The State of Quantum-Safe Cryptography

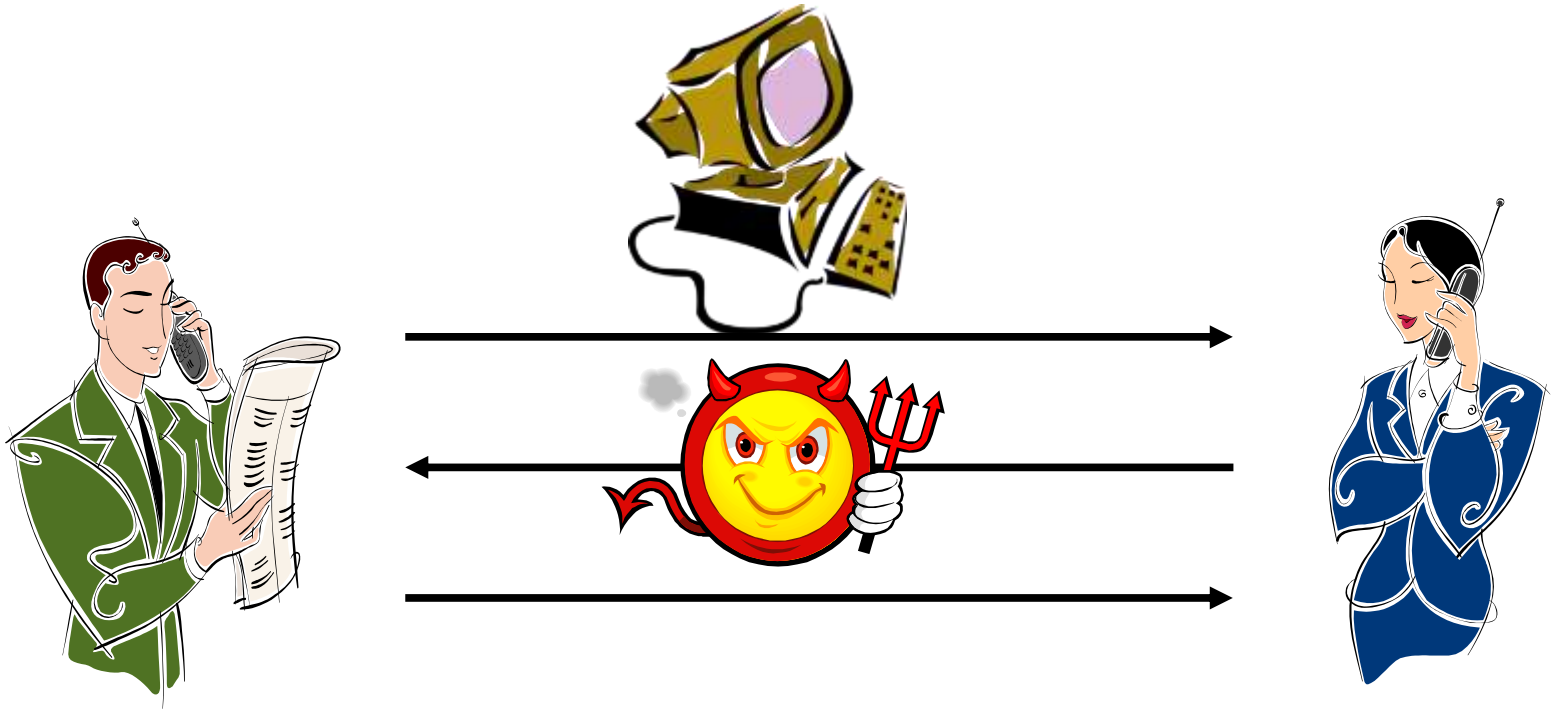
Vadim Lyubashevsky

Principal Research Scientist

IBM Research Europe, Zurich

Cryptography

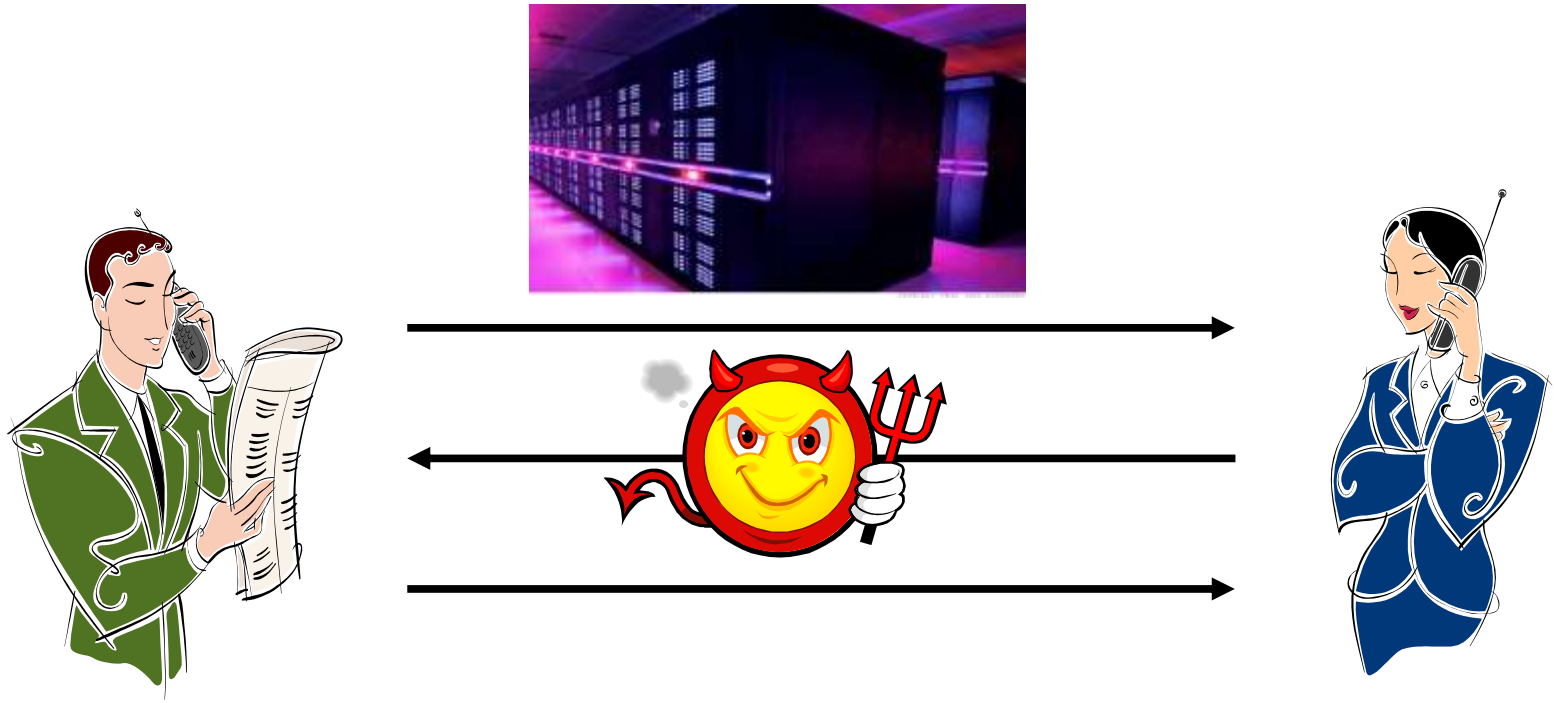
Allows for secure communication in the presence of malicious parties



Cryptography

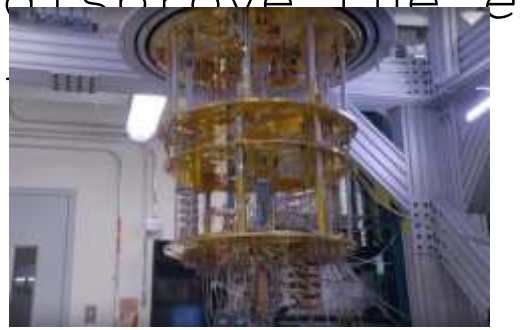
Large increase in the adversary's computing power

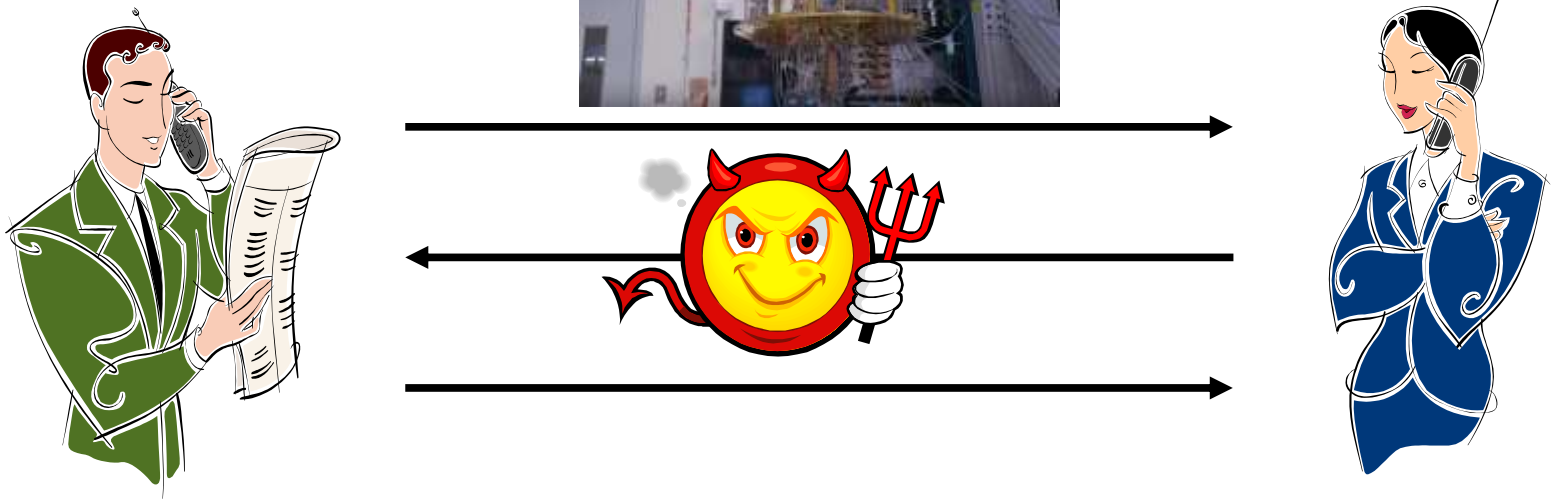
requires only a small increase in the key size



Cryptography

A quantum computer is outside the standard
(i.e. “Turing Machine”)
model of computation for efficiency purposes
Quantum computers disprove the *extended* Church-
T

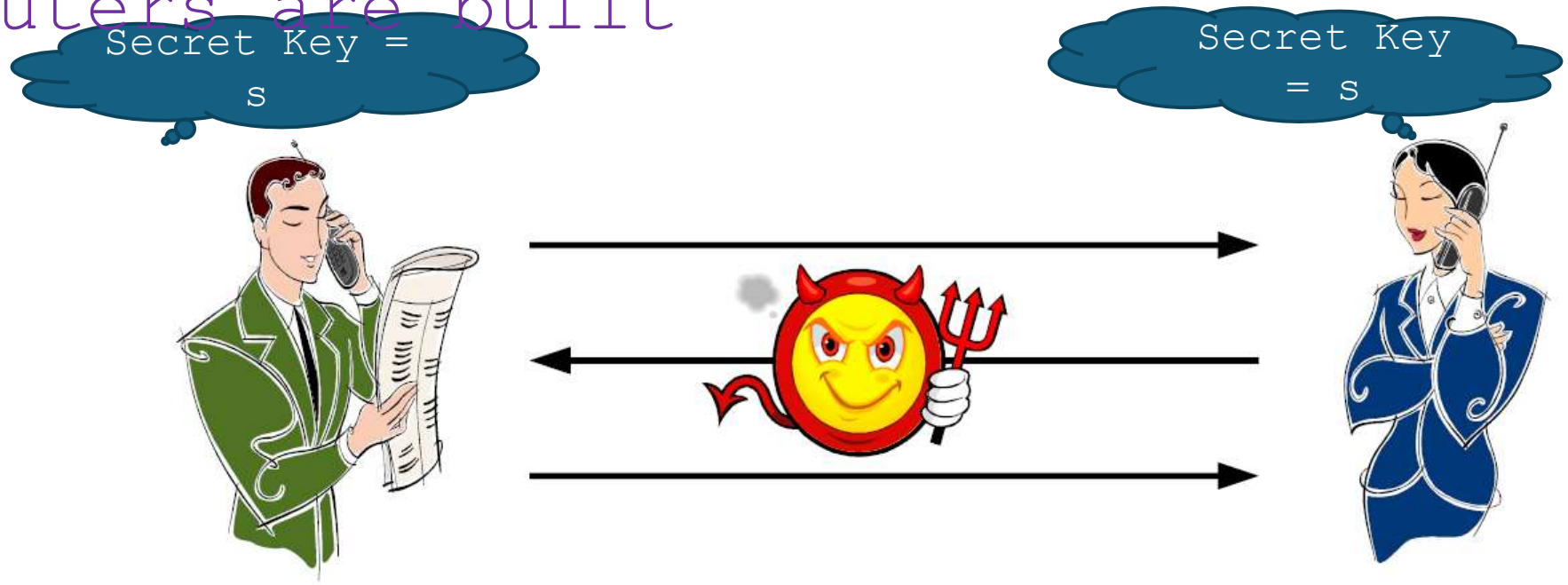




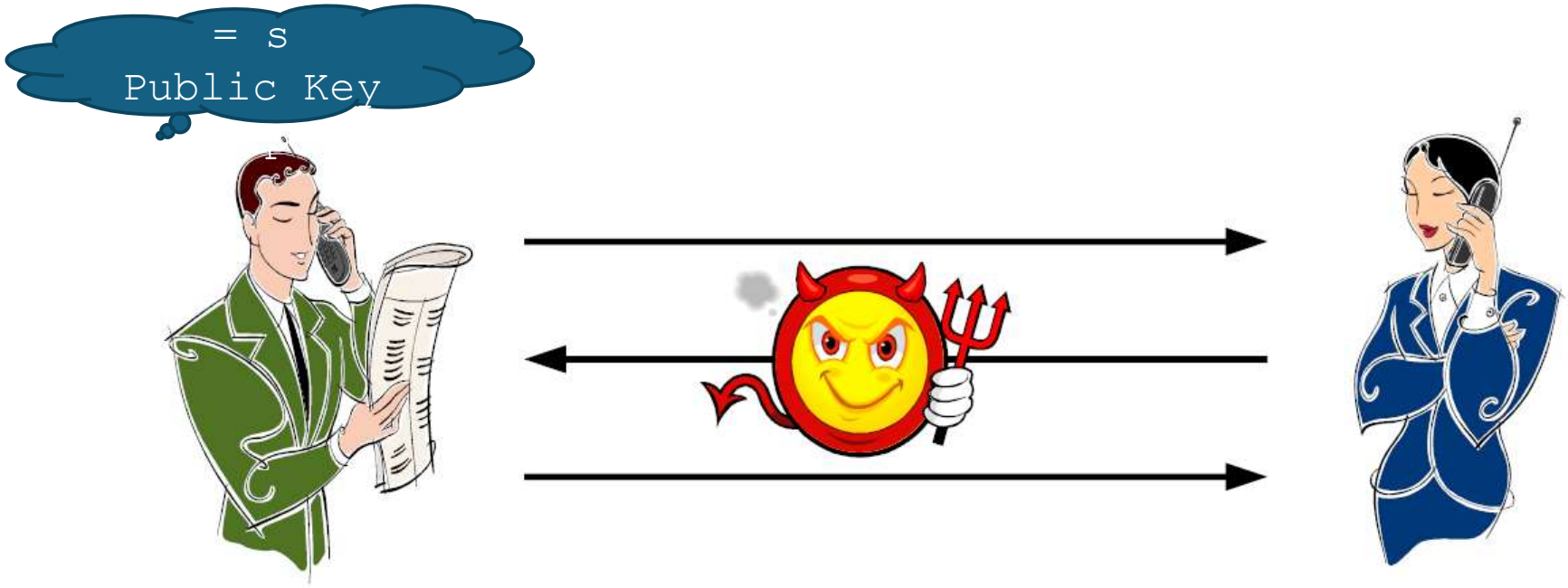
Symmetric-Key Cryptography

Symmetric-Key Cryptography

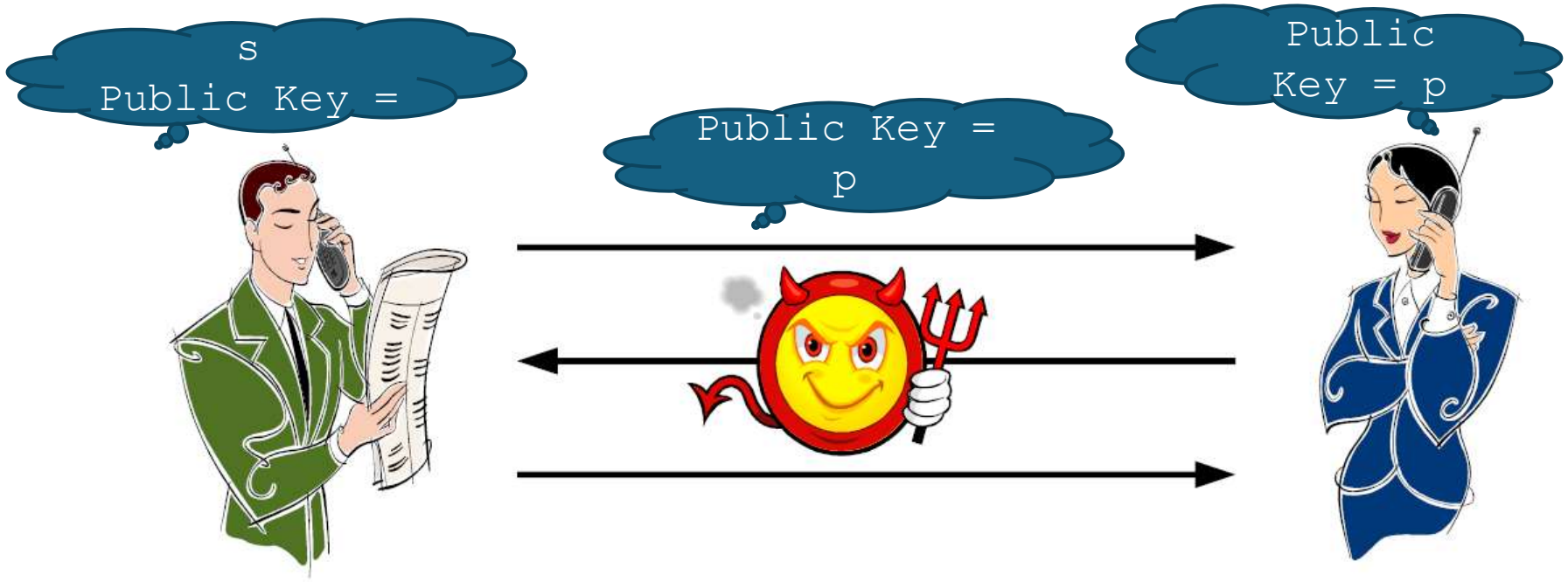
Still exists in current state when quantum computers are built



Public-Key Cryptography



Public-Key Cryptography



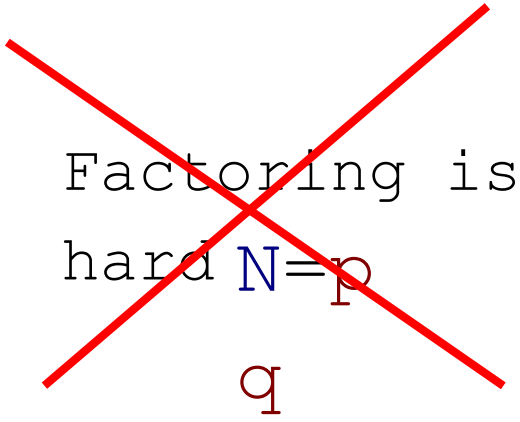
Effect of quantum computers

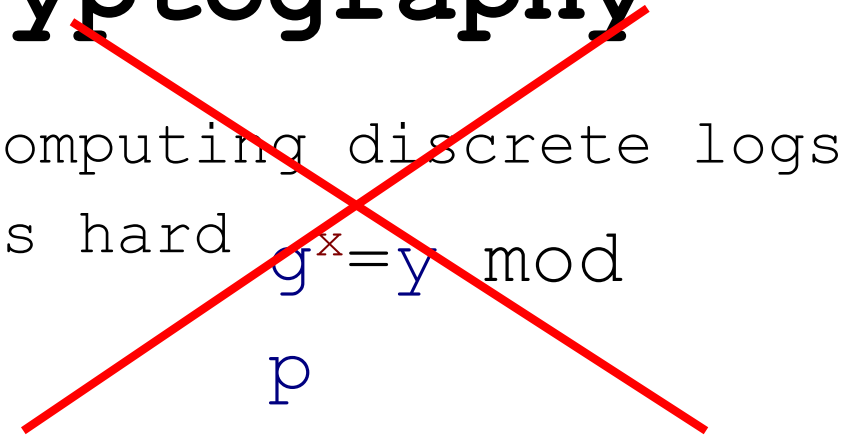
- Symmetric Cryptography
  - AES, SHA, etc. are all OK if you increase key / output size
  - e.g. use AES-256 (instead of 128),
  - hash onto a 384-bit range (instead of 256)

  - … though even that is arguable. Staying at 128 may be enough for a long time

- Public-Key Cryptography
  - Everything used up until a few years ago is broken !

Mathematical assumptions for public-key cryptography



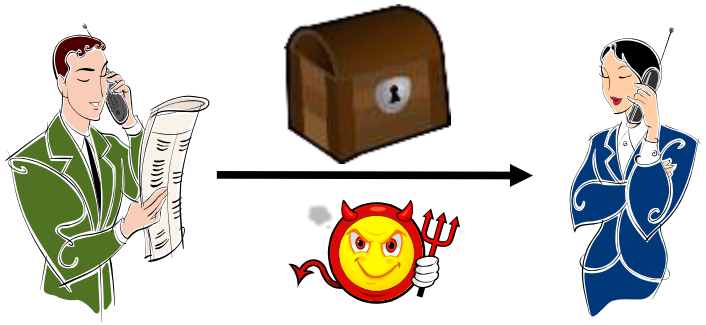


Mostly problems from number theory

All broken once a quantum computer is built

Why we care about quantum computing today

Harvest **N**ow **D**ecrypt **L**ater attacks



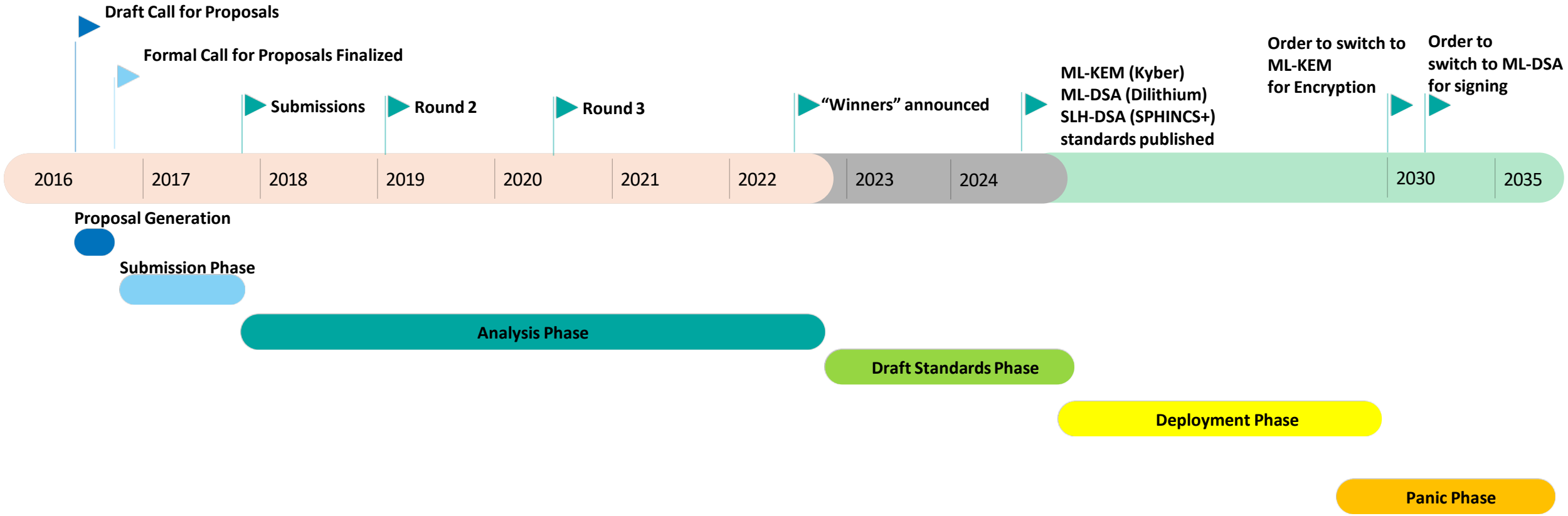
Do not need quantum to defend against quantum

Quantum computers are not all-powerful.

They simply solve some problems faster.

Base cryptography on problems they don’t solve.

Quantum-Safe Cryptography Standardization Timeline at NIST (US National Institute of Standards and Technology)



Cryptography at IBM Research (Zurich)

We are all-in on Quantum-Safe Cryptography

Quantum-Safe Encryption and Signature Standards

	IBM		Tota
Round 1	3	/	69
Round 2	3	/	26
Round 3	3	/	15
Standard	3	/	5

All 3 IBM submissions became NIST standards

Additional Quantum-Safe Signature Standards

	IBM		Tota
Round 1	4	/	50
Round 2	4	/	14
Round 3	4	/	9
Standard	?	/	?

All 4 IBM submissions made it through the first 3 rounds of the (2nd) NIST competition

transitioning more complex protocols

Simple Protocols

Classic TLS

Classic encryption

Classic signatures

Quantum-Safe TLS

ML-KEM

ML-DSA

NIST standards already used in products you use …

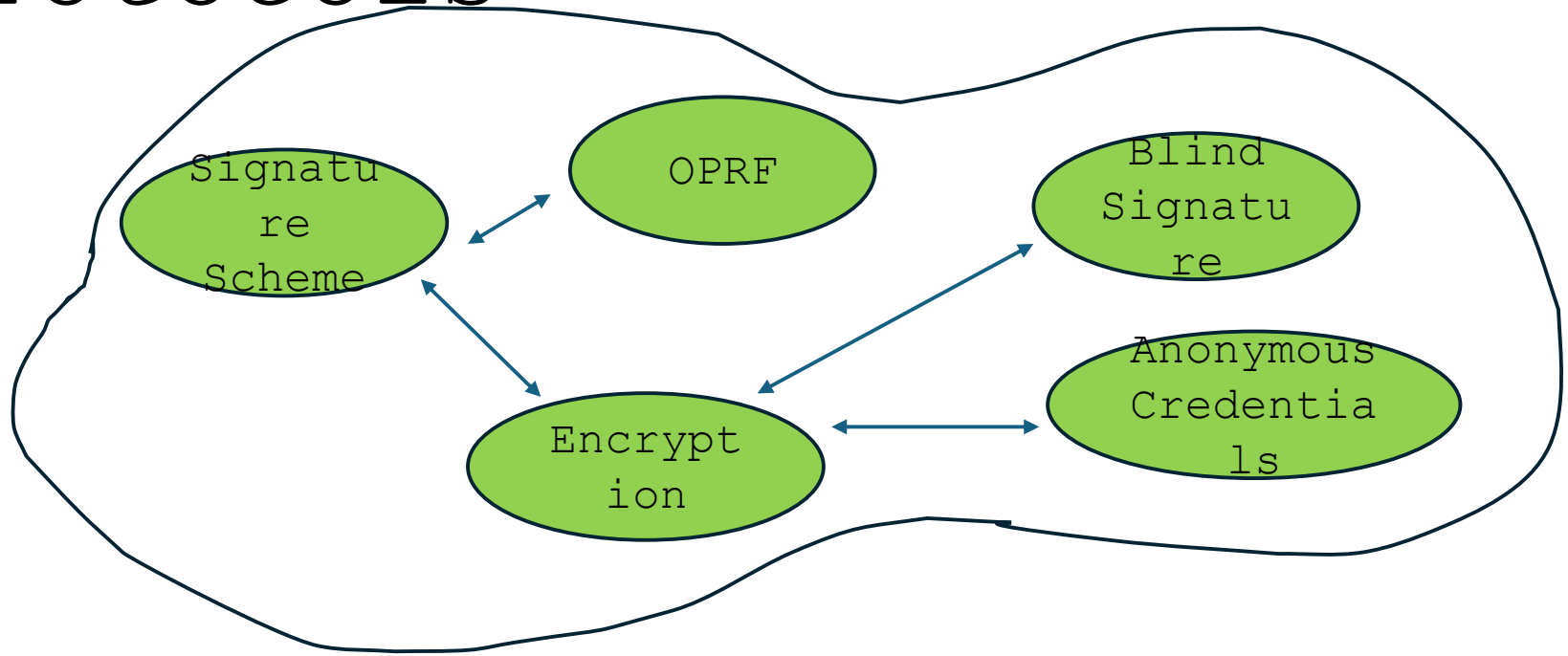
CLOUDFLARE

Akamai

iOS

Red Hat
Enterprise Linux

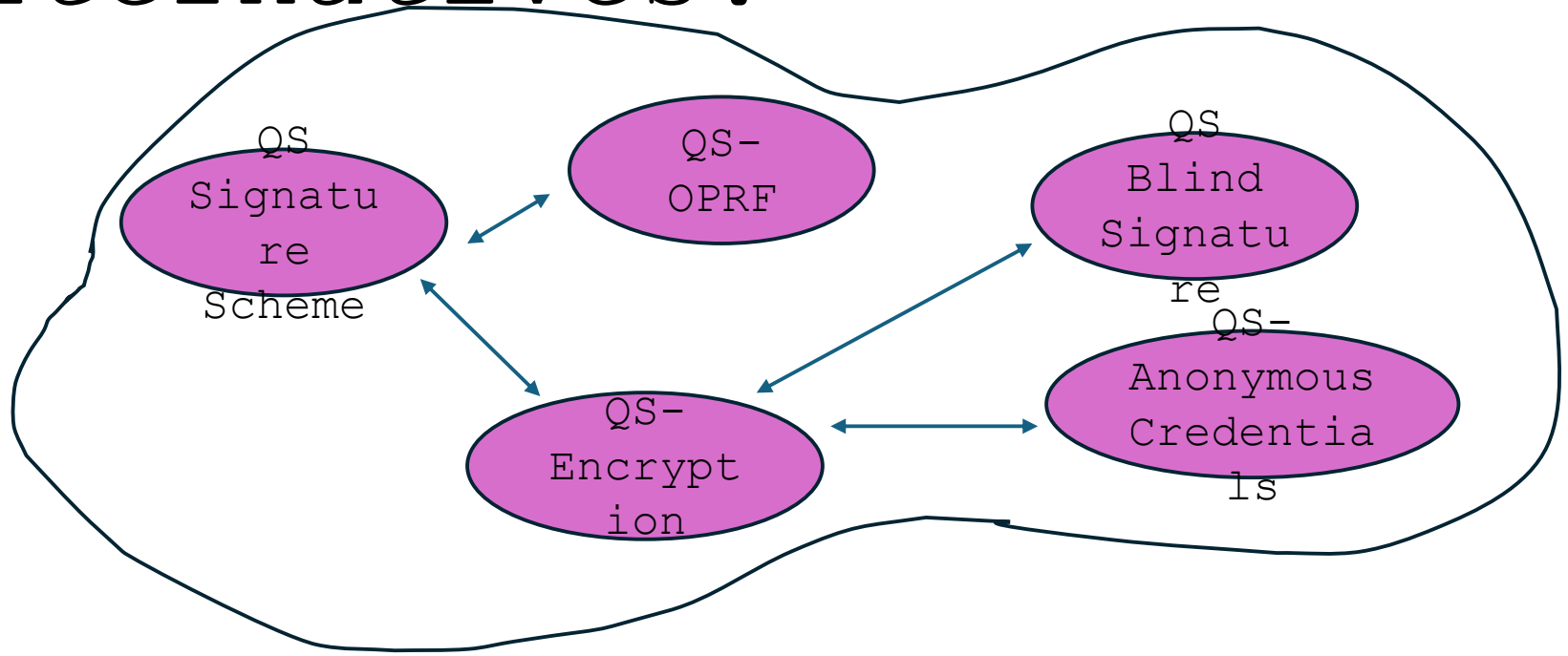
Classically-secure "Complex Protocols"



Compact and efficient

Rich algebraic structure (esp. pairings) makes everything work nicely together

Just plug in quantum-safe alternatives?



QS primitives (noticeably) less efficient

Some do not have the right algebraic structure to properly interact

The inefficiency multiplies the more primitives you try to

Constraint: No one wants to use more cryptography than is needed

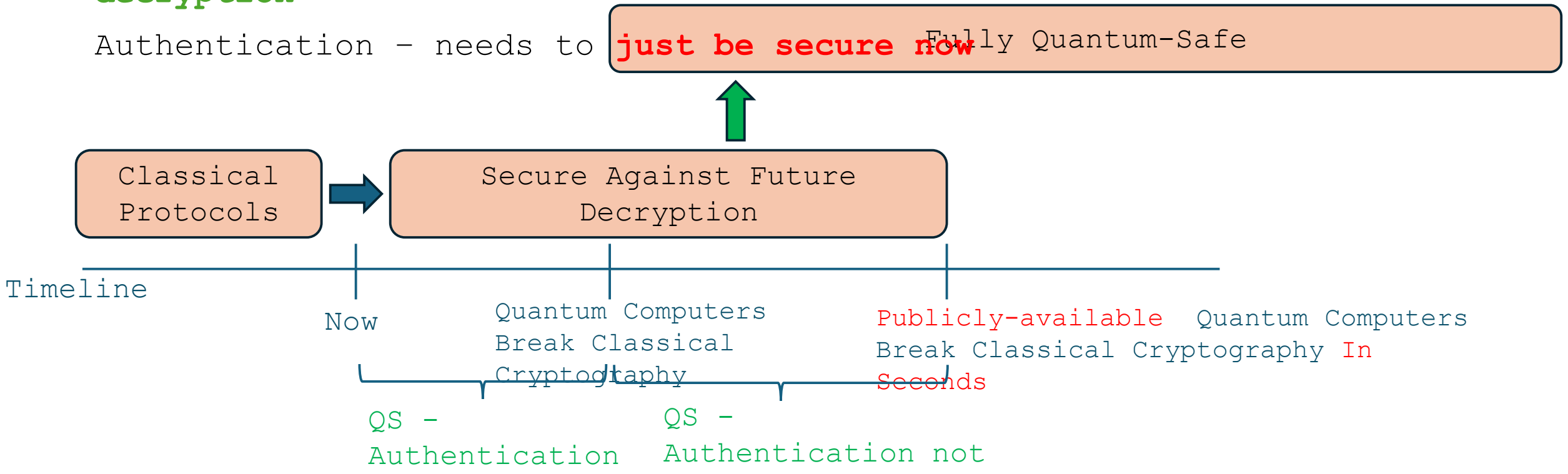
Fact: QS cryptography is generally less efficient than classical

Conclusion: **Only use QS cryptography where and when necessary**

(Roughly) two types of cryptographic building blocks:

Encryption – needs to be **secure against future decryption**

Authentication – needs to **just be secure now**



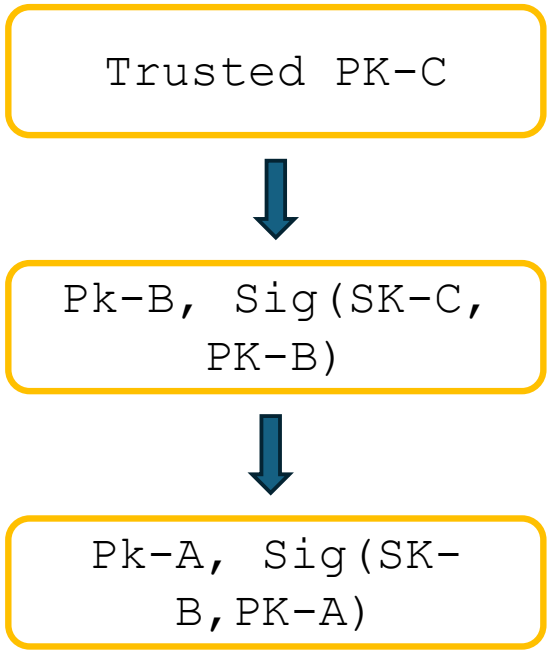
How to Design Systems for the QS Future

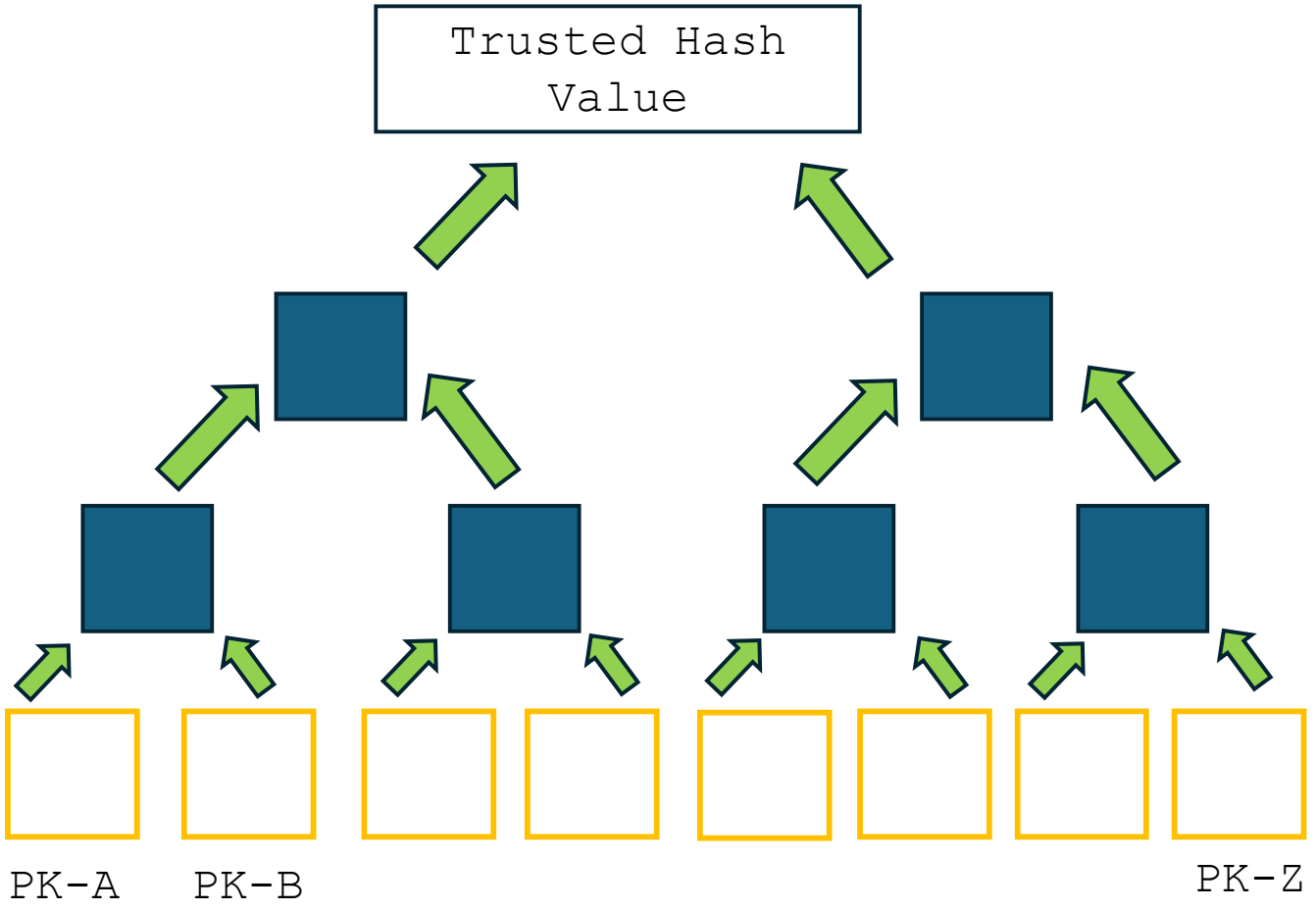
1. Understand what needs to be quantum-safe now vs later
   - Encryption should be QS now because of HNDL attacks
   - If we're sure there are no quantum computers now, digital signatures don't necessarily need to be QS yet

2. Design an *efficient* fully QS equivalent system

3. Use QS components that need to be QS now, and be able to seamlessly transition to a full QS-scheme later

Certificate Chain → Merkle Tree Certificates





One ML-DSA signature is already around 3KB

Proving a leaf is 32Bytes per level

Case study: Signal

- Using FN-DSA with optimizations
  - Key-recovery mode and message-recovery signature scheme modes
  - Saves 20% in transmission rates
  - Hoping to standardize for use outside of signal
- Re-doing private group management
  - Current scheme would be too inefficient with quantum-safe components
  - Had to rethink the protocol from the ground-up

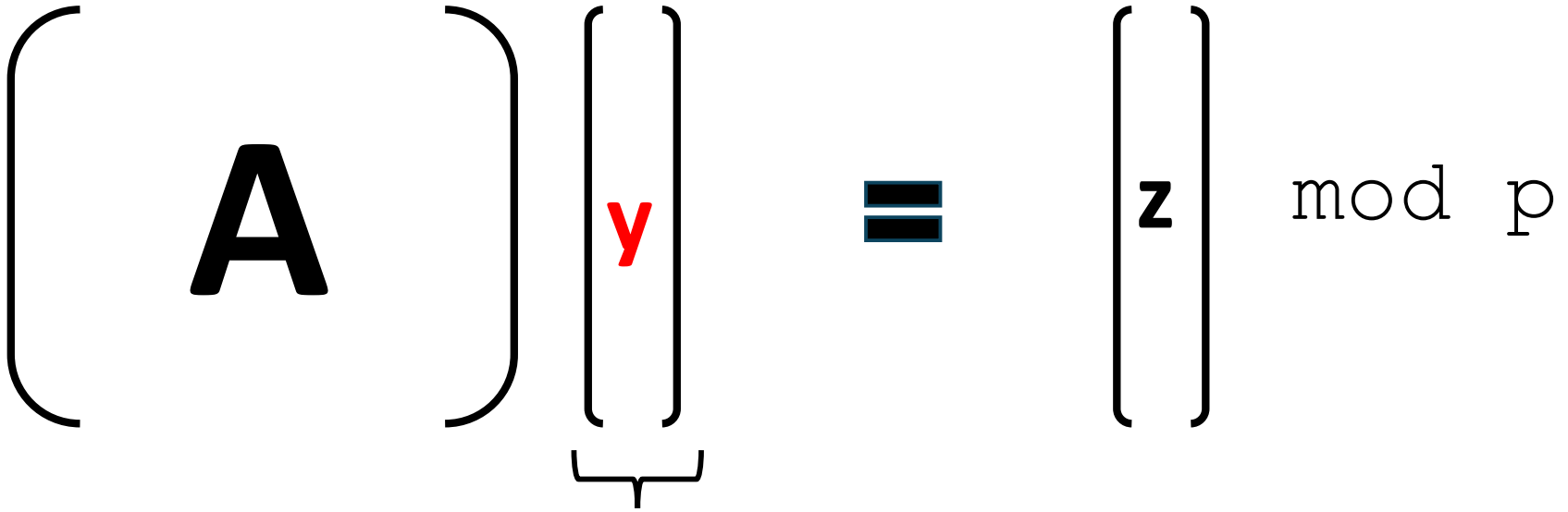
quantum-safe foundational cryptography

3 out of the 5 NIST standards

including both of the main standards (ML-KEM and ML-DSA)

are based on the hardness of **lattice problems**

Hard Problem Intuition

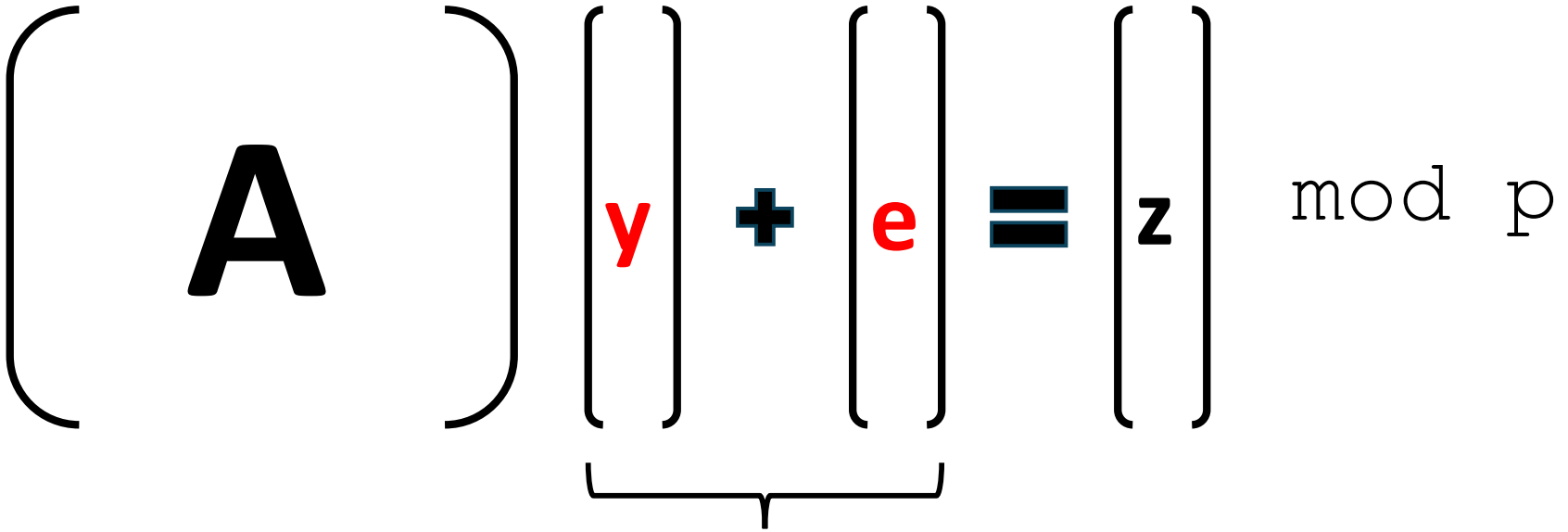


Given (**A**,**z**), find **y**

Easy! Use Gaussian elimination.

(* the smallness of y's coefficients is not relevant here)

Hard Problem Intuition



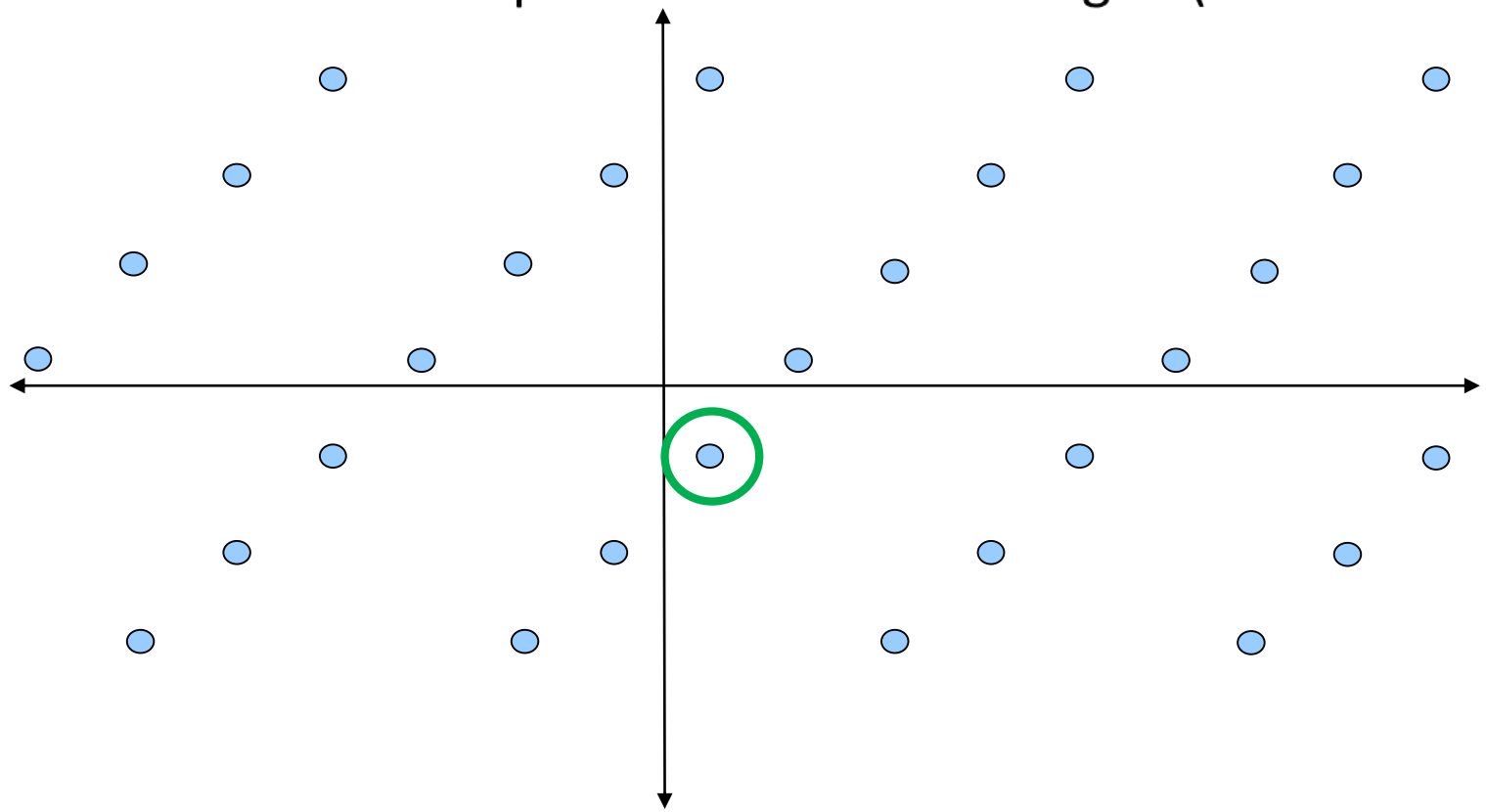
Given (**A**,**z**), find (**y**,**e**)

Seems hard.

Why is this "Lattice" Crypto?

All solutions $\begin{pmatrix} y \\ e \end{pmatrix}$ to **Ay+e=z** mod p form a "shifted" lattice.

We want to find the point closest to the origin (BDD Problem).



CRYSTALS

Cryptographic Suite for Algebraic Lattices

- Kyber: A CCA-secure KEM → Became ML-KEM
  - Encryption scheme
  - (Authenticated) key exchange
- Dilithium: A Digital Signature Scheme → Became ML-DSA
  - Signing digital documents

All are very easily implemented in constant time → no side channels

crystals math

Operations in CRYSTALS

Only two main operations needed (and both are very fast):

1. Evaluations of a hash function (like SHA-3)
2. Add / multiply in the polynomial ring $Z_p[X]/(X^{256}+1)$
   - $p = 2^{13} - 2^9 + 1$ (for Kyber)
   - $p = 2^{23} - 2^{13} + 1$ (for Dilithium)

To increase security, just do more of the same operations

The exact same hardware/software can be reused

Example ring $Z_{17}[x]/(x^4+1)$

Elements are $z(x)=z_3x^3+z_2x^2+z_1x+z_0$

where z_i are integers mod 17

Addition is the usual coordinate-wise addition

Multiplication is the usual polynomial multiplication followed by reduction modulo x^4+1

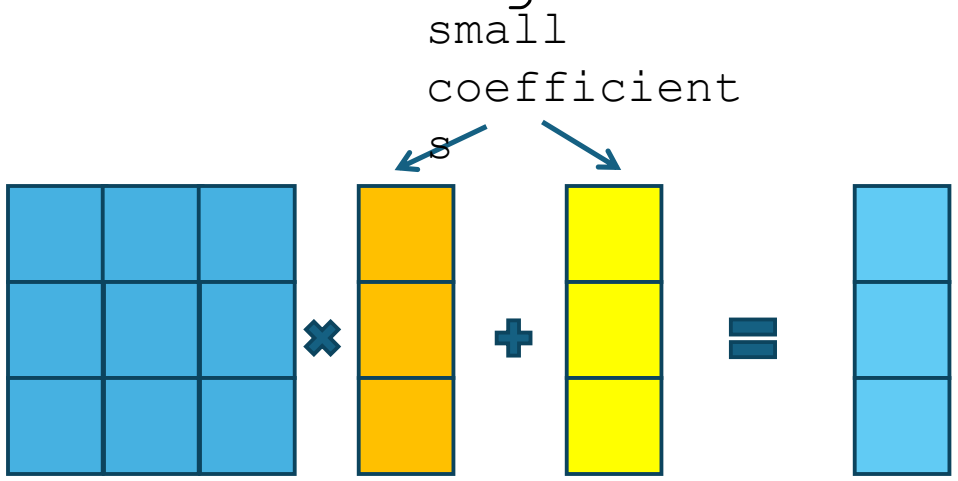
Operations in CRYSTALS

Basic Computational Domain:

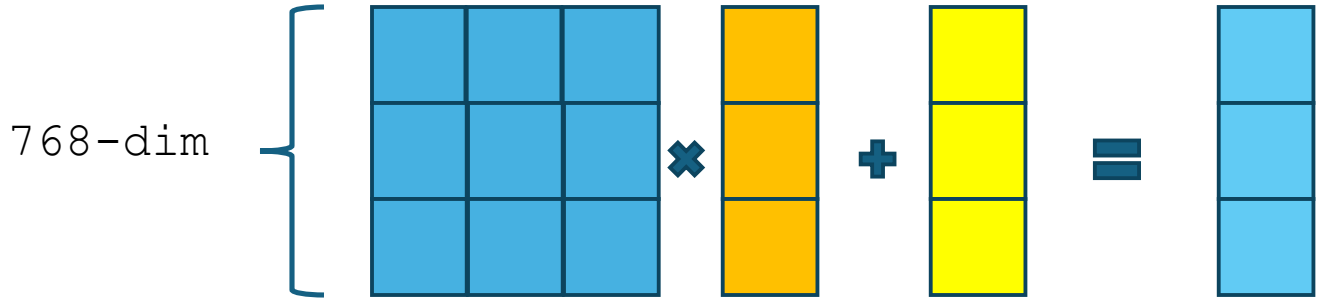
Polynomial ring $Z_p[x]/(x^{256}+1)$

Operations used in the schemes: + ×

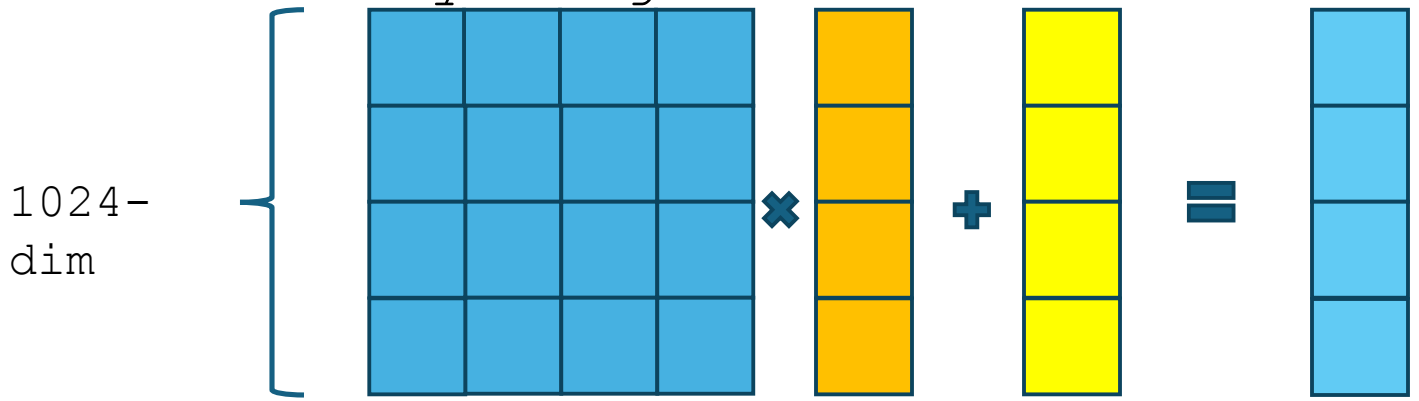
and in the ring:



Modular security



to increase the security margin



Operations in $R=Z_p[x]/(x^n+1)$

- Additions are easy and cheap
- To get cheap multiplications, we need to pick a special p

Want to write $x^n+1 = (x-r_1)(x-r_2) \cdot \cdot \cdot (x-r_n) \bmod p$

- Polynomials in R can be represented in two ways

Coefficient representation

$$f = a_0 + a_1 x + \cdots + a_{n-1} x^{n-1}$$

$$\hat{f} = (f(r_1), f(r_2), \ldots, f(r_n))$$

Chinese Remainder Theorem (CRT) representation

Multiplication using NTT (Number Theoretic Transform)

Want to multiply $f * g$, which are in coefficient representation

1. $f \rightarrow \hat{f} = (f(r_1), f(r_2), \ldots, f(r_n))$
2. $g \rightarrow \hat{g} = (g(r_1), g(r_2), \ldots, g(r_n))$
3. $\widehat{f * g} = \big(f(r_1) * g(r_1), \ldots, f(r_n) * g(r_n)\big)$
4. $\widehat{f * g} \rightarrow f * g$

Example in $R=Z_{17}[x]/(x^4+1)$

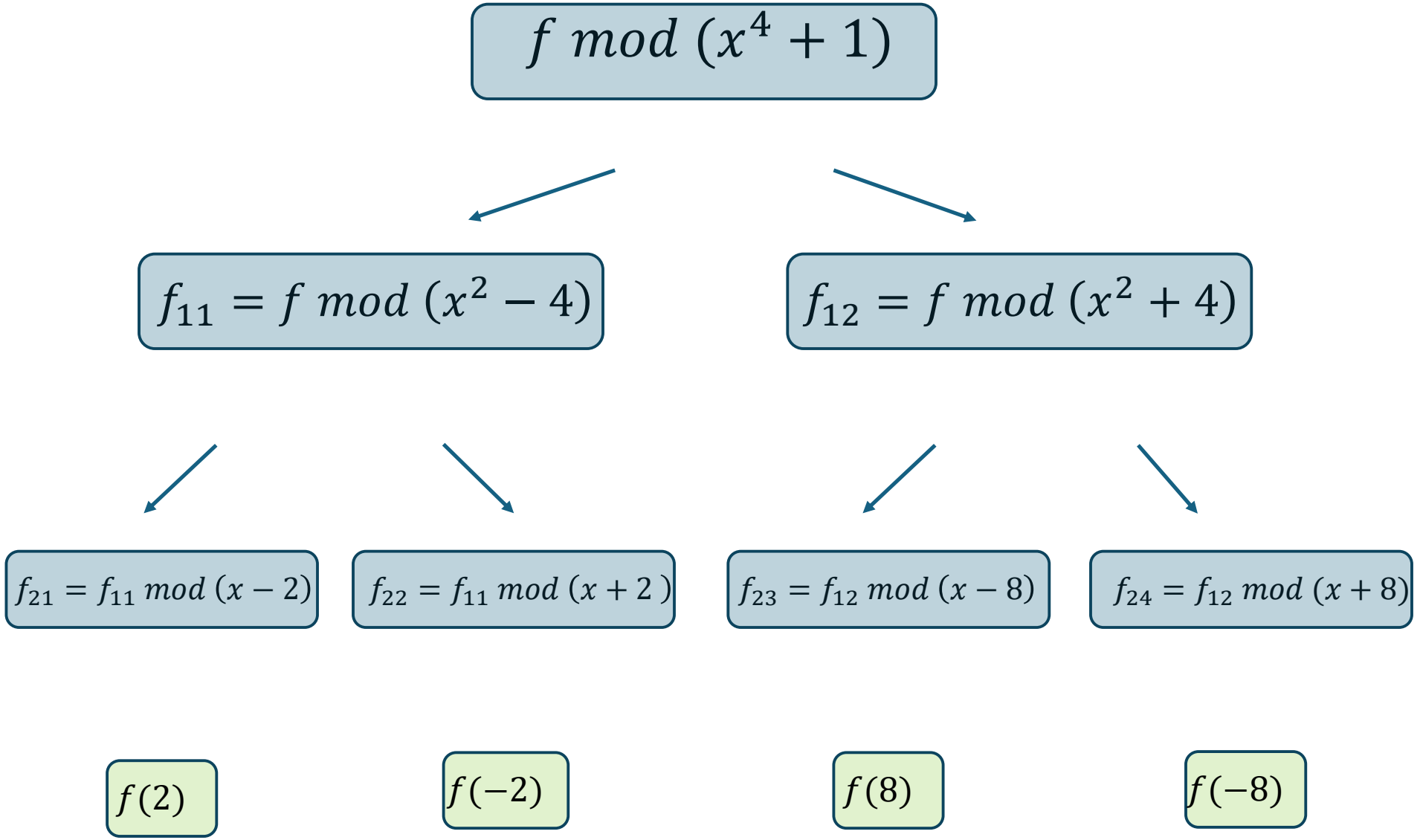
$$x^4 + 1 = (x-2)(x+2)(x-8)(x+8) \; mod \; 17$$

We will show how to efficiently convert between the coefficient and CRT representations

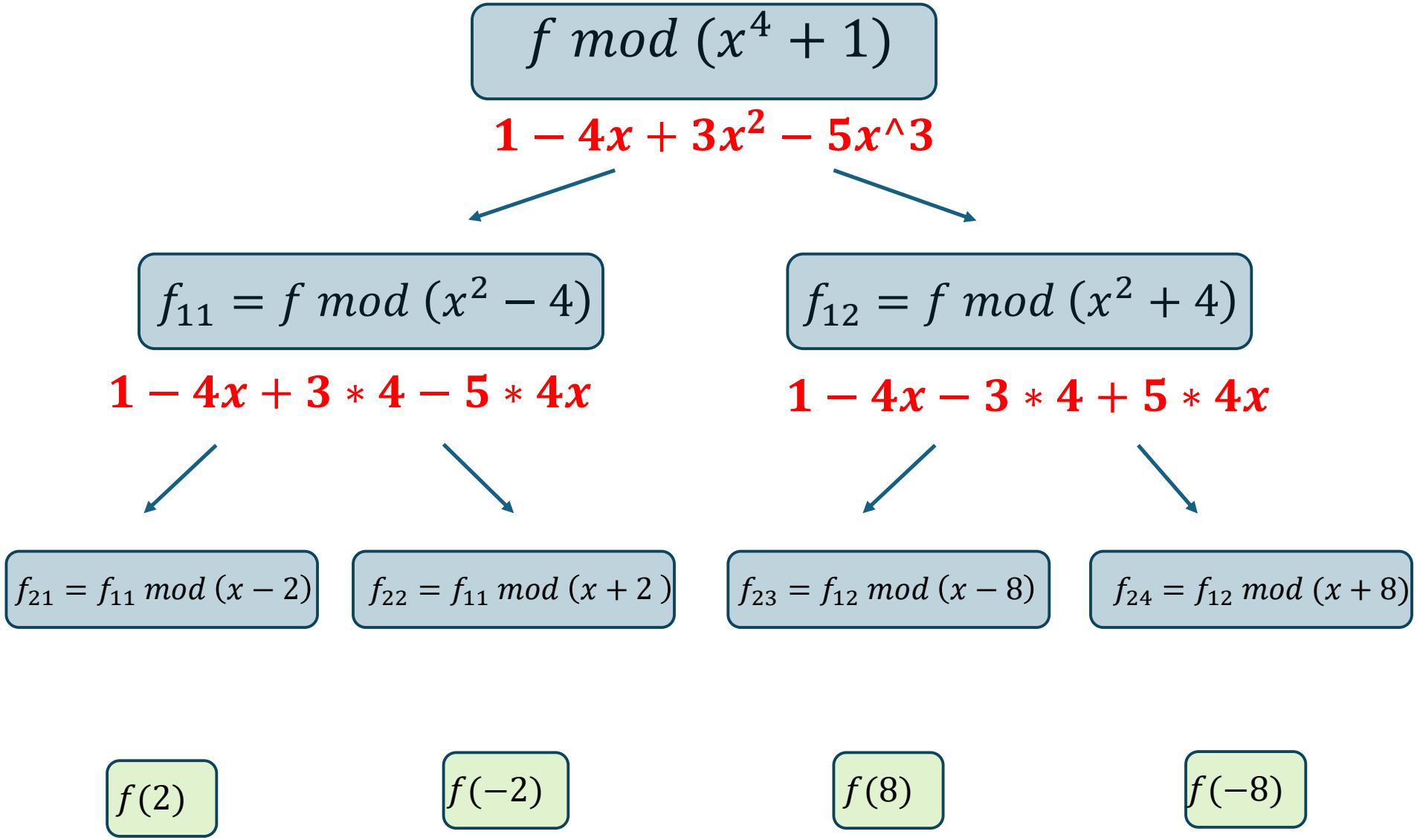
This is the main part of polynomial multiplication

(A lot of the time, one of the multiplicands is already in NTT representation)

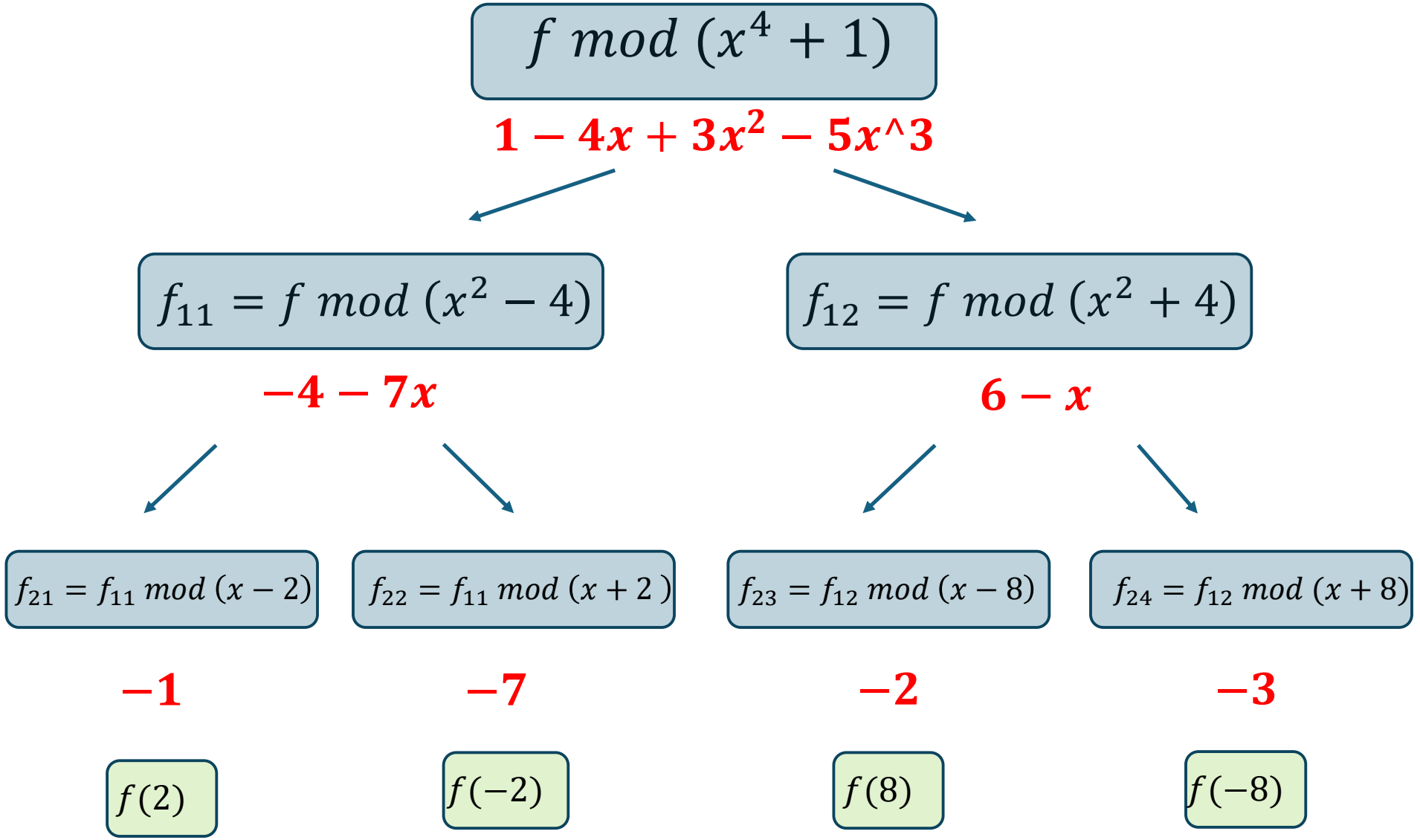
$$f \rightarrow \hat{f}$$



$$f \rightarrow \hat{f}$$



$$f \rightarrow \hat{f}$$



$$\hat{f} \rightarrow f$$

$$f \bmod (x^4 + 1)$$

$$f_{11} = f \bmod (x^2 - 4)$$

$$f_{12} = f \bmod (x^2 + 4)$$

$a + bx$

$a + 2b = -1$

$a - 2b = -7$

$f_{21} = f_{11} \bmod (x - 2)$

$f_{22} = f_{11} \bmod (x + 2)$

$f_{23} = f_{12} \bmod (x - 8)$

$f_{24} = f_{12} \bmod (x + 8)$

-1

-7

-2

-3

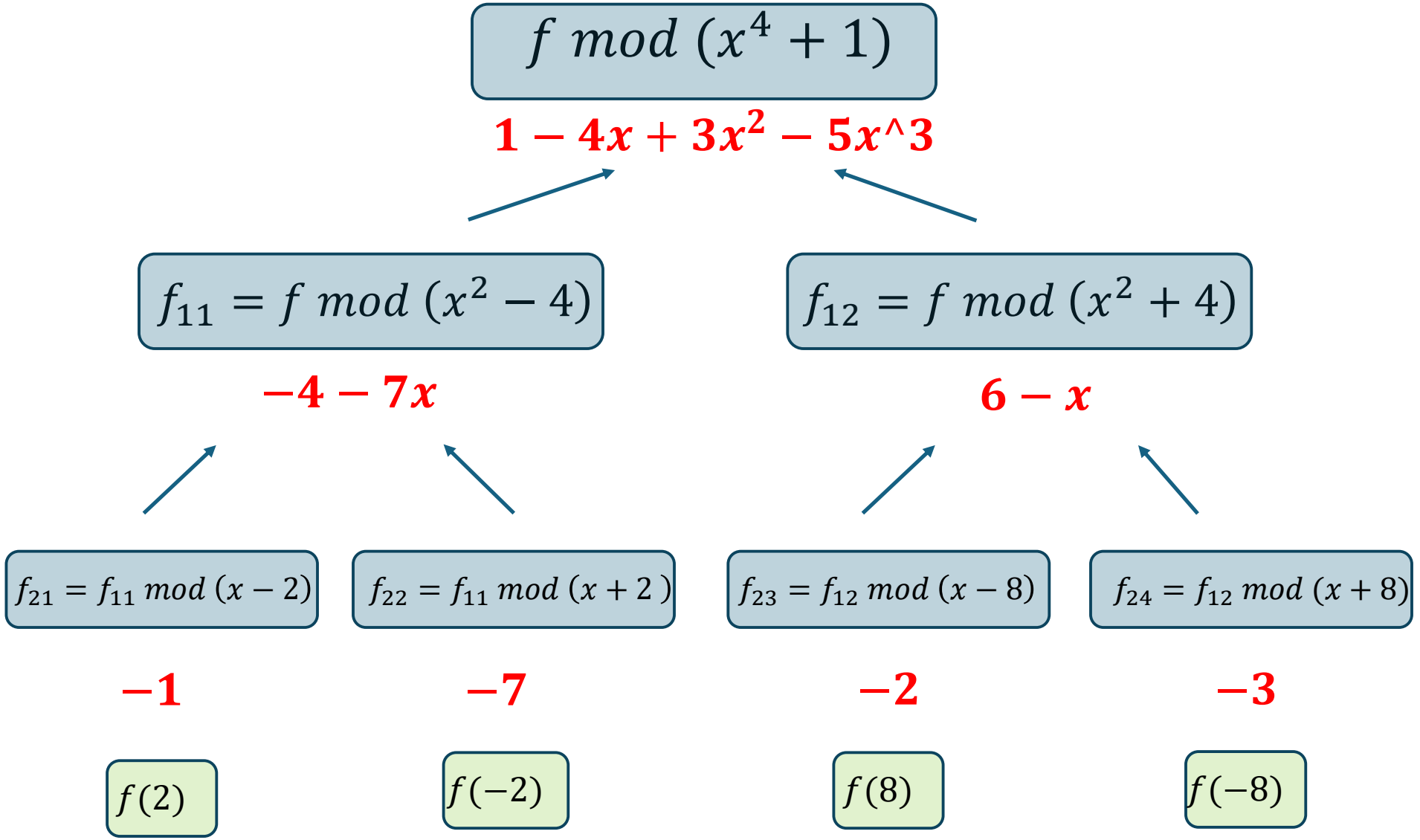
$f(2)$

$f(-2)$

$f(8)$

$f(-8)$

$$\hat{f} \rightarrow f$$



Computation time for $R=Z_p[x]/(x^n+1)$

$f \rightarrow \hat{f}$ and $\hat{f} \rightarrow f$

log n levels

(n/2 multiplications) and (n additions) modulo p per level

Implementation for n=256:

~ 6K cycles each without any optimizations (on a Haswell)

a 4X – 16X speedup using AVX2 instructions

(smaller p → larger speedup)

What encryption looks like

KeyGen:

$A \leftarrow R^{n \times n}$

$s, e \leftarrow \psi^n$

$t := As + e$

pk: (A, t)

sk: s

Encrypt(μ):

$r', e' \leftarrow \psi^n$

$f \leftarrow \psi$

$u' := r'A + e'$

$v := r't + f + [q/2]\mu$

ciphertext: (u', v)

Decrypt(u', v):

$w := v - u's$

$\mu := \text{Round}(w)$

The signature scheme uses similar operations

Larger Sizes but Faster

Scheme (128-bit security)	Public Key Size	Ciphertext Size
RSA-3072	384 Bytes	384 Bytes
ECDH-256	32 Bytes	32 Bytes
Kyber-512	**800 Bytes**	**768 Bytes**

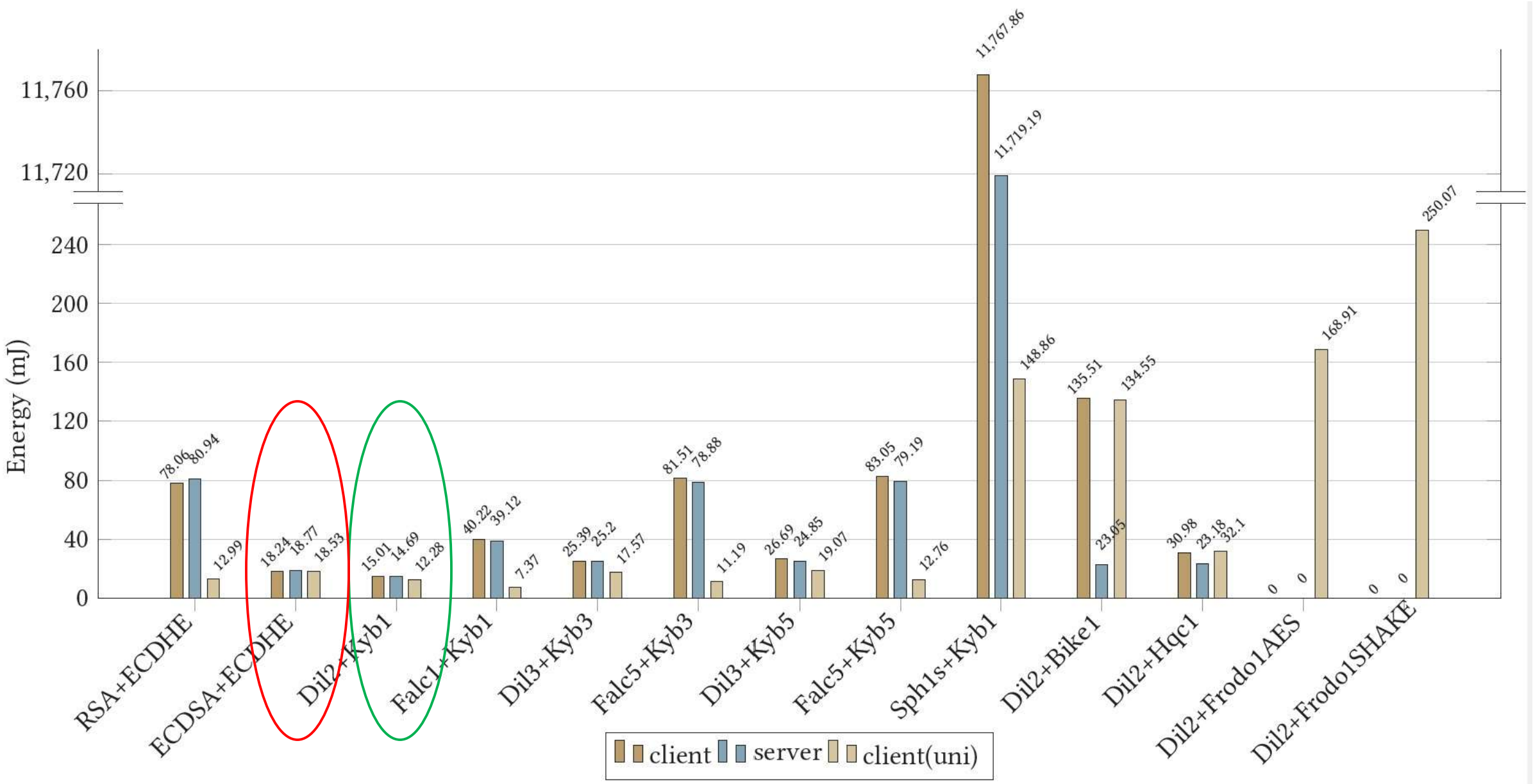


Figure 3: Average Energy Consumption (mJ) of Traditional and PQ TLS 1.3 handshake

Tasopoulos et. al, "Energy Consumption Evaluation of Post-Quantum TLS 1.3 for Resource-Constrained Embedded

Thank You