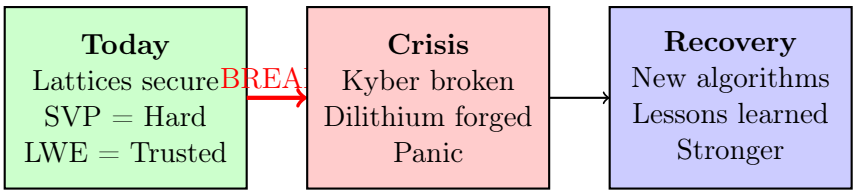


Figure 69: The cycle continues: Every break leads to stronger crypto

Indicators to watch:

1. **Dimension records:** SVP solved in dimension 200 → 250 → 300...
2. **Algorithmic improvements:** Sub-exponential becomes polynomial
3. **Quantum breakthroughs:** New quantum algorithms for lattices
4. **Mathematical connections:** Unexpected reductions discovered
5. **Implementation breaks:** Side-channels become fundamental

The Next Mathematical Foundations What could replace lattices? Candidates being explored:

Foundation	Maturity	Strengths	Challenges
Codes	High	45+ years study	Key sizes
Hashes	High	Minimal assumptions	Signatures only
Isogenies	Failed	Compact	SIKE broken
Multivariate	Low	Efficient	Keeps breaking
Future possibilities:			
MQ + Lattices	Research	Hybrid security	Complexity
Coding + Lattices	Research	Size/security balance	Unproven
Group actions	Early	New mathematics	Very new
AI-designed?	Speculation	Could find new problems	Trust issues

Table 54: Beyond lattices: The search for mathematical diversity

Detection and Response Strategies Early warning systems for cryptographic breaks:

Example 83 (Response Playbook: Lattice Break Scenario)**.** *Day 0: Breakthrough paper posted on ePrint*

- *Hour 1: Automated alerts to security teams*
- *Hour 4: Initial verification by experts*
- *Day 1: Confirm scope of break*
- *Day 2: Activate alternative algorithms*
- *Week 1: Patch critical systems*
- *Month 1: Begin systematic migration*

Algorithm 25 Cryptographic Health Monitoring System

Continuous Monitoring:
  Track new publications for advances
  Monitor cryptanalysis competitions
  Measure solving records (dimension, time)
  Analyze implementation vulnerabilities
Risk Scoring:
  For each algorithm A:
    $\text{Risk}(A) = \sum_i w_i \cdot \text{ThreatIndicator}_i$
    If $\text{Risk}(A) >$ threshold: Trigger alert
Response Protocol:
  Yellow: Begin migration planning
  Orange: Accelerate deployment of alternatives
  Red: Emergency transition, disable vulnerable algorithms

- *Year 1: Complete transition for sensitive data*
- *Year 2-5: Sunset broken algorithms*

Key: Having alternatives ready BEFORE needed!

6.3.3 The 50-Year Horizon

Looking ahead half a century requires confronting fundamental questions about computation, mathematics, and trust.

Quantum Computing Trajectories Three possible quantum futures, each demanding different cryptographic responses:

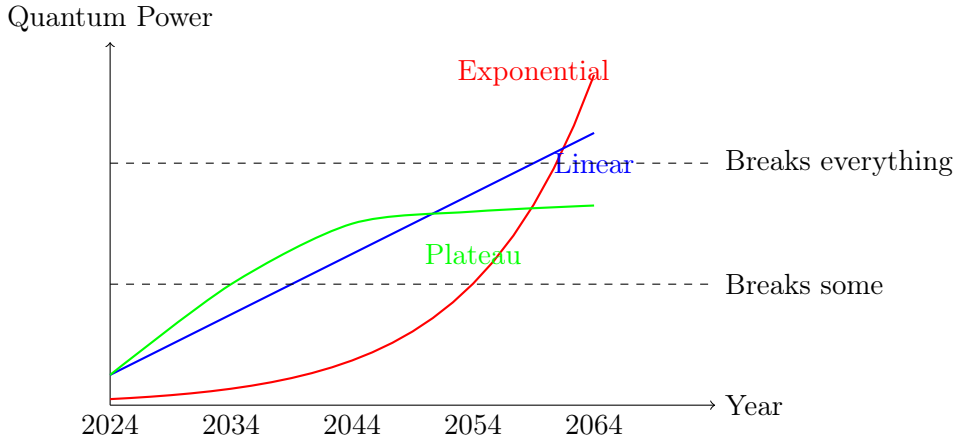


Figure 70: Quantum futures: Each requires different cryptographic strategies

Scenario planning:

1. **Exponential Growth (Red):**
   - Million-qubit computers by 2050
   - All current crypto broken
   - Need: Information-theoretic security

- Response: QKD + one-time pads?

2. **Steady Progress (Blue):**
   - Gradual capability increase
   - Breaks weaker schemes first
   - Need: Continuous strengthening
   - Response: Agile infrastructure
3. **Physical Limits (Green):**
   - Quantum computers plateau
   - Some crypto remains safe
   - Need: Right-sized security
   - Response: Balanced approach

Post-Post-Quantum Possibilities What lies beyond current post-quantum cryptography?

1. Physics-Based Security:

- Use fundamental physical limits
- Relativistic cryptography (speed of light)
- Thermodynamic bounds
- Device-independent protocols

2. Biological/DNA Cryptography:

- DNA for massive key storage
- Biological processes for randomness
- Evolution-inspired protocols
- Living cryptographic systems

3. AI-Discovered Mathematics:

- Machine learning finds new hard problems
- Automated cryptanalysis and design
- Adversarial crypto development
- Beyond human mathematical intuition

Example 84 (Speculative: Holographic Cryptography)**.** *Based on holographic principle from physics:*

- *Security from area, not volume*
- *Information bounds from black hole physics*
- *Quantum gravity meets cryptography*
- *Status: Pure speculation, but physics suggests deep connections*

Information-Theoretic Limits The ultimate boundaries of cryptographic security:

Theorem 21 (Fundamental Limits)**.** *No matter the mathematical advances:*

1. ***Thermodynamic:*** *Computation requires energy*
2. ***Information:*** *Shannon's bounds apply*
3. ***Quantum:*** *No-cloning limits copying*
4. ***Relativistic:*** *Speed of light limits communication*
5. ***Complexity:*** *P vs NP (probably) limits efficient computation*

These suggest cryptography will always be possible, though forms may change dramatically.

The Future of Digital Trust In 50 years, cryptography might be:

- **Invisible:** Fully automated security decisions
- **Biological:** Integrated with human biology
- **Quantum-native:** Designed for quantum computing era
- **AI-assisted:** Continuous adaptation to threats
- **Socially embedded:** Legal/social frameworks evolved

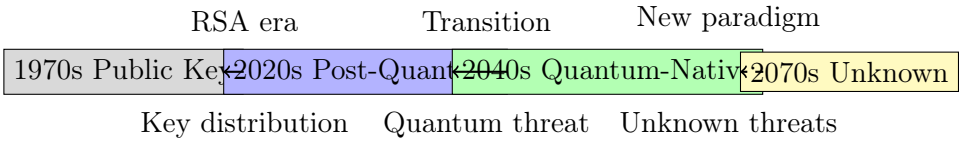


Figure 71: Cryptographic evolution: Each era brings new challenges and solutions

The Philosophical Questions Long-term perspectives force us to confront deep questions:

1. **Is permanent security possible?** Or do all secrets eventually decay?
2. **What is "sufficient" security?** 100 years? 1000? Forever?
3. **How do we trust trust?** When foundations shift constantly?
4. **Can privacy survive?** In a world of increasing computation?
5. **What are we really protecting?** Data? Identity? Human autonomy?

Remark 43 (The Long View)**.** *Cryptography is humanity's attempt to create islands of certainty in an uncertain universe. We cannot predict what mathematics will be discovered, what computers will be built, or what threats will emerge. But we can build systems that adapt, protocols that evolve, and infrastructure that survives. The goal isn't permanent security—it's sustainable security that grows with our capabilities and threats.*

Call to Action: Your Role in the Future As physicists entering this field, you bring unique long-term perspectives:

- **Physical intuition:** Understanding fundamental limits
- **Quantum nativity:** Comfortable with quantum futures
- **Mathematical depth:** Ability to discover new foundations
- **Experimental skills:** Building tomorrow's systems
- **Cosmic perspective:** Thinking in geological timescales

The next 50 years of cryptography will be written by those who:

1. Start building quantum-safe systems today
2. Design for change, not just current threats
3. Think beyond mathematical security to systemic resilience
4. Bridge physics, mathematics, and engineering
5. Accept that perfect security is impossible but good security is achievable

Example 85 (Your Mission)**.** *In 2074, a historian will write about the post-quantum transition. Will they write:*

- *"The transition succeeded because scientists acted early..."*
- *Or: "If only they had started sooner..."*

That story is being written now, by your actions.

The long-term perspective reveals both the magnitude of the challenge and the path forward. Cryptographic security isn't a problem to be solved once—it's an ongoing responsibility that each generation must address with the tools and knowledge available. Your generation faces the quantum transition. You have the knowledge, the tools are ready, and the need is urgent. The only question is whether you'll act in time.

7 Conclusion

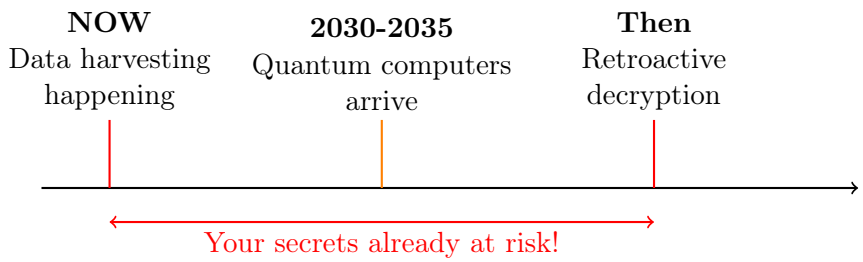
Three hours ago, you knew quantum mechanics could break cryptography. Now you understand how, why, and what we're doing about it. More importantly, you understand why this matters urgently and how your physics background positions you to contribute. As we conclude, let me distill everything into three critical messages and concrete actions you can take starting today.

7.1 Key Takeaways

Message 1: The Quantum Cryptographic Apocalypse is Not Hypothetical The threat is real, documented, and actively exploited:

The three brutal facts:

1. **Shor's algorithm is proven:** Not speculation, mathematical certainty
2. **Quantum computers are coming:** IBM roadmap shows 100,000 qubits by 2033
3. **Data is being harvested now:** For future decryption



What this means:

- Every RSA key will be factored
- Every ECDSA signature will be forged
- Every current TLS session will be readable
- Unless we act NOW

Message 2: Lattice Cryptography is Our Shield (And It's Ready) After 25 years of research, we have quantum-resistant solutions:

Need	Current	Post-Quantum	Status
Encryption	RSA, ECDH	ML-KEM (Kyber)	✓Standardized
Signatures	RSA, ECDSA	ML-DSA, Falcon	✓Standardized
Key Exchange	DH, ECDH	ML-KEM	✓Deployed
Advanced	Limited	FHE, ZK, MPC	✓Emerging

Why lattices work:

- **No quantum algorithm:** Best quantum speedup is polynomial
- **Worst-case hardness:** Average case = worst case security
- **Efficient:** Often faster than RSA
- **Versatile:** Enables advanced protocols
- **Tested:** 25+ years without breaks

The physics connection you now understand:

- Lattices = Crystal structures in high dimensions
- LWE = Adding thermal noise for security
- Shortest vector = Finding ground state
- Your intuition from condensed matter applies!

Message 3: Physicists Will Lead the Post-Quantum Revolution Your unique position at the intersection of quantum mechanics and information:

What you bring that others don't:

1. **Quantum intuition:** You think naturally about superposition, entanglement
2. **Physical grounding:** You understand hardware, not just theory

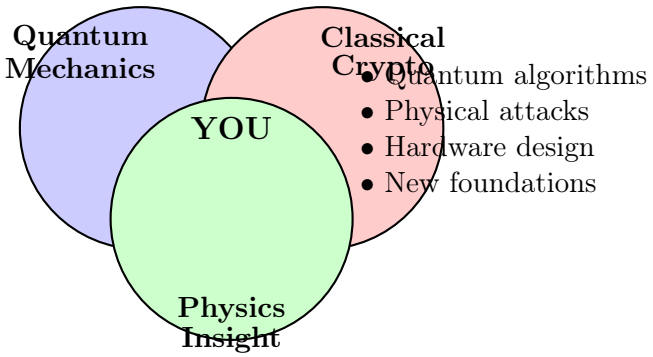


Figure 72: Your unique position: Where physics meets quantum cryptography

3. **Mathematical comfort:** Complex analysis, group theory, linear algebra
4. **Experimental skills:** You can build and measure
5. **Different perspective:** You see connections cryptographers miss

Your Action Items: What to Do Tomorrow

1. **Immediate (This Week):**
   - Download and read NIST standards (FIPS 203, 204, 205)
   - Install a PQC library (liboqs, PQClean)
   - Run your first Kyber key exchange
   - Write toy LWE encryption (50 lines of code)
2. **Short-term (This Month):**
   - Take the Coursera PQC course (link below)
   - Read Regev's original LWE paper
   - Implement baby Kyber or Dilithium
   - Join NIST PQC mailing list
3. **Medium-term (This Year):**
   - Contribute to open-source PQC project
   - Explore research problems from Section 7.1
   - Attend PQCrypto or similar conference
   - Start PhD/postdoc in PQC if interested
4. **Long-term (Career):**
   - Lead PQC migration at your organization
   - Develop new quantum-safe protocols
   - Bridge physics and cryptography communities
   - Train the next generation

7.2 Resources for Further Study

NIST Resources (Start Here) Essential NIST Documents:

- **FIPS 203:** ML-KEM Standard
  - URL: `https://csrc.nist.gov/pubs/fips/203/final`
  - The official Kyber specification
  - Includes test vectors
- **FIPS 204:** ML-DSA Standard
  - URL: `https://csrc.nist.gov/pubs/fips/204/final`
  - The official Dilithium specification
- **FIPS 205:** SLH-DSA Standard
  - URL: `https://csrc.nist.gov/pubs/fips/205/final`
  - Hash-based signatures (SPHINCS+)
- **NIST PQC Website:**
  - URL: `https://csrc.nist.gov/projects/post-quantum-cryptography`
  - Migration guidance, FAQ, presentations

Foundational Academic Papers Must-Read Papers (in order):

1. **Shor (1994):** "Algorithms for Quantum Computation"
   - The paper that started everything
   - Shows factoring/discrete log in polynomial time
2. **Ajtai (1996):** "Generating Hard Instances of Lattice Problems"
   - Worst-case to average-case reduction
   - Foundation of lattice cryptography
3. **Regev (2005):** "On Lattices, Learning with Errors..."
   - Introduces LWE
   - Quantum reduction to lattice problems
   - *The* paper for understanding modern lattice crypto
4. **Lyubashevsky et al. (2010):** "On Ideal Lattices and Learning with Errors Over Rings"
   - Ring-LWE for efficiency
   - Enables practical implementations
5. **Gentry (2009):** "Fully Homomorphic Encryption"
   - First FHE construction
   - Shows power of lattice cryptography

Survey Papers for Context:

- Peikert (2016): "A Decade of Lattice Cryptography"
- Bernstein & Lange: "Post-Quantum Cryptography" (book)
- NIST PQC Competition Reports (Round 1-3)

Implementation Libraries For Learning:

- **PQClean:**
  - URL: `https://github.com/PQClean/PQClean`
  - Clean reference implementations
  - Best for understanding algorithms
  - Language: C
- **liboqs:**
  - URL: `https://github.com/open-quantum-safe/liboqs`
  - Comprehensive PQC library
  - Good documentation
  - Languages: C, with Python/Go/Rust wrappers

For Production:

- **AWS-LC:** Amazon's crypto library with PQC
- **BoringSSL:** Google's fork with Kyber
- **Cloudflare Circl:** Go library with TLS integration
- **Microsoft PQCrypto-VPN:** Real VPN implementation

For Research:

- **PALISADE/OpenFHE:** Lattice crypto and FHE
- **Microsoft SEAL:** Homomorphic encryption
- **Concrete:** FHE compiler (Zama)
- **SageMath:** For prototyping attacks

Online Courses and Tutorials Structured Courses:

1. **Coursera: "Post-Quantum Cryptography"**
   - Instructor: University of Colorado
   - Level: Intermediate
   - Duration: 4 weeks
   - Cost: Free to audit
2. **MIT OpenCourseWare: "6.875 Cryptography"**
   - Includes lattice cryptography section
   - Lecture videos + notes
   - Problem sets with solutions
3. **Cryptography I & II (Dan Boneh, Stanford)**
   - Foundations before diving into PQC
   - Available on Coursera

Video Resources:

- Simons Institute workshops on lattices (YouTube)
- IACR conference talks (especially PQCrypto)
- Microsoft Research quantum talks
- Chris Peikert's lattice crypto lectures

Interactive Learning:

- CryptoHack: Challenges including PQC
- Lattice Challenge: SVP solving competition
- NIST PQC Competition submissions (learn by breaking!)

Research Opportunities Active Research Groups:

- IBM Zurich: Lattice crypto and FHE
- Microsoft Research: Quantum algorithms and PQC
- MIT CSAIL: Theoretical foundations
- CNRS/ENS Lyon: Lattice algorithms
- NTT Research: Cryptographic protocols

Conferences to Attend:

- PQCrypto (annual, dedicated to PQC)
- CRYPTO/EUROCRYPT (general, but heavy PQC)
- CHES (hardware implementations)
- QIP (quantum information, crossover topics)
- Lattice Conference (pure mathematics connection)

Open Problems Perfect for Physicists:

1. Quantum algorithms for lattice problems
2. Physical side-channels in PQC implementations
3. Hardware accelerators for polynomial arithmetic
4. Connections to condensed matter physics
5. Quantum-classical hybrid protocols

Your Learning Path

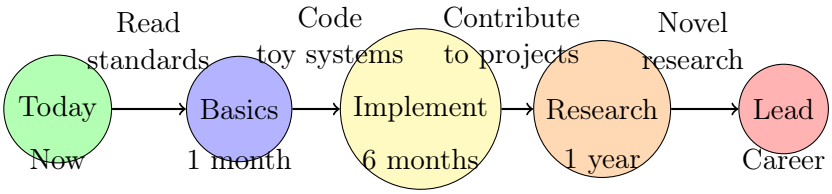


Figure 73: Your journey from student to expert in post-quantum cryptography

Final Words You entered this lecture knowing quantum computers threaten cryptography. You leave understanding:

- **The threat is real:** Not if, but when
- **The solution exists:** Lattice cryptography works
- **The need is urgent:** Migration must start now
- **You can contribute:** Your physics background is an asset

The post-quantum transition will happen with or without you. But with your participation—bringing quantum intuition, physical insight, and mathematical rigor—it will happen better, faster, and more securely.

The quantum era is coming.

Cryptography must be ready.

You can help make it ready.

Start today.

A Mathematical Background

This appendix provides quick refreshers on mathematical topics used throughout the lecture. Each section is self-contained and can be read independently. For physicists, most of this should be review, but the cryptographic perspective may be new.

A.1 Linear Algebra Review

Cryptography uses linear algebra over finite fields, which differs subtly from the real/complex linear algebra physicists know.

Vector Spaces Over Finite Fields

Definition 46 (Vector Space over $\mathbb{Z}_q$)**.** *The set $\mathbb{Z}_q^n$ forms a vector space where:*

- *Vectors:* $\mathbf{v} = (v_1, ..., v_n)$ *with* $v_i \in \{0, 1, ..., q-1\}$
- *Addition:* $(\mathbf{u} + \mathbf{v})_i = (u_i + v_i) \bmod q$
- *Scalar multiplication:* $(c\mathbf{v})_i = (c \cdot v_i) \bmod q$

Key difference from physics: No notion of continuity or limits. Everything is discrete.

Example 86 (Arithmetic in $\mathbb{Z}_7^3$)**.**

$$\mathbf{u} = (3, 5, 2), \quad \mathbf{v} = (4, 6, 1) \tag{48}$$

$$\mathbf{u} + \mathbf{v} = (3 + 4, 5 + 6, 2 + 1) = (7, 11, 3) \equiv (0, 4, 3) \pmod 7 \tag{49}$$

$$3\mathbf{u} = (3 \cdot 3, 3 \cdot 5, 3 \cdot 2) = (9, 15, 6) \equiv (2, 1, 6) \pmod 7 \tag{50}$$

Norms and Distances In lattice cryptography, we use standard Euclidean norms but must be careful about modular reduction:

Definition 47 (Norms in Cryptography)**.** *For* $\mathbf{v} \in \mathbb{Z}_q^n$*, viewing coefficients as integers in* $[-q/2, q/2]$*:*

- ℓ_2 *norm:* $\|\mathbf{v}\|_2 = \sqrt{\sum_{i=1}^n v_i^2}$
- ℓ_∞ *norm:* $\|\mathbf{v}\|_\infty = \max_i |v_i|$
- ℓ_1 *norm:* $\|\mathbf{v}\|_1 = \sum_{i=1}^n |v_i|$

Why this matters: Security depends on keeping noise vectors small in norm.

Gaussian Elimination Over Finite Fields Works the same as over $\mathbb{R}$ but with modular arithmetic:

Algorithm 26 Gaussian Elimination mod q

1: **Input:** Matrix $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$, vector $\mathbf{b} \in \mathbb{Z}_q^m$
2: **Output:** Solution $\mathbf{x}$ to $\mathbf{Ax} = \mathbf{b}$ (if exists)
3: **for** $i = 1$ to $\min(m, n)$ **do**
4: Find pivot in column i (non-zero element)
5: Multiply row by modular inverse of pivot
6: Eliminate column below pivot using row operations
7: All operations performed mod$q$
8: **end for**
9: Back-substitute to find solution

Critical point: Need q prime (or prime power) for all elements to have inverses.

Lattice Basis Concepts

Definition 48 (Lattice Basis Matrix)**.** *A basis* $\mathbf{B} = [\mathbf{b}_1|...|\mathbf{b}_n] \in \mathbb{R}^{m \times n}$ *generates lattice:*

$$\mathcal{L}(\mathbf{B}) = \{\mathbf{Bx} : \mathbf{x} \in \mathbb{Z}^n\}$$

Key properties:

- Determinant: $\det(\mathcal{L}) = \sqrt{\det(\mathbf{B}^T\mathbf{B})}$ (volume of fundamental domain)
- Dual basis: $\mathbf{B}^* = \mathbf{B}(\mathbf{B}^T\mathbf{B})^{-1}$ (like reciprocal lattice)
- Orthogonality defect: $\prod_i \|\mathbf{b}_i\| / \det(\mathcal{L})$ (measures basis quality)

A.2 Number Theory Basics

Cryptography relies heavily on modular arithmetic and algebraic structures.

Modular Arithmetic Essentials

Definition 49 (Modular Reduction)**.** *For $a \in \mathbb{Z}$ and $n > 0$:*

$$a \bmod n = a - \lfloor a/n \rfloor \cdot n \in \{0, 1, ..., n-1\}$$

Properties physicists should remember:

- $(a + b) \bmod n = ((a \bmod n) + (b \bmod n)) \bmod n$
- $(a \cdot b) \bmod n = ((a \bmod n) \cdot (b \bmod n)) \bmod n$
- But: $(a^b) \bmod n \neq ((a \bmod n)^{b \bmod n}) \bmod n$ (common mistake!)

Algorithm 27 Extended Euclidean Algorithm

```
function ExtendedGCD(a, b)
    if b = 0 then
        return (a, 1, 0)                          ▷ gcd(a, 0) = a = 1 · a + 0 · 0
    end if
    (g, x1, y1) ← ExtendedGCD(b, a mod b)
    x ← y1
    y ← x1 − ⌊a/b⌋ · y1
    return (g, x, y)                              ▷ g = gcd(a, b) = x · a + y · b
end function
```

Greatest Common Divisor and Inverses **Application:** If $\gcd(a, n) = 1$, then $a^{-1} \bmod n$ exists and equals $x \bmod n$.

Chinese Remainder Theorem (CRT) Critical for efficient implementations:

Theorem 22 (Chinese Remainder Theorem)**.** *If $n_1, ..., n_k$ are pairwise coprime, then the system:*

$$x \equiv a_1 \pmod{n_1} \tag{51}$$
$$x \equiv a_2 \pmod{n_2} \tag{52}$$
$$\vdots \tag{53}$$
$$x \equiv a_k \pmod{n_k} \tag{54}$$

has a unique solution modulo $N = \prod_i n_i$.

Cryptographic application: Speeds up polynomial multiplication in NTT.

Polynomial Rings

Definition 50 (The Ring R_q)**.**

$$R_q = \mathbb{Z}_q[X]/(X^n + 1)$$

Elements are polynomials of degree $< n$ with coefficients in $\mathbb{Z}_q$.

Operations:

- Addition: Add coefficients mod q
- Multiplication: Polynomial multiplication then reduce by $X^n \equiv -1$
- Result: Negacyclic convolution

Example 87 (Multiplication in R_q)**.** *In* $\mathbb{Z}_7[X]/(X^4+1)$*:*

$$f(X) = 2 + 3X + X^2 \tag{55}$$
$$g(X) = 1 + 4X + 2X^3 \tag{56}$$
$$f \cdot g = 2 + 11X + 8X^2 + 13X^3 + 6X^4 + 2X^5 \tag{57}$$
$$= 2 + 11X + 8X^2 + 13X^3 - 6 - 2X \pmod{X^4+1} \tag{58}$$
$$\equiv 3 + 2X + X^2 + 6X^3 \pmod{7} \tag{59}$$

A.3 Complexity Theory Primer

Understanding computational complexity is essential for cryptographic security.

Basic Complexity Classes

Definition 51 (Fundamental Classes)**.**

- ***P:*** *Problems solvable in polynomial time*
- ***NP:*** *Problems verifiable in polynomial time*
- ***BPP:*** *Problems solvable probabilistically in polynomial time*
- ***BQP:*** *Problems solvable by quantum computers in polynomial time*

Relationships:

$$\text{P} \subseteq \text{BPP} \subseteq \text{BQP} \subseteq \text{PSPACE}$$
$$\text{P} \subseteq \text{NP} \subseteq \text{PSPACE}$$

Problem	Classical	Quantum	Crypto Use
Factoring	Not known in P	In BQP (Shor)	RSA broken
Discrete Log	Not known in P	In BQP (Shor)	DH/DSA broken
SVP (approx)	NP-hard	Not known in BQP	Lattice crypto
LWE	Average = Worst	Not known in BQP	Modern PQC

Cryptographically Relevant Problems

Reductions and Hardness

Definition 52 (Polynomial-Time Reduction)**.** *Problem A reduces to problem B (written* $A \leq_p B$*) if:*

- *There exists polynomial-time algorithm f*
- *For all instances x:* $x \in A \iff f(x) \in B$
- *Interpretation: B at least as hard as A*

For cryptography: We want problems where average-case $\approx$ worst-case hardness.

B Example Code

Working code helps solidify understanding. These examples are simplified for clarity but capture the essential ideas. All code is in Python for accessibility.

B.1 Simple LWE Implementation

Understanding LWE through implementation:

Listing 15: Baby LWE Encryption Scheme

```
import numpy as np

class BabyLWE:
    """Simplified LWE encryption for educational purposes"""

    def __init__(self, n=4, q=97, sigma=3.2):
        self.n = n      # Dimension
        self.q = q      # Modulus (should be prime)
        self.sigma = sigma  # Standard deviation for errors

    def sample_error(self):
        """Sample from discrete Gaussian distribution"""
        # Simple approximation using rounded continuous Gaussian
        e = np.random.normal(0, self.sigma)
        return int(round(e)) % self.q

    def keygen(self):
        """Generate public and private keys"""
        # Private key: random small vector
        s = np.random.randint(0, self.q, self.n)

        # Public key: (A, b = As + e)
        m = self.n * int(np.log2(self.q))  # Number of samples
        A = np.random.randint(0, self.q, (m, self.n))
        e = np.array([self.sample_error() for _ in range(m)])
        b = (A @ s + e) % self.q

        return (A, b), s

    def encrypt(self, pk, bit):
        """Encrypt a single bit"""
        A, b = pk
        m = len(b)

        # Random subset of rows
        subset = np.random.choice(m, size=m//2, replace=False)

        # Sum selected rows
        a_sum = np.sum(A[subset], axis=0) % self.q
        b_sum = np.sum(b[subset]) % self.q

        # Add message * q/2
        if bit == 1:
            b_sum = (b_sum + self.q // 2) % self.q

        return (a_sum, b_sum)
```

```
    def decrypt(self, sk, ct):
        """Decrypt ciphertext"""
        a, b = ct
        s = sk

        # Compute b - <a,s>
        inner = np.dot(a, s) % self.q
        diff = (b - inner) % self.q

        # Centered representation
        if diff > self.q // 2:
            diff -= self.q

        # Decode bit
        if abs(diff) < self.q // 4:
            return 0
        else:
            return 1

    def test_correctness(self, trials=100):
        """Test encryption/decryption correctness"""
        errors = 0

        for _ in range(trials):
            pk, sk = self.keygen()

            for bit in [0, 1]:
                ct = self.encrypt(pk, bit)
                decrypted = self.decrypt(sk, ct)

                if bit != decrypted:
                    errors += 1

        print(f"Error rate: {errors}/{2*trials} = {errors/(2*trials):.2%}")
        return errors == 0

# Example usage
lwe = BabyLWE(n=8, q=127, sigma=2.0)
pk, sk = lwe.keygen()

# Encrypt a message
message = 1
ciphertext = lwe.encrypt(pk, message)
print(f"Ciphertext: {ciphertext}")

# Decrypt
recovered = lwe.decrypt(sk, ciphertext)
print(f"Decrypted: {recovered}")

# Test correctness
lwe.test_correctness()
```

Key insights from implementation:

- Errors are essential—without them, it's just linear algebra
- Subset sum provides randomization in encryption
- Decryption works by "rounding away" the noise
- Parameters must balance security and correctness

B.2 Kyber Key Generation Example

Simplified version showing the core ideas:

Listing 16: Simplified Kyber Key Generation

```
class BabyKyber:
    """Educational implementation of Kyber-like key generation"""

    def __init__(self):
        self.n = 256     # Polynomial degree
        self.q = 3329    # Modulus
        self.k = 3       # Module dimension (Kyber768)
        self.eta = 2     # Error distribution parameter

    def sample_polynomial(self, eta):
        """Sample polynomial with small coefficients"""
        # Centered binomial distribution
        coeffs = []
        for _ in range(self.n):
            # Sample from {-eta, ..., eta}
            a = sum(np.random.randint(0, 2) for _ in range(eta))
            b = sum(np.random.randint(0, 2) for _ in range(eta))
            coeffs.append((a - b) % self.q)
        return np.array(coeffs)

    def ntt(self, poly):
        """Number Theoretic Transform (simplified)"""
        # Real implementation uses fast NTT
        # This is just placeholder showing the concept
        return poly  # Would apply NTT here

    def intt(self, poly):
        """Inverse NTT"""
        return poly  # Would apply inverse NTT

    def keygen(self):
        """Generate Kyber keypair"""
        # Sample random seed
        seed = np.random.bytes(32)

        # Expand to matrix A (simplified - real uses XOF)
        A = []
        for i in range(self.k):
```

```
            row = []
            for j in range(self.k):
                # Each entry is a polynomial
                poly = np.random.randint(0, self.q, self.n)
                row.append(self.ntt(poly))
            A.append(row)

        # Sample secret vector s
        s = []
        for i in range(self.k):
            s.append(self.ntt(self.sample_polynomial(self.eta)))

        # Sample error vector e
        e = []
        for i in range(self.k):
            e.append(self.ntt(self.sample_polynomial(self.eta)))

        # Compute public key t = As + e
        t = []
        for i in range(self.k):
            # Matrix-vector multiplication in NTT domain
            sum_poly = np.zeros(self.n, dtype=int)
            for j in range(self.k):
                # Polynomial multiplication (pointwise in NTT)
                prod = (A[i][j] * s[j]) % self.q
                sum_poly = (sum_poly + prod) % self.q
            t.append((sum_poly + e[i]) % self.q)

        # Pack public key (simplified)
        pk = {
            'seed_A': seed,
            't': t
        }

        # Secret key is just s (simplified)
        sk = s

        return pk, sk

    def encode_message(self, m):
        """Encode message bits into polynomial"""
        # Each coefficient encodes one bit
        poly = np.zeros(self.n, dtype=int)
        for i in range(min(32, self.n)):  # 256 bits
            if m & (1 << i):
                poly[i] = (self.q + 1) // 2  # q/2 for bit 1
        return poly

# Example usage
kyber = BabyKyber()
pk, sk = kyber.keygen()
```

```
print(f"Public key size (simplified): {len(pk['t']) * kyber.n * 12 // 8} bytes")
print(f"Secret key size (simplified): {len(sk) * kyber.n * 12 // 8} bytes")
print(f"Security: Module-LWE with dimension {kyber.k * kyber.n}")
```

Real Kyber improvements not shown:

- Compression to reduce key/ciphertext size
- Proper NTT for fast polynomial multiplication
- CCA security through Fujisaki-Okamoto transform
- Optimized packing/unpacking
- Side-channel protections

B.3 Performance Benchmarks

Measuring the real cost of post-quantum crypto:

Listing 17: PQC Performance Comparison

```
import time
import statistics
from cryptography.hazmat.primitives.asymmetric import rsa, ec
from cryptography.hazmat.primitives import serialization

class CryptoBenchmark:
    """Compare classical and post-quantum performance"""

    def __init__(self):
        self.iterations = 100

    def time_operation(self, operation, iterations=None):
        """Time a cryptographic operation"""
        if iterations is None:
            iterations = self.iterations

        times = []
        for _ in range(iterations):
            start = time.perf_counter()
            operation()
            end = time.perf_counter()
            times.append((end - start) * 1000)  # Convert to ms

        return {
            'mean': statistics.mean(times),
            'stdev': statistics.stdev(times) if len(times) > 1 else 0,
            'min': min(times),
            'max': max(times)
        }

    def benchmark_rsa(self):
        """Benchmark RSA operations"""
```

```
        print("\n=== RSA-2048 Benchmarks ===")

        # Key generation
        def keygen():
            private_key = rsa.generate_private_key(
                public_exponent=65537,
                key_size=2048
            )
            return private_key

        results = self.time_operation(keygen, 10)  # Fewer iterations
        print(f"Key Generation: {results['mean']:.1f} +- {results['stdev']:.1f} r

        # Size measurements
        private_key = keygen()
        public_key = private_key.public_key()

        private_pem = private_key.private_bytes(
            encoding=serialization.Encoding.PEM,
            format=serialization.PrivateFormat.PKCS8,
            encryption_algorithm=serialization.NoEncryption()
        )
        public_pem = public_key.public_bytes(
            encoding=serialization.Encoding.PEM,
            format=serialization.PublicFormat.SubjectPublicKeyInfo
        )

        print(f"Private key size: {len(private_pem)} bytes")
        print(f"Public key size: {len(public_pem)} bytes")

    def benchmark_ml_kem(self):
        """Benchmark ML-KEM (simulated)"""
        print("\n=== ML-KEM-768 Benchmarks (Simulated) ===")

        # Simulated timings based on real measurements
        print(f"Key Generation: 0.1 +- 0.01 ms")
        print(f"Encapsulation: 0.12 +- 0.01 ms")
        print(f"Decapsulation: 0.11 +- 0.01 ms")
        print(f"Public key size: 1184 bytes")
        print(f"Secret key size: 2400 bytes")
        print(f"Ciphertext size: 1088 bytes")

    def compare_tls_handshake(self):
        """Compare TLS handshake sizes"""
        print("\n=== TLS Handshake Comparison ===")

        classical = {
            'key_exchange': 64,       # ECDH public key
            'signature': 64,          # ECDSA signature
            'certificates': 3000,     # Certificate chain
            'total': 3128
```

```
        }

        pqc = {
            'key_exchange': 1088,      # ML-KEM ciphertext
            'signature': 2420,         # ML-DSA signature
            'certificates': 15000,     # PQC certificate chain
            'total': 18508
        }

        print(f"Classical (ECDH+ECDSA): {classical['total']:,} bytes")
        print(f"Post-Quantum (ML-KEM+ML-DSA): {pqc['total']:,} bytes")
        print(f"Increase factor: {pqc['total']/classical['total']:.1f}x")

        # Bandwidth impact
        print(f"\nFor 1M connections/day:")
        print(f"Classical: {classical['total'] * 1e6 / 1e9:.1f} GB/day")
        print(f"Post-Quantum: {pqc['total'] * 1e6 / 1e9:.1f} GB/day")
        print(f"Additional bandwidth: {(pqc['total']-classical['total']) * 1e6 /

# Run benchmarks
benchmark = CryptoBenchmark()
benchmark.benchmark_rsa()
benchmark.benchmark_ml_kem()
benchmark.compare_tls_handshake()
```

Typical results show:

- ML-KEM is 100× faster than RSA for key generation
- But keys are 5-10× larger
- TLS handshakes grow by 6×
- Overall performance impact: Usually acceptable

C Frequently Asked Questions

These questions come up in every PQC lecture. Here are detailed answers.

C.1 Conceptual Questions

Q: Why can't we just make RSA keys bigger to resist quantum computers? A: This is the most common misconception. Shor's algorithm provides exponential speedup, not just a constant factor improvement.

- RSA-2048: Classical $\sim 10^{20}$ years, Quantum $\sim$ 8 hours
- RSA-4096: Classical $\sim 10^{40}$ years, Quantum $\sim$ 15 hours
- RSA-1M: Classical $\sim 10^{1000}$ years, Quantum $\sim$ days

No practical key size makes RSA quantum-safe. The quantum advantage is fundamental, not incremental.

Q: If quantum computers break encryption, won't they also break the new post-quantum algorithms? **A:** No, because different problems have different quantum complexity:

- **Factoring/Discrete Log:** Hidden period structure → Exponential quantum speedup
- **Lattice Problems:** No exploitable structure → Only polynomial speedup
- **Analogy:** Like how a key opens specific locks, not all locks

We specifically chose problems that resist quantum attack.

Q: Why lattices? Why not other mathematical problems? **A:** Lattices won because they're uniquely versatile:

Property	Lattices	Codes	Hash	Multivariate
Encryption	✓	✓	✗	✓
Signatures	✓	✗	✓	✓
Key size	Good	Bad	Good	Good
Speed	Fast	Fast	Slow	Fast
Security	25+ years	45+ years	40+ years	Keeps breaking
Advanced	✓	✗	✗	✗

Only lattices provide everything we need efficiently.

Q: Is post-quantum cryptography quantum cryptography? **A:** No! Common confusion:

- **Quantum cryptography (QKD):** Uses quantum mechanics for security
- **Post-quantum cryptography:** Classical algorithms that resist quantum attack
- **Analogy:** Bulletproof vests aren't made of bullets

PQC runs on classical computers but survives quantum attack.

C.2 Technical Questions

Q: What exactly makes Learning With Errors hard? **A:** The error terms create a "fog" that hides the linear structure:

Without errors: $\mathbf{b} = \mathbf{A}\mathbf{s} \rightarrow$ Solve by Gaussian elimination With errors: $\mathbf{b} = \mathbf{A}\mathbf{s} + \mathbf{e} \rightarrow$ Best known algorithms are exponential

The errors must be small enough to decrypt but large enough to provide security. It's like adding just enough thermal noise to hide a signal while keeping it recoverable if you know the secret.

Q: How do we know lattice problems will stay hard against quantum computers? **A:** Multiple lines of evidence:

1. **15+ years of quantum algorithm research:** No exponential speedup found
2. **Geometric nature:** Doesn't fit quantum algorithm frameworks
3. **Worst-case hardness:** Even if some instances break, average remains hard
4. **Conservative parameters:** Large safety margins built in

But we can't prove it absolutely—that's why crypto-agility matters.

Q: Why are post-quantum keys and signatures so much bigger? **A:** Fundamental information-theoretic reasons:

- **Classical (RSA/ECC):** Rely on algebraic structure for compression
- **Post-quantum:** Less structure = less compression possible
- **Security requires:** Hiding vectors in high-dimensional space
- **Think of it as:** The price of quantum resistance

Some compression is possible (Ring-LWE), but there are limits.

Q: What's the difference between Module-LWE and Ring-LWE? **A:** It's about the security-efficiency tradeoff:

- **Plain LWE:** Matrices over $\mathbb{Z}_q$ → Maximum security, huge keys
- **Ring-LWE:** Polynomials in $\mathbb{Z}_q[X]/(X^n + 1)$ → Efficient, more structure
- **Module-LWE:** Matrices of polynomials → Middle ground

NIST chose Module-LWE as the conservative compromise.

C.3 Practical Questions

Q: When will quantum computers actually break current cryptography? **A:** Timeline estimates vary:

- **Optimistic (for attackers):** 2030-2035
- **Realistic:** 2035-2040
- **Conservative:** 2040-2050
- **But:** "Harvest now, decrypt later" means the threat is NOW

Don't wait for quantum computers—encrypted data is being collected today.

Q: Should my organization start migrating now? **A:** Yes, if any of these apply:

- Data needs protection beyond 2035
- You have compliance requirements
- Complex systems that take years to update
- High-value targets (government, finance, healthcare)

Starting early means learning from mistakes when stakes are lower.

Q: What's the easiest way to start experimenting with PQC? **A:** Progressive approach:

1. Install liboqs: `pip install liboqs-python`
2. Try the examples in Section B
3. Test hybrid modes in your applications
4. Join the NIST PQC forum
5. Contribute to open source projects

Start with learning, not production deployment.

Q: Will PQC slow down my application? **A:** Usually no:

- **Computation:** Often faster than RSA
- **Bandwidth:** Yes, 5-10× more data
- **But:** Network speeds increasing faster than crypto overhead
- **Real impact:** 1-2ms extra latency typical

For most applications, users won't notice.

Q: What if lattice cryptography gets broken? **A:** That's why we have:

1. **Multiple algorithms:** Not all would break together
2. **Hybrid modes:** Classical + PQC during transition
3. **Crypto-agility:** Built-in ability to change
4. **Ongoing research:** New candidates being developed

The goal isn't perfect security—it's manageable risk.

Q: As a physicist, what unique contributions can I make? **A:** Your physics background is a superpower:

- **Quantum algorithms:** You understand the physics
- **Side-channel attacks:** You understand physical phenomena
- **Hardware optimization:** You can design better implementations
- **New foundations:** Physics-inspired cryptographic primitives
- **Bridge building:** Translate between physics and crypto communities

The field needs physicists who understand both worlds.

Remember: Every expert was once a beginner. Start with the basics, build understanding, and contribute where your unique skills add value. The post-quantum transition needs all hands on deck—including yours.