Post Quantum Crypto

Bernd Fix <brf@hoi-polloi.org>

„***Encryption works.*** Properly implemented strong crypto systems are one of the few things that you can rely on.“

Edward Snowden

Intro

Go to **https://cryptoparty.in** to find one near you...

Intro

The upcoming *cryptocalypse*:

- ***Most encrypted communication*** (like OpenPGP emails) and a lot of transient communication (with SSL/TLS) ***does not provide PFS*** („Perfect Forward Secrecy“).

- ***Most encrypted communication is stored long-term*** in datacenters around the world by secret agencies (*Bluffdale, Utah* is just one of them).

- ***Most public-key encryption schemes will be broken within the next ten years*** due to advancements in quantum computer technology.

Intro

Things we need to start doing right ***NOW*** :

- ***Only use PFS crypto schemes when communicating online***: Get rid of OpenPGP email and move to systems like ***Pond*** (https://pond.imperialviolet.org/). Fix the SSL/TLS settings on your own servers and/or kick ass with operators. Stop using services that don't care to comply.

- ***Design, implement and deploy new public-key crypto schemes that can not be broken by quantum computers***

Table of Contents

- Existing asymmetric key algorithms (public key cryptos)
- Attack vectors on public key cryptos
    - Classical approach
    - Quantum computing
- Quantum-resistent public key cryptos
    - Lattice-based crypto
    - Cryptos bases on encoding problems

RSA algorithm (1977)

$$m = p \cdot q$$

$$r := \varphi(m) = (p-1) \cdot (q-1)$$

r can only be computed with knowledge of (p,q)

$$\Rightarrow \quad g^{n \cdot r + 1} \equiv g \pmod{m} \equiv g^{d \cdot e}$$

Choose a public exponent e and *compute* a private exponent d:

$$d \cdot e \equiv 1 \pmod{r} \quad \Rightarrow \quad d = e^{-1} \pmod{r}$$

Public key: (e, m)

Private key: (d, m)

RSA algorithm (1977)

(DLP: Discrete Logarithm Problem)

- **Encryption**:

$$b = a^e \ mod \, m$$

- **Decryption**:

$$b^d \equiv a^{e \cdot d} \ mod \, m = a$$

- **Signature**:

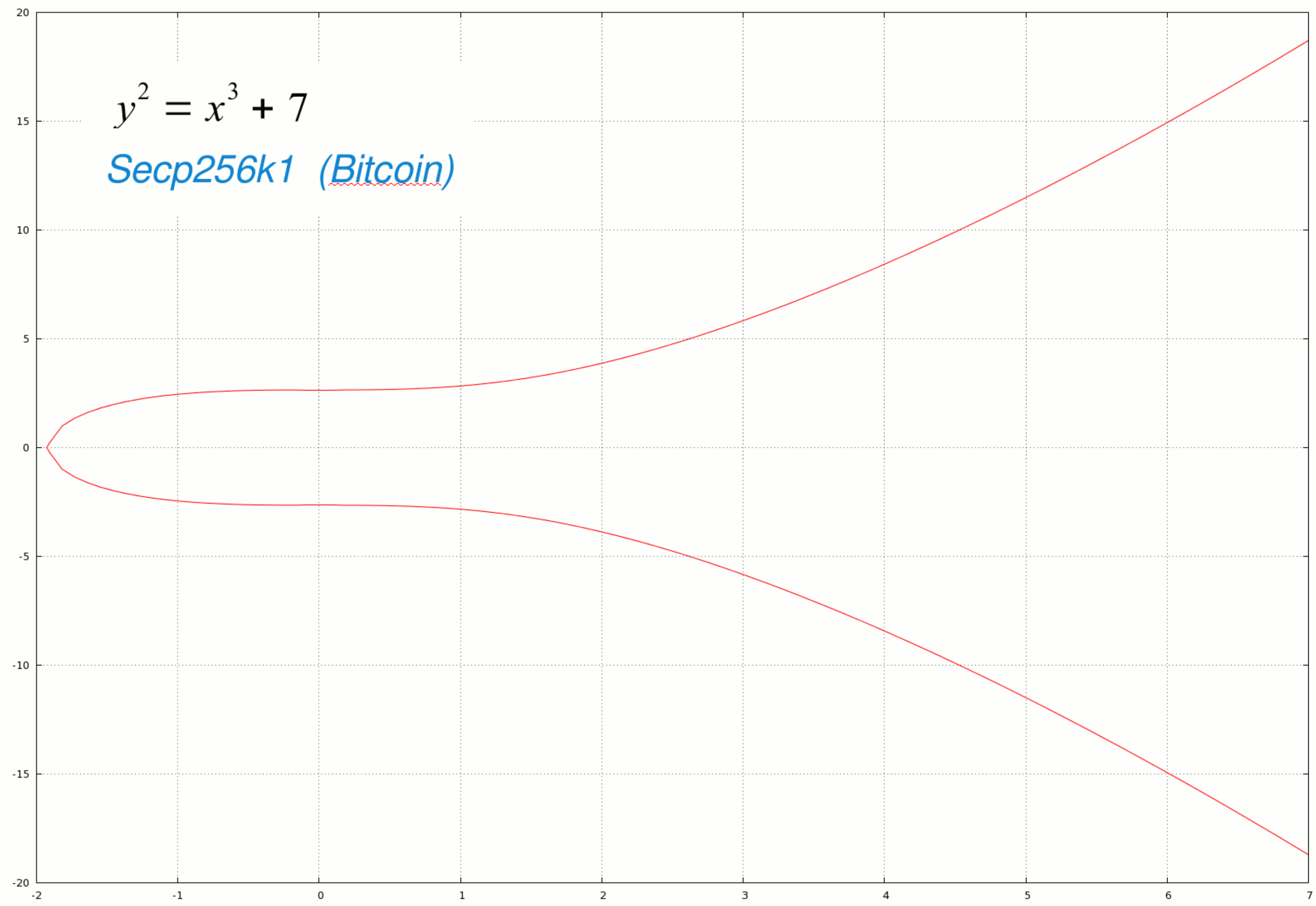
$$b = a^d \ mod \, m$$

- **Verification**:

$$b^e \equiv a^{d \cdot e} \ mod \, m = a$$

Elliptic Curve Crypto (1985)

$$y^2 = x^3 + a \cdot x + b \pmod{p}$$



Bernd Fix <brf@hoi-polloi.org>

Elliptic Curve Crypto (1985)

Generator point G forms an additive cyclic group $\langle G \rangle_{Fp}$ on curve

The order n of G on the curve is the smallest value with $n \cdot G = \infty$

$\Rightarrow$ all points on the curve have the form $P = a \cdot G$ with scalar $a \pmod{n}$

It is easy to compute $P = a \cdot G$,
but „infeasible" to compute a from P and G

(analog to DLP: Discrete Logarithm Problem, but much more difficult to solve than DLP over finite fields $\Rightarrow$ shorter keys)

Private key: d
Public key: $d \cdot G$

Elliptic Curve Crypto (1985)

Every DLP-based cryptosystem (DSA, ElGamal, DH) can be transformed into an ECC-based cryptosystem!

- **Signature / Verification:** ***ECDSA***
- **En-/Decryption:** ***ECDH***

DH (Diffie-Hellman)

- Parameter $\boldsymbol{g}, \boldsymbol{p}$
- Random secrets: $\boldsymbol{d}_A$ and $\boldsymbol{d}_B$
- Public: $e_X = g^{d_X} \ mod\ p$
- Shared: $s = e_A^{d_B} = e_B^{d_A} \ (mod\ p)$

ECDH

- Parameter $\boldsymbol{G}, \boldsymbol{n}$
- Random secrets: $\boldsymbol{d}_A$ and $\boldsymbol{d}_B$
- Public: $e_X = d_X \cdot G \quad mod\ n$
- Shared: $S = e_A \cdot d_B = e_B \cdot d_A \ (mod\ n)$

Bernd Fix <brf@hoi-polloi.org>

Attack vectors

Classical approach (number theory):

- **Discrete Logarithm Problem:** $a = b^e \ (mod\ m)$ [RSA]

  $P = a \cdot G \ (mod\ n)$ [ECC]

  Pollard-Rho algorithm, Baby-step giant-step

- **Integer Factorization:** $m = p \cdot q$ [RSA]

  All forms of quadratic sieves to find congruences $a^2 \equiv b^2 \ (mod\ m)$

$$p = (a + b), \quad q = (a - b)$$
$$\Rightarrow \quad m = p \cdot q = (a + b) \cdot (a - b) = a^2 - b^2$$
$$\Rightarrow \quad a^2 \equiv b^2 \quad (mod\ m)$$

Bernd Fix <brf@hoi-polloi.org>

Quantum computing (1994)

Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer*

Peter W. Shor†

Abstract

A digital computer is generally believed to be an efficient universal computing device; that is, it is believed able to simulate any physical computing device with an increase in computation time by at most a polynomial factor. This may not be true when quantum mechanics is taken into consideration. This paper considers factoring integers and finding discrete logarithms, two problems which are generally thought to be hard on a classical computer and which have been used as the basis of several proposed cryptosystems. Efficient randomized algorithms are given for these two problems on a hypothetical quantum computer. These algorithms take a number of steps polynomial in the input size, e.g., the number of digits of the integer to be factored.

Keywords: algorithmic number theory, prime factorization, discrete logarithms, Church's thesis, quantum computers, foundations of quantum mechanics, spin systems, Fourier transforms

AMS subject classifications: 81P10, 11Y05, 68Q10, 03D10

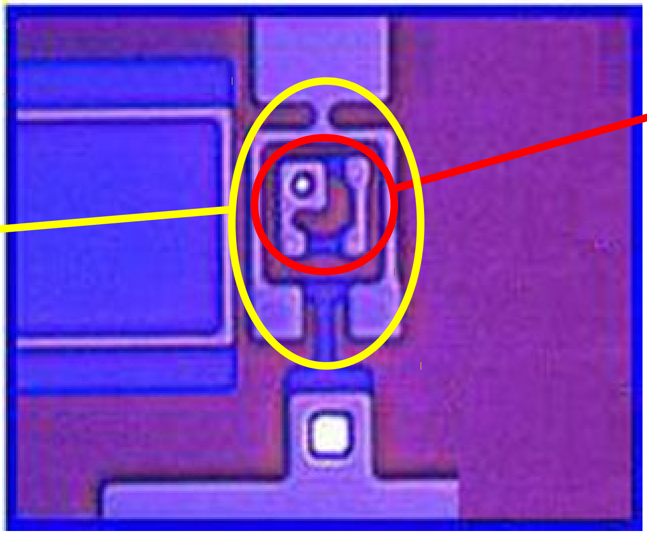
Quantum computers

Qubits:

- Two states in superposition: $\alpha\,|0\rangle + \beta\,|1\rangle \;=\; \alpha \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \beta \begin{pmatrix} 0 \\ 1 \end{pmatrix}$

- Realized with ion traps, NMR, **Josephson junctions**, photons, ...

SQUID
(used for read-out)



Two superconducting regions (loop) separated by a weak link (insulator)

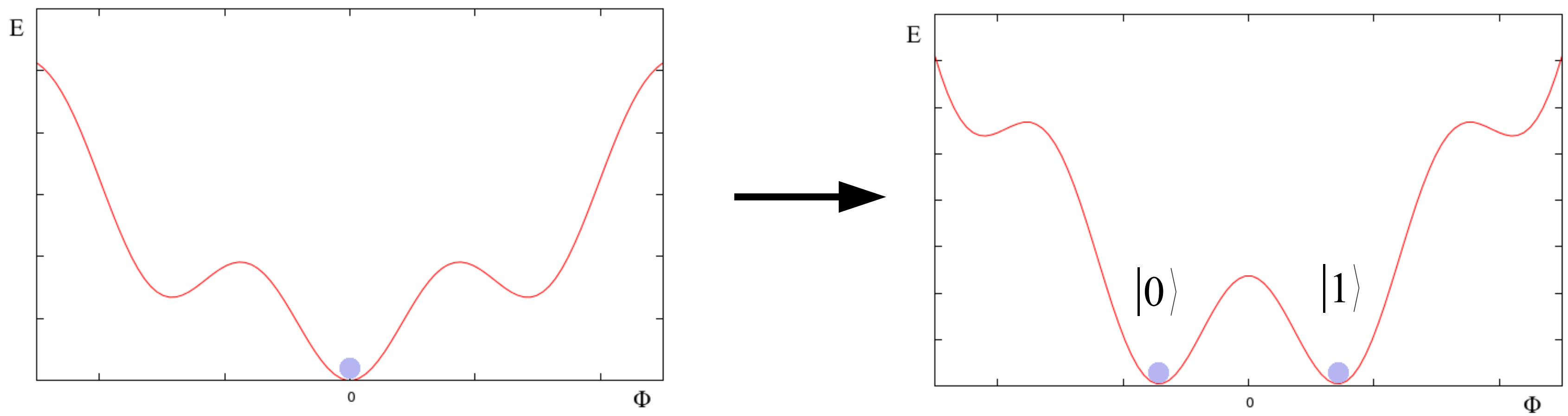
Source: en.wikipedia.org

Quantum computers

Qubits (Josephson junction):

- **Writing**: Apply a magnetic field, currents will flow in the loop

  Apply a *particular* magnetic field and the ground state is split into two states in superposition.



- **Reading**: Use a squid to measure the flows in the loop

Quantum computers

Quantum gates (doing computations):

Classic computers: NOT, AND, OR

(quantum computer: only reversible operations = unitary matrices)

NOT: $A = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$

C-NOT: $B = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$

CC-NOT: $C = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$

Sufficient to build a universal computer!

Bernd Fix <brf@hoi-polloi.org>

Quantum computers

Quantum gates (doing computations):

C-NOT:

$$N = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

C-SHIFT:

$$P = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{i\varphi} \end{pmatrix}$$

HADAMARD:

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

Composite gates:

DQT$_n$

U$_f$

...

Attack vectors

Quantum computing (Shor's algorithm):

Find a non-trivial solution for b such that $b^2 \equiv 1 \ (mod\ m)$

1. Pick a random $a < m$ with $gcd(a,m) = 1$
2. Find the period r of $f(x) = a^x \ mod\ m$ such that $f(x+r) = f(x)$
3. If r is odd or $a^{r/2} \equiv \pm 1 \ (mod\ m)$, go back to step 1
4. $b = a^{r/2}$ and $gcd(b \pm 1, m)$ is a non-trivial factor of n

Substitute „***factoring problem***“ with „***order-finding problem***“ which is more suitable for quantum computing

50% chance of finding a non-trivial factor for each pass

Bernd Fix <brf@hoi-polloi.org>

Attack vectors

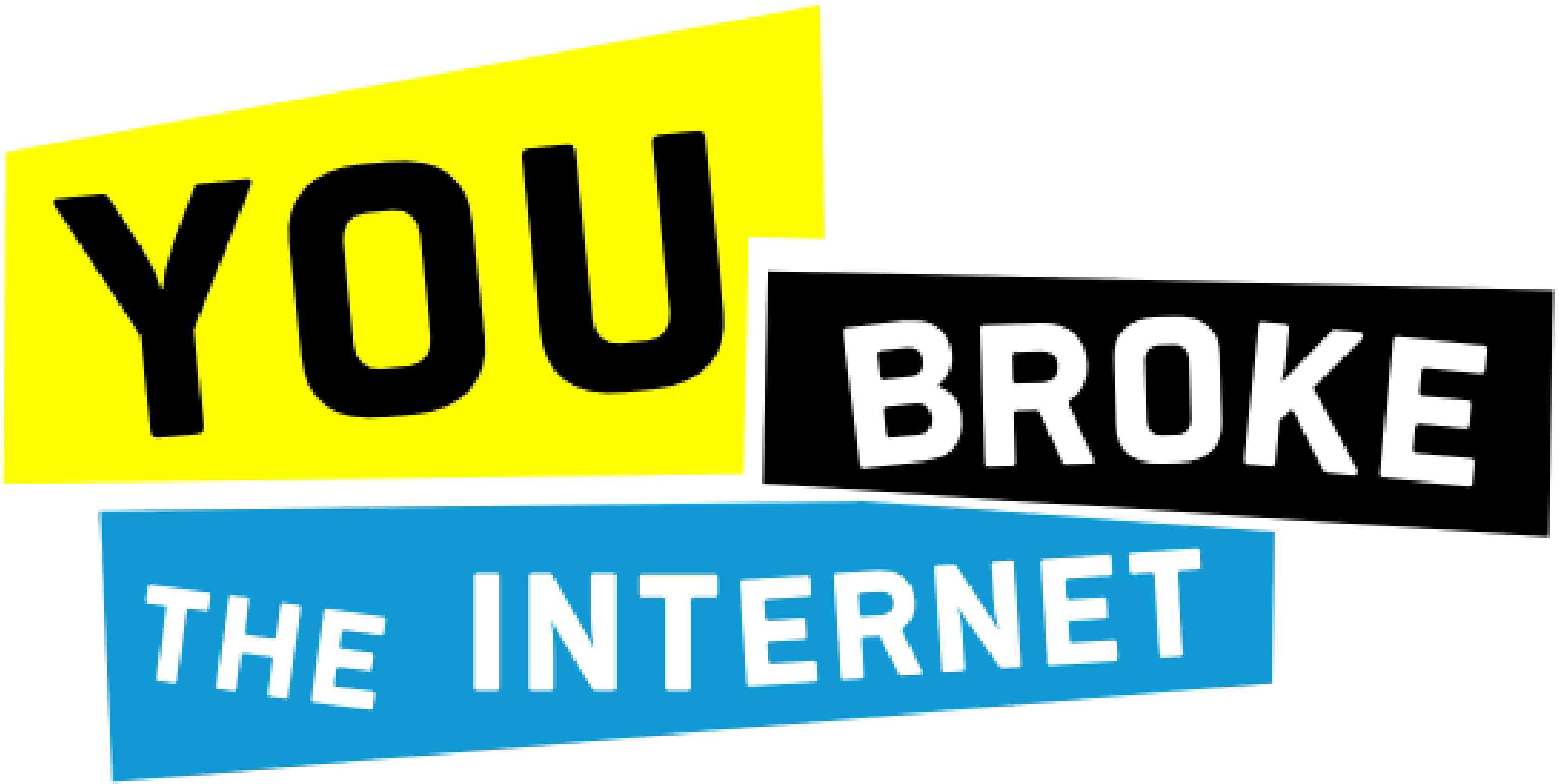
Quantum computing (Shor's algorithm):

1. Select q such that $m^2 \leq q\ (= 2^L) < 2m^2$
2. Prepare qubit register $|a\rangle$ of length L and initialize to state $|\mathbf{0}\rangle$
3. Prepare qubit register $|b\rangle$ of length $\lceil log_2\ m \rceil$ and initialize to state $|\mathbf{0}\rangle$
4. Create highest superposition of $|a\rangle$ by appying Hadamard gates
5. Apply (composite) U_f gate to $|a\rangle$ and $|b\rangle$: $|a,b\rangle \rightarrow |a, b \oplus f(a)\rangle$
6. Transform $|a\rangle$ into a different basis by a QFT (Quantum Fourier Transformation)
7. Observe $|a\rangle$ and compute the period r

Attack vectors

NIST ECC domain parameters (and others ?!) becoming ***fubar***

Thank you, stupid assholes!



We need new asymmetric key crypto:

- with resistence to quantum computer attacks
- developed as free software with no patents whatsoever
- with open peer review by crypto community
- „*do what you want, anything goes*“
  ignore commercial / govermental standardization
  promote community-agreed, decentralized „standards“

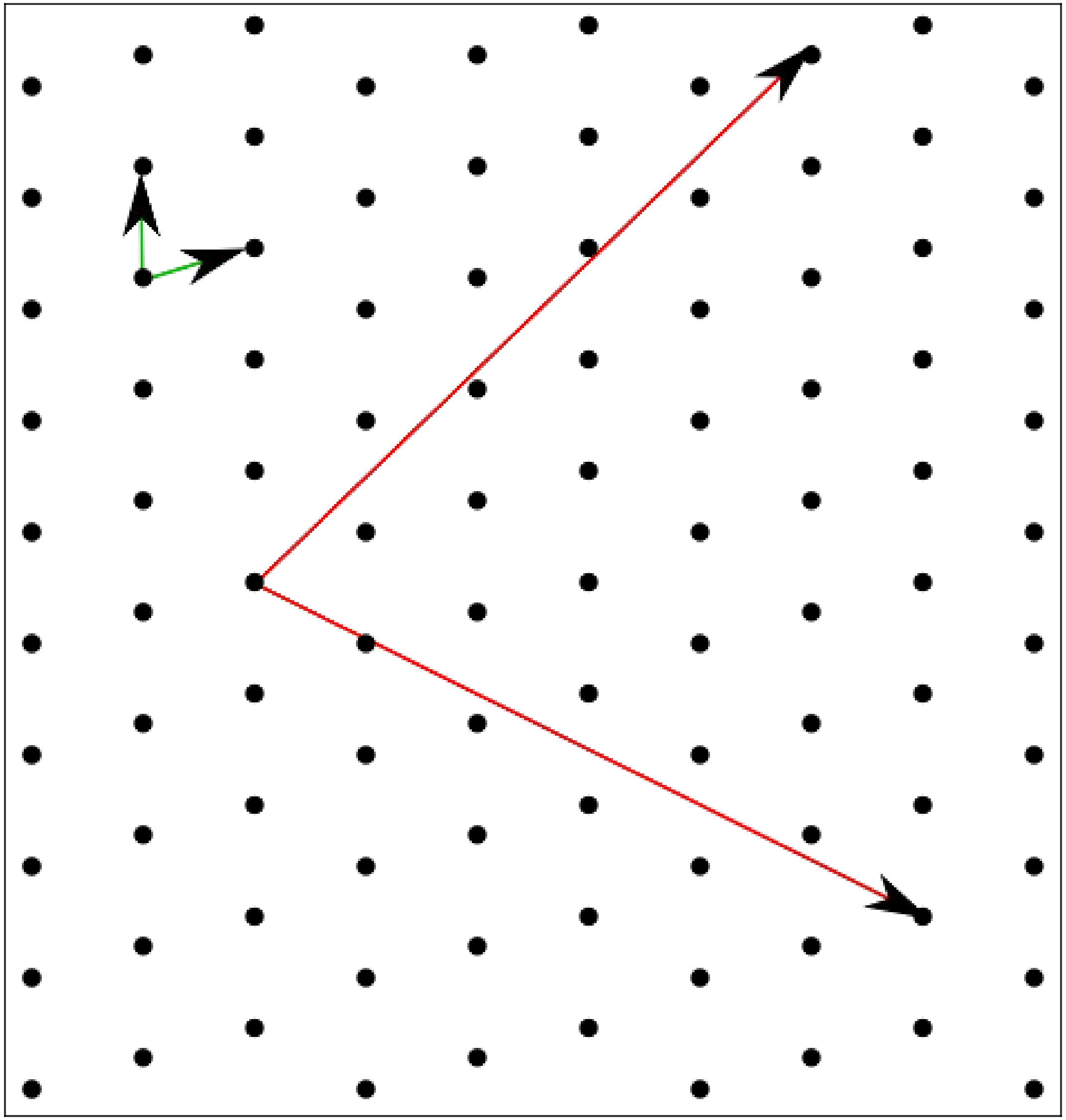
- **Lattice-based cryptography:** **nTru**, GGH
- Multivariate cryptography
- Hash-based signatures: Lamport-, Merkle-signatures
- **Code-based cryptography:** **McEliece enc.**, Niederreiter sigs

Post Quantum Crypto

Lattice-based crypto:

„good“ base

„bad“ base

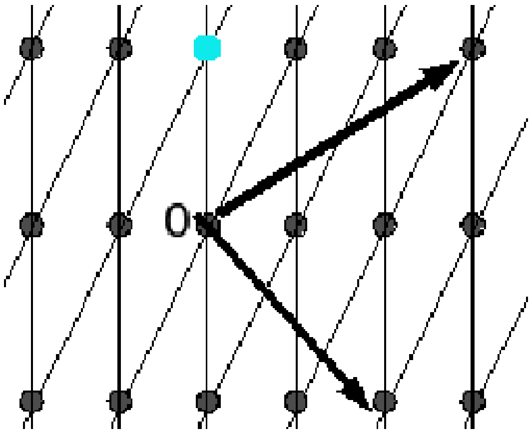


Find problems that are *easy* to solve with a *good base*, but are *very hard* to solve with a *bad base*...

Lattice-based crypto

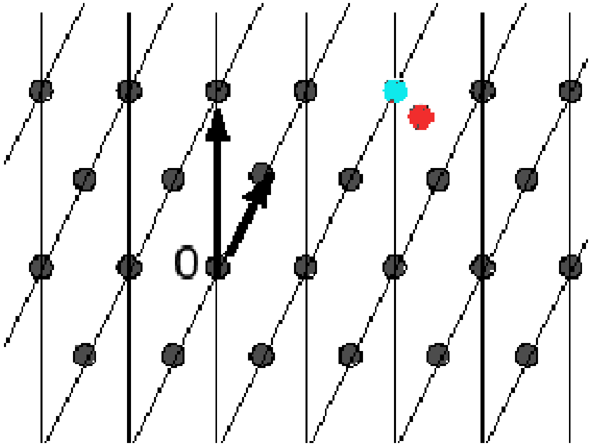
- **Shortest Vector Problem (SVP)**

  Find the shortest vector $v \in L$



- **Closest Vector Problem (CVP)**

  Find the vector $v \in L$ closest to a vector $w \notin L$



Source: en.wikipedia.org

Lattice-based crypto: nTru (https://github.com/NTRUOpenSourceProject/ntru-crypto)

- Based on objects in a truncated polynominal ring $\mathbb{Z}[X] / (X^N - 1)$:

$$a = a_0 + a_1 X + a_2 X^2 + a_2 X^2 + \cdots + a_{N-1} X^{N-1}$$

- Domain parameters (N, p, q) with N prime, $q > p$ and $p \perp q$

- **Key generation**: two polynominals f and g with $a_n \in \{-1,0,1\}$

  Private key: $(f,\ f^{-1} \bmod p)$

  Public key: $p \cdot (f^{-1} \bmod q) \cdot g \pmod q$

- **Encryption**: polynominals m, r results in $e = r \cdot h + m \pmod q$

- **Decryption**: $a = e \cdot f \pmod q,\ b = a \pmod p,\ m = (f^{-1} \bmod p) \cdot b$

Code-based cryptography: (McEliece encryption)

- Linear binary codes $[n,k,d]$ have length n, rank k and distance d
    1. Binary matrix G encodes blocks of k bits into blocks of n bits
    2. Minimal Hamming distance of rows (base vectors!) of G is d
    3. Efficient decoding algorithm to transform n bits back into k bits
    4. Matrix H detects t errors at any position in blocks of k bits

- _Example:_ Hamming code $[2^r, 2^r - r - 1, 3]$ with $r \geq 2$
- _Example:_ Hadamard code $[2^r, r, 2^{r-1}]$ with $r \geq 2$

Bernd Fix <brf@hoi-polloi.org>

Post Quantum Crypto

Code-based cryptography: **(McEliece encryption)**

- Key generation:

1. Construct a $k \times n$ binary matrix G that can correct t errors
2. Construct a random $k \times k$ invertible binary matrix S
3. Construct a random $n \times n$ permutation matrix P
4. Compute matrix $K = S \cdot G \cdot P$

Public key: (K, t)

Private key: (S, G, P)

Bernd Fix <brf@hoi-polloi.org>

Post Quantum Crypto

Code-based cryptography: (McEliece encryption)

- Encryption using public key (K, t):

    1. Construct a k-bit message m to be encrypted
    2. Compute n-bit encrypted message $e = m \cdot K$
    3. Construct a random n-bit vector r with t bits set
    4. Compute ciphertext $c = e \oplus t$

- Decryption using private key (S, G, P):

    1. Compute n-bit message $p = c \cdot P^{-1}$
    2. Decode n-bit message p into k-bit message d
    3. Compute k-bit plaintext message $m = p \cdot S^{-1}$

Bernd Fix <brf@hoi-polloi.org>

Post Quantum Crypto

Bernd Fix <brf@hoi-polloi.org>