

---



# Post-Quantum Cryptography



**Paulo S. L. M. Barreto**

LARC/PCS/EPUSP

---

# What is “Post-Quantum Cryptography”?

- To be sure, nobody really knows what PQC is...
- Reason: nobody knows exactly what quantum computers can do...
- ... and for that matter, nobody knows whether a full-fledged, reasonably scaled quantum computer can be built at all.





# What is “Post-Quantum Cryptography”?

- What is the point in discussing PQC?
- Well, nobody knows exactly what *classical* computers can do either (recall  $P$  vs.  $NP$ ), but classically secure cryptosystems are discussed nevertheless...
- ...and in practice cryptosystems are based on problems not known to be intractable.

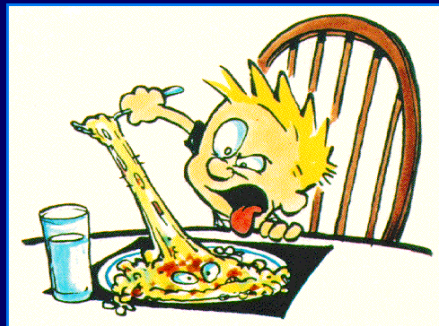


# Approach to PQC

- Develop cryptosystems based on provably intractable problems (at least on quantum computers...).
- Assess security and usability in practice.
- Quantify the quantum complexity of the intractability assumptions.

# A Preliminary Word

- This course is an introductory overview, and hence necessarily incomplete.
- The choice of topics is, to a large extent, a matter of personal taste (shame on me for unfulfilled expectations).





# Computational Problems

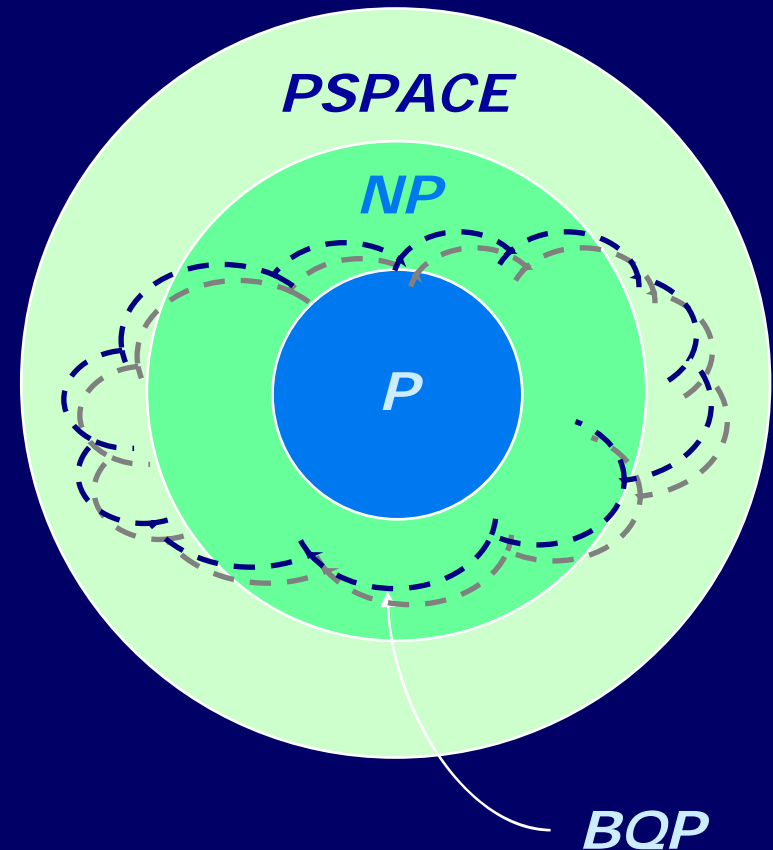
- $P$ : solvable in deterministic polynomial time.
- $NP$ : solvable in non-deterministic polynomial time.
- $PSPACE$ : solvable in polynomial space.
- $BQP$ : solvable by a quantum computer with polynomial quantum gate count.

# Computational Problems

- What one knows:

- $P \subseteq NP \subseteq PSPACE$
- $P \subseteq BQP \subseteq PSPACE$

- The exact relationships among the classes are not known.





# Deployed Cryptosystems

- Integer Factorization – IFP: RSA.
- Discrete Logarithm and Diffie-Hellman – DLP, CDHP, DDHP, BDHP, ...: (EC)DSA, (EC)DH, pairing-based cryptosystems.
- These intractability assumptions (and several others) reduce to that of the *Hidden Subgroup Problem* – HSP.



# Hidden Subgroup Problem

- Let  $G$  be a group,  $H \leq G$ , and  $f$  a function on  $G$ . We say that  $f$  separates cosets of  $H$  if  $f(u) = f(v) \Leftrightarrow uH = vH, \forall u, v \in G$ .
- Hidden Subgroup Problem (HSP):
  - Let  $\mathcal{A}$  be an oracle to compute a function that separates cosets of some subgroup  $H \subset G$ . Find a generating set for  $H$  using information gained from  $\mathcal{A}$ .
- Important special cases:
  - Hidden Abelian Subgroup Problem (HASP)
  - Hidden Dihedral Subgroup Problem (HDSP)

# Quantum Computing

- Quantum algorithms can solve particular cases of the HASP (including IFP and DLP) in random polynomial time.



# Solutions?

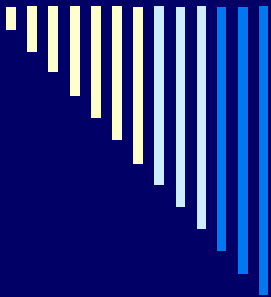
- Purely quantum cryptography:
  - What schemes are available?
  - Migration?
- Classical alternatives:
  - If  $BQP \subset NP$ , systems based on problems not in  $BQP$  (“post-quantum”).
  - Otherwise...





# Proposed Post-Quantum Cryptosystems

- Non-Abelian Groups
  - Conjugacy equivalents of DL schemes.
- Lattice Reduction
  - Ajtai-Dwork, GGH, Regev, NTRU, ...
- Syndrome Decoding
  - McEliece, Niederreiter, CFS, ...
- Multivariate Quadratic Systems
  - HFE, SFLASH, ...
- Other systems:
  - Permuted Kernels and Perceptrons, Constrained Linear Equations, Merkle signatures...



# Non-Abelian Groups



# Non-Abelian Groups

- Word Problem (WP): given a finitely generated group  $G = \langle g_1, \dots, g_n \rangle$  and two words  $p$  and  $q$  in the generators of  $G$  (i.e.  $p = \prod_{i \in I} g_i$  and  $q = \prod_{j \in J} g_j$ , where  $I$  and  $J$  are sequences of indices from  $[1..n]$ ), decide whether  $p$  and  $q$  stand for the same element of  $G$ .
- WP is in general undecidable.



# Non-Abelian Groups

- There exist groups where WP is solvable, e.g. automatic groups:
  - Finite groups.
  - Braid groups (Birman-Ko-Lee 1998).
  - Hyperbolic groups (Bridson-Howie 2003).
  - Polycyclic groups (Eick-Kahrobaei 2004).
  - Euclidean groups.
  - Coxeter groups.
- Conjugacy Decision Problem (CDP):
  - Given  $p, q \in G$ , decide whether there exists  $s \in G$  such that  $p = s^{-1}qs$  (if so, we write  $p \sim q$ ).
- CDP is (usually) solvable if WP is solvable.



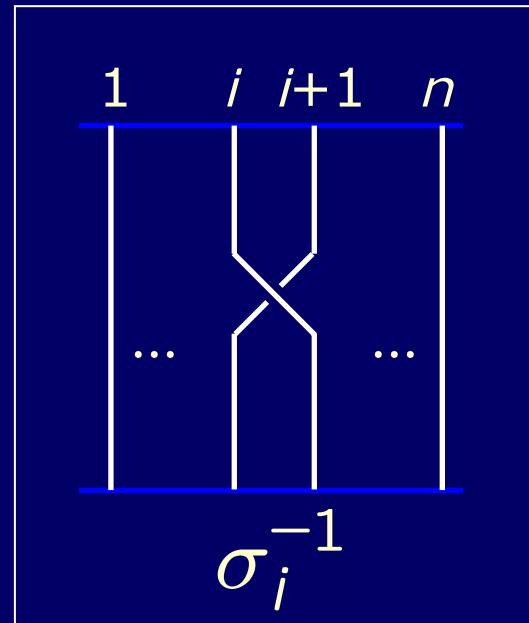
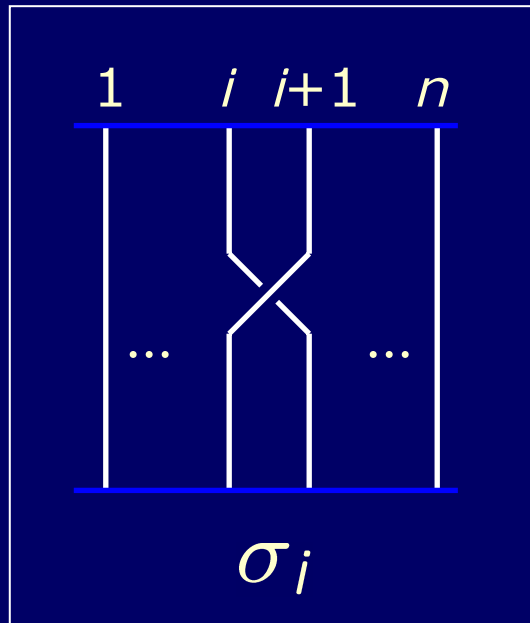
# Braid Groups

$$B_n = \left\langle \sigma_1, \dots, \sigma_{n-1} \mid \begin{array}{l} \sigma_i \sigma_j \sigma_i = \sigma_j \sigma_i \sigma_j \quad \text{for } |i - j| = 1 \\ \sigma_i \sigma_j = \sigma_j \sigma_i \quad \text{otherwise} \end{array} \right\rangle$$

- Though non-Abelian, braid groups contain large commuting subgroups.
- Protocols designed for braid groups often remain interesting when instantiated over other groups where certain problems related to the CDP are intractable.

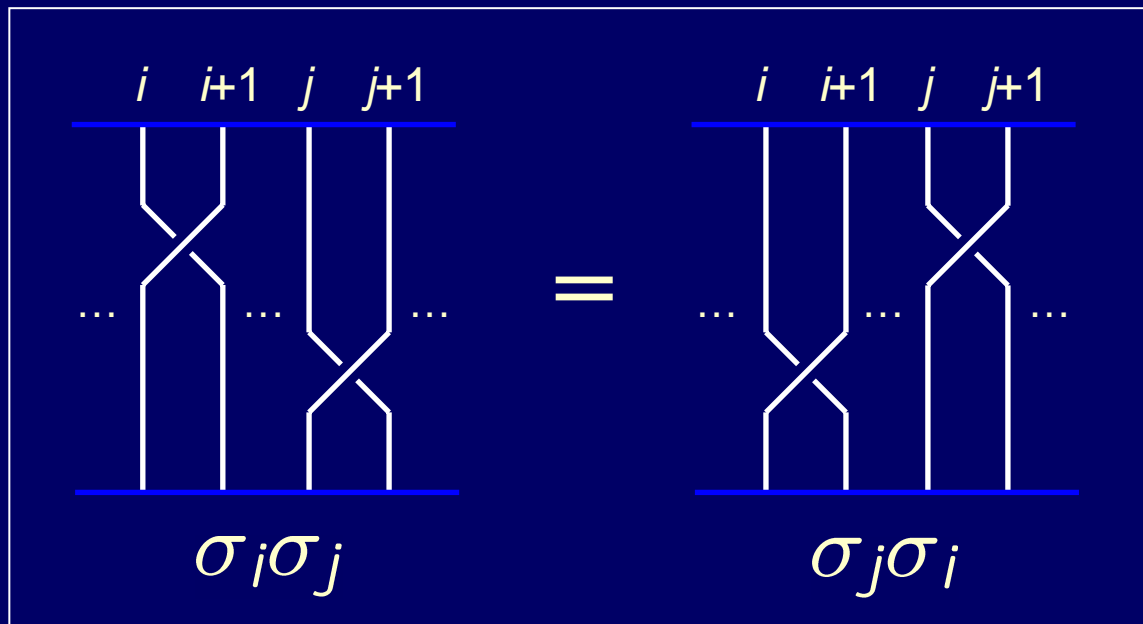
# Braid Groups

$$B_n = \left\langle \sigma_1, \dots, \sigma_{n-1} \mid \begin{array}{l} \sigma_i \sigma_j \sigma_i = \sigma_j \sigma_i \sigma_j \text{ for } |i - j| = 1 \\ \sigma_i \sigma_j = \sigma_j \sigma_i \text{ otherwise} \end{array} \right\rangle$$



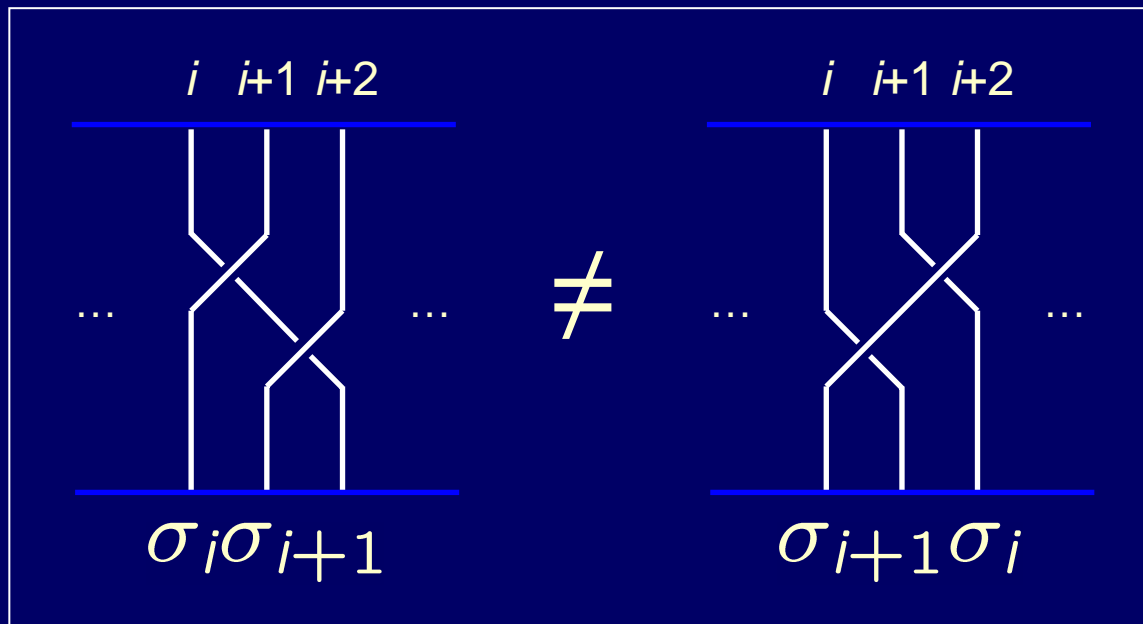
# Braid Groups

$$B_n = \left\langle \sigma_1, \dots, \sigma_{n-1} \mid \begin{array}{l} \sigma_i \sigma_j \sigma_i = \sigma_j \sigma_i \sigma_j \text{ for } |i - j| = 1 \\ \sigma_i \sigma_j = \sigma_j \sigma_i \text{ otherwise} \end{array} \right\rangle$$



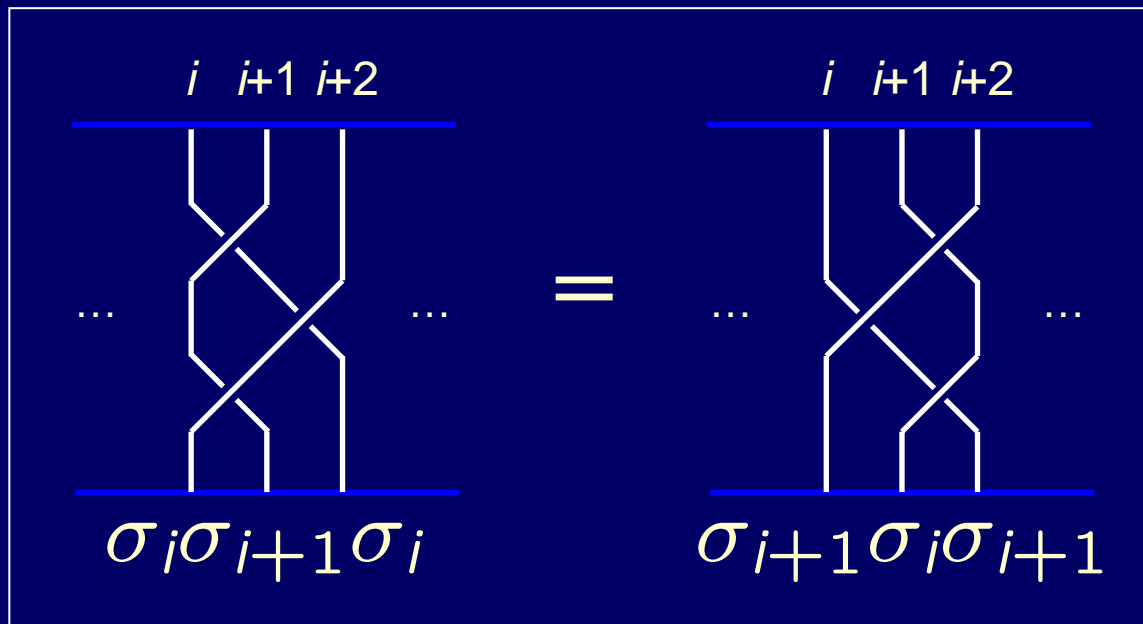
# Braid Groups

$$B_n = \left\langle \sigma_1, \dots, \sigma_{n-1} \mid \begin{array}{l} \sigma_i \sigma_j \sigma_i = \sigma_j \sigma_i \sigma_j \text{ for } |i - j| = 1 \\ \sigma_i \sigma_j = \sigma_j \sigma_i \text{ otherwise} \end{array} \right\rangle$$



# Braid Groups

$$B_n = \left\langle \sigma_1, \dots, \sigma_{n-1} \mid \begin{array}{l} \sigma_i \sigma_j \sigma_i = \sigma_j \sigma_i \sigma_j \text{ for } |i - j| = 1 \\ \sigma_i \sigma_j = \sigma_j \sigma_i \text{ otherwise} \end{array} \right\rangle$$





# Braid Groups

- Birman-Ko-Lee normal form of  $B_n$  braids:  $p = (u, \pi_1, \dots, \pi_k)$ , where  $u \in \mathbb{Z}$ , and the  $\pi_i$  are permutations over  $[1..n]$ . We say that  $k$  is the length of the braid  $p$  and write  $k = \text{length}(p)$ .
- Cost:
  - Normalization:  $O(k^2 n \log n)$ .
  - Storage:  $O(kn \log n)$  bits per braid.
  - Group operations:  $O(kn)$ .
  - Cryptographic operations: usually  $O(k^2 n \log n)$ .



# Computational Problems on Non-Abelian Groups

- Membership Decision Problem (MDP):
  - Given  $x, u_1, \dots, u_r \in G$ , decide whether  $x$  can be written as a word in  $u_j$ .
- Membership Search Problem (MSP):
  - Given  $x, u_1, \dots, u_r \in G$ , find a sequence  $I$  of indices  $[1..r]$  such that  $x = \prod_{j \in I} u_j$ .
- Often the MDP is not only intractable but *undecidable* (e.g. on a  $B_n$  with  $n \leq 6$ ).



# Computational Problems on Non-Abelian Groups

## □ Conjugacy Search Problem (CSP):

- Given  $p, q \in G$  with  $p \sim q$ , find  $s \in G$  such that  $p = s^{-1}qs$ .

## □ Matching Conjugate Search Problem (MCSP):

- Given  $p, p', q \in G$  with  $p \sim p'$ , find  $q' \in G$  such that  $p'q' \sim pq$ .

## □ Multiple Simultaneous Conjugacy Search Problem (MSCSP):

- Given  $2m \geq 1$  elements  $v_i, w_i \in G$  satisfying  $w_i = s^{-1}v_i s$  ( $1 \leq i \leq m$ ) for the same (unknown)  $s \in G$ , find  $x \in G$  such that  $w_i = x^{-1}v_i x$  ( $1 \leq i \leq m$ ).



# Computational Problems on Non-Abelian Groups

- Subgroup Conjugacy Search Problem (SCSP):
  - Given a subgroup  $A \leq G$  and  $p, q \in G$ , find  $s \in A$  such that  $p = s^{-1}qs$  (if any exists).
  
- Commutator Diffie-Hellman Conjugacy Problem (CDHCP):
  - Let  $L$  and  $R$  be *commuting* subgroups of  $G$ , Given  $a, u = x^{-1}ax, v = y^{-1}ay \in G$  for some (unknown)  $x \in L, y \in R$ , compute  $x^{-1}y^{-1}axy$ .



# Anshel-Anshel-Goldfeld Key Agreement

- Let  $f: G \times G \rightarrow G$  be a function satisfying  $f(x, y_1 y_2) = f(x, y_1) f(x, y_2)$ .
  - Example:  $f(x, y) = x^{-1} y x$ , since  $f(x, y_1 y_2) = x^{-1} y_1 y_2 x = x^{-1} y_1 x x^{-1} y_2 x = f(x, y_1) f(x, y_2)$ .
  
- Decomposition: if  $y = \prod_{j \in J} g_j$ , then  $f(x, y) = f(x, \prod_{j \in J} g_j) = \prod_{j \in J} f(x, g_j)$ .



# Anshel-Anshel-Goldfeld Key Agreement

- Commutator:  $[x, y] \equiv x^{-1}y^{-1}xy$ .
- Let  $h_1, h_2: G \times G \rightarrow G$  be functions satisfying  $h_1(x, f(y, x)) = h_2(y, f(x, y)) = [x, y]$ .
  - Example: for  $f(x, y) = y^{-1}xy$ , take  $h_1(x, y) = x^{-1}y$  and  $h_2(x, y) = y^{-1}x = h_1(y, x)$ .
  - $h_1(x, f(y, x)) = h_1(x, y^{-1}xy) = x^{-1}y^{-1}xy$ .
  - $h_2(y, f(x, y)) = h_2(y, x^{-1}yx) = x^{-1}y^{-1}xy$ .



# Anshel-Anshel-Goldfeld Key Agreement

Alice:

- $\langle a_1, \dots, a_A \rangle \leq_R G,$   
 $I \in_R [1..A]^*;$
- $x \leftarrow \prod_{i \in I} a_i.$
- $(f(x, b_j) \mid 1 \leq j \leq B)$

Bob:

- $\langle b_1, \dots, b_B \rangle \leq_R G,$   
 $J \in_R [1..B]^*;$
- $y \leftarrow \prod_{j \in J} b_j.$
- $(f(y, a_i) \mid 1 \leq i \leq A)$

- $f(y, x) \leftarrow \prod_{i \in I} f(y, a_i),$   
 $K \leftarrow h_1(x, f(y, x)).$

- $f(x, y) \leftarrow \prod_{j \in J} f(x, b_j),$   
 $K \leftarrow h_2(y, f(x, y)).$



# KLCHKP Key Agreement

- Choose  $G$  and two disjoint, commuting subgroups  $LG, RG \leq G$ .
  - Example: braid group  $G = \langle \sigma_1, \dots, \sigma_{2n-1} \rangle$ ;
  - $LG = \langle \sigma_1, \dots, \sigma_{n-1} \rangle$ ,  $RG = \langle \sigma_{n+1}, \dots, \sigma_{2n-1} \rangle$ .
  
- Choose a “sufficiently complicated” public word  $g \in_R G$ .



# KLCHKP Key Agreement

Alice:

- $s_A \in_R \text{LG};$

- $v_A \leftarrow s_A^{-1} g s_A.$

- $K \leftarrow s_A^{-1} v_B s_A =$   
 $s_A^{-1} s_B^{-1} g s_A s_B.$

Bob:

- $s_B \in_R \text{RG};$

- $v_B \leftarrow s_B^{-1} g s_B.$

- $K \leftarrow s_B^{-1} v_A s_B =$   
 $s_A^{-1} s_B^{-1} g s_A s_B.$

- Easy to generalize to ElGamal-like encryption.



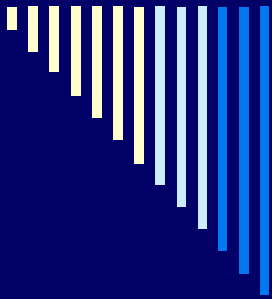
# KCCL Signatures

- Random oracle  $H: \{0, 1\}^* \rightarrow G$ .
- Key generation:
  - Choose  $p, s \in G$  at random and compute  $p' \leftarrow s^{-1}ps$ .
  - $K_{\text{private}} = s; K_{\text{public}} = (p, p')$ .
- Signing a message  $m$ :
  - Hash  $m$  to  $q \leftarrow H(m) \in G$ .
  - Compute the signature  $q' \leftarrow s^{-1}qs$ .
- Verifying a signature  $q'$  for message  $m$ :
  - Validate the public key by checking  $p \sim p'$ .
  - Hash  $m$  to  $q = H(m) \in G$ .
  - Accept the signature iff  $pq \sim p'q'$ .



# Security Assessment

- The CDHCP over  $B_n$  can be solved in time  $O(n^{4\lg 7 + 2\lg 3} k^{2\lg 3}) \approx O(n^{14.4} k^{3.2})$  (Cheon-Jun 2003).
- The security of AAG and variants (Anshel-Anshel-Goldfeld 2001, Lee-Lee 2002, Hughes 2002) depends on the SCSP rather than the CSP (Shpilrain-Ushakov 2004).
- Heuristic attacks based on the MSCSP succeed with high probability against AAG for certain parameters (Hofheinz-Steinwandt 2003).
- Length-based attacks are also possible against certain instances of conjugacy problems on braid groups (Myasnikov-Ushakov 2007).



# Permuted Kernels and Perceptrons



# Permuted Kernels

- The kernel of a matrix  $A$  mod  $p$  is the vector set  $\ker(A) = \{V \mid AV = 0 \pmod{p}\}$ .
- Vector permutation: if  $V = (V_i \in \mathbb{Z}_p \mid i \in [1..n])$ , then  $\pi(V) = (V_{\pi(i)} \mid i \in [1..n])$ .
- Permuted Kernels Problem (PKP): given an  $m \times n$  matrix  $A$ , an  $n$ -vector  $V$ , and a prime  $p$ , find a permutation  $\pi \in P_n$  such that  $\pi(V) \in \ker(A)$ .
- Identification scheme (Shamir 1989): robust for small parameters ( $p = 251$ ,  $m = 15$ ,  $n = 32$ ), well-suited for 8-bit processors.



# Permuted Kernels

## □ Setup:

- Choose a random  $m \times n$  matrix  $A \pmod{p}$ .
- Random oracle  $H$  over  $P_n \times (\mathbb{Z}_p)^n$ .

## □ Key pair:

- Choose a random permutation  $\pi$  and obtain a random vector  $V \in (\mathbb{Z}_p)^n$  such that  $\pi(V) \in \ker(A)$ .
- $K_{\text{private}} = \pi$ ;  $K_{\text{public}} = V$ .



# Permuted Kernels

- The prover chooses a random vector  $R \in (\mathbb{Z}_p)^n$  and a random permutation  $\sigma \in P_n$ , and sends  $h_\sigma \leftarrow H(\sigma, AR)$ ,  $h_{\pi\sigma} \leftarrow H(\pi\sigma, \sigma(R))$  to the verifier.
- The verifier chooses a random  $c \pmod{p}$  and asks the prover to send  $W = \sigma(R) + c\pi\sigma(V)$ .
- After receiving  $W$ , the verifier challenges the prover to reveal either  $\sigma$  or  $\pi\sigma$ . In the former case the verifier checks that  $H(\sigma, \sigma(A)W) = h_\sigma$ , in the latter case the verifier checks that  $H(\pi\sigma, W - c\pi\sigma(V)) = h_{\pi\sigma}$ .



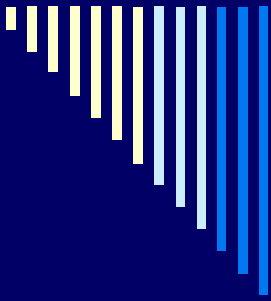
# Permuted Perceptrons

- Perceptrons Problem (PP): given an  $m \times n$  matrix  $A$  with  $A_{ij} = \pm 1$ , find an  $n$ -vector  $V$  with  $V_j = \pm 1$  such that  $(AV)_i \geq 0$ .
- Permuted Perceptrons Problem (PPP): given an  $m \times n$  matrix  $A$  with  $A_{ij} = \pm 1$  and an  $m$ -vector  $S$  with  $S_k \geq 0$ , find an  $n$ -vector  $V$  with  $V_j = \pm 1$  such that  $AV$  is a permutation of  $S$ .



# Permuted Perceptrons

- Identification scheme (Pointcheval 1995).
- Simulated annealing attack (Knudsen-Meier 1999): Cost of solving PPP for the smallest proposed parameters ( $m = 101$ ,  $n = 117$ ) estimated at  $2^{64}$ , refined to  $2^{56}$ , sometimes much less (as low as  $2^{31}$ ).
- For larger parameters the scheme is less competitive compared to other protocols.



# Multivariate Quadratic Systems



# Multivariate Quadratic Systems

- Multivariate Quadratic System Problem (MQSP): Let  $n \in \mathbb{N}$ , and for  $1 \leq i \leq n$  let  $v_i \in \mathbb{F}_q$  and  $g_i \in \mathbb{F}_q[X_1, \dots, X_n]$  of degree 2. Find  $x_1, \dots, x_n \in \mathbb{F}_q$  such that  $g_i(x_1, \dots, x_n) = v_i$ , for all  $1 \leq i \leq n$ .
- MQSP is *NP*-hard.
- Usual solution involves computing the Gröbner basis for the system.



# Multivariate Quadratic Systems

- MQSP cryptosystems disguise a solvable instance as the  $g_i$  polynomials, and use the disguise transform as private key.
- Let  $\sigma$  denote the Frobenius map in  $\mathbb{F}_{q^n}$ . Let  $r \in \mathbb{N}$  with  $r < n$  and  $f \in \mathbb{F}_{q^n}[X_0, \dots, X_r]$  of degree 2, and  $m \in \mathbb{F}_{q^n}$ .
- The equation  $f(\sigma^0(x), \sigma^1(x), \dots, \sigma^r(x)) = m$  can be efficiently solved (von zur Gathen and Shoup 1992).



# Multivariate Quadratic Systems

- $(\omega_1, \dots, \omega_n)$ : basis of  $\mathbb{F}_{q^n}$  interpreted as a vector space over  $\mathbb{F}_q$ .
- For  $x = \sum_i x_i \omega_i$ , the equation  $f(\sigma^0(x), \dots, \sigma^r(x)) = m$  can be written as a quadratic system  $f_i(x_1, \dots, x_n) = m_i$  over  $\mathbb{F}_q$ .
- If this system is solvable, for any affine mappings  $s, t$  on  $(\mathbb{F}_q)^n$  and  $g_i = (t \circ f_i \circ s)$  the system  $g_i(x_1, \dots, x_n) = t(m_i)$  is also solvable.



# Multivariate Quadratic Systems

## □ Hidden Field Equations (HFE):

- $K_{\text{private}} = (s, t, f_1, \dots, f_n).$

- $K_{\text{public}} = (g_1, \dots, g_n)$  with  $g_i = (t \circ f_i \circ s).$

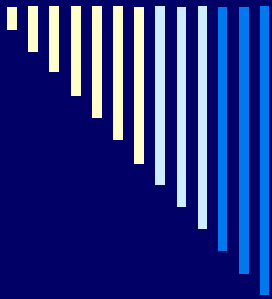
□ Faugère-Joux: if  $q = 2$  or  $q^{r-1}(q+1) \leq 512$ , HFE over  $\mathbb{F}_q$  can be broken in  $O(n^{10})$  steps with a Gröbner basis attack.



# Multivariate Quadratic Systems

## □ SFLASH:

- Selected by the NESSIE project.
  - Security based on a variant of HFE called C\* or Matsumoto-Imai.
  - Exponentiation in  $\mathbb{F}_{qn}$ .
- All C\* schemes can be broken in time  $O(n^6 \log^2 q)$ .
- One second or less in the case of SFLASH! (Dubois et al. 2007).



# Lattice Reduction



# Lattice Reduction

- Lattice: set of intersection points of an infinite and regular (but not necessarily orthogonal)  $n$ -dimensional grid.
- Formally: the lattice generated by  $n$  linearly independent vectors  $v_1, \dots, v_n \in \mathbb{R}^n$  is the set

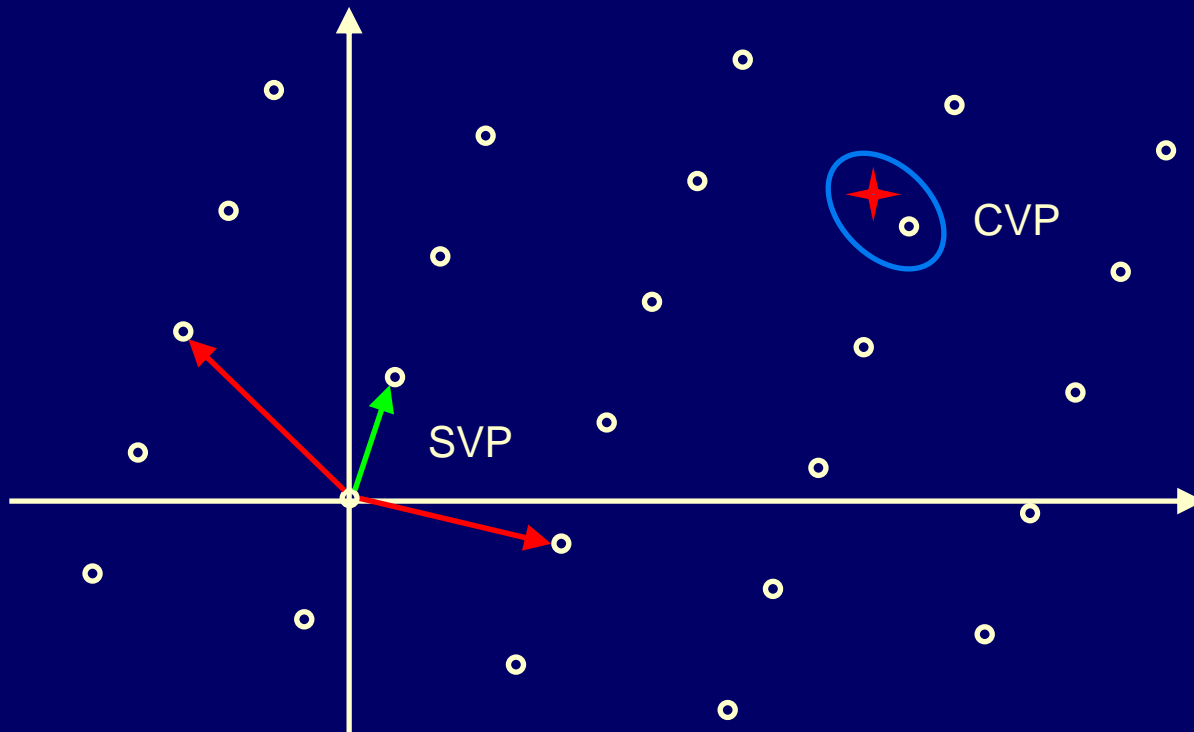
$$L(v_1, \dots, v_n) = \left\{ \sum_{i=1}^n \alpha_i v_i \mid \alpha_i \in \mathbb{Z} \right\}.$$



# Lattice Reduction

- *Shortest Vector Problem (SVP)*: find the shortest nonzero vector in a given lattice.
  - unique-SVP: particular configuration of SVP where the shortest vector is unique.
  - SIVP: find a set of shortest linearly independent vectors.
- *Closest Vector Problem (CVP)*: given a lattice and a target point (not necessarily in the lattice), find the lattice point closest to the target.

# Lattice Reduction





# Lattice Reduction

- All problems are equally intractable.
  - CVP is *NP*-hard (van Emde Boas 1981).
  - SVP is not harder than CVP (Micciancio *et al.* 1999).
  - SVP is *NP*-hard to solve even approximately (Micciancio 1998).
  - SIVP is not harder than CVP, and can be solved in  $O(n!)$  steps (Micciancio 2007).



# Ajtai-Dwork Cryptosystem

- First cryptosystem with coincident worst-case security and average-case security.
- Security based on unique-SVP.
- Decryption errors (fixed by Goldreich, Goldwasser and Halevi).
- $O(n^4)$  key storage,  $O(n^4)$  operation time.



# Ajtai-Dwork Cryptosystem

## □ Nguyen-Stern (1998):

- Heuristic attack based on the LLL algorithm.
- Secure instances of Ajtai-Dwork must use large keys (fine for PQC).

## □ Improvements:

- BKZ: better approximation than LLL to the shortest vector.
- Schnorr: random sampling heuristics (can be further improved with quantum computers!).



# Goldreich-Goldwasser-Halevi Cryptosystem

- Based on the CVP rather than the unique-SVP.
- Similar to the McEliece cryptosystem!
- Security parameters:  $\sigma, n \gg \sigma$  (typically  $\sigma \approx 3, n \approx 300$ ).
- Key size: typically  $O(n^2)$  bits, but can be as large as  $O(n^3 \log n)$ .
- Operation time:  $O(n^2)$ .



# Goldreich-Goldwasser-Halevi Cryptosystem

## □ Key generation:

- Choose a lattice  $L \subseteq \mathbb{Z}^n$ .
- Generate a reduced basis  $R$  of  $L$  (nonsingular matrix with short row vectors), e.g.  $R = \sqrt{n} \cdot I + Q$ , where  $Q_{ij} \in \{-\sigma-1, \dots, \sigma+1\}$ .
- Compute a non-reduced basis  $B$  from  $R$ .
- $K_{\text{private}} = R$ ;  $K_{\text{public}} = B$ .



# Goldreich-Goldwasser-Halevi Cryptosystem

- Encryption of a message  $m \in \mathbb{Z}^n$ :
  - Choose  $e \in \{-\sigma, \sigma\}^n$  uniformly at random.
  - Compute the ciphertext  $c \leftarrow mB + e$ .
- Decryption of a ciphertext  $c$ :
  - Compute  $m \leftarrow \lfloor cR^{-1} \rfloor RB^{-1}$ .
- Trouble: ciphertext leaks information on the plaintext; decrypting reduces to a special CVP much easier than the general CVP (Nguyen 1999).



# Micciancio Cryptosystem

- Modification of GGH, over an order of magnitude faster and more compact for the same lattice dimension.
- Key generation:
  - Choose a random matrix  $M$  with  $M_{ij} \in \{-n, \dots, n\}$ , and apply lattice basis reduction to the basis defined by the columns of  $M$ , obtaining  $R$  as result.
  - Compute the Hermite normal form  $B$  of  $R$  (upper triangular, strictly positive diagonal,  $0 \leq B_{ij} < B_{ii}$ ).
  - $K_{\text{private}} = R$ ;  $K_{\text{public}} = B$ .
- The Micciancio cryptosystem can only be secure if  $n \geq 780$  (Ludwig 2004).



# Quantum Security of Lattice Reduction

- The ability to solve the HDSP yields unique shortest vectors in lattices (Regev 2002).
- $\exists$  quantum algorithm to solve the HDSP with subexponential complexity  $2^{O(\sqrt{n})}$  (Kuperberg 2003).
- Cost of solving the HDSP on a quantum computer is comparable to that of solving the IFP or DLP on a classical computer.



# Regev Cryptosystem (I)

- Let  $N = 2^{8n^2}$  for a security parameter  $n$ .
- Key generation:
  - Choose uniformly at random a value  $h$  with  $N^{1/2} \leq h < 2N^{1/2}$ ;  $m = O(\log N)$  integer values  $a_1, \dots, a_m \in [0..N-1]$  that are close to integer multiples of  $N/h$ ; and an index  $j$  such that  $a_j$  is close to an odd multiple of  $N/h$ .
  - $K_{\text{private}} = h$ ;  $K_{\text{public}} = (a_1, \dots, a_m, j)$ .



# Regev Cryptosystem (I)

- Encryption of a *single* bit  $b$ :
  - Choose a random subset  $S \subseteq [1..m]$ .
  - Ciphertext is  $c = (b \lfloor a_j/2 \rfloor + \sum_{i \in S} a_i) \bmod N$ .
- Decryption of a ciphertext  $c$ :
  - Set  $b \leftarrow 0$  if  $\text{frac}(c / (N/h)) < 1/4$ , otherwise set  $b \leftarrow 1$ .
- The size of  $V$  is  $O(m \log N) = \tilde{O}(n^4)$ , and the encryption expands the message by a factor  $O(\log N) = \tilde{O}(n^2)$ .



# Regev Cryptosystem (II)

- Security parameter  $n$ .
- System parameters:  $m$ , prime  $p \approx n^2$ , probability distribution  $\chi$  on  $\mathbb{Z}_p$ .
- Key pair:
  - $K_{\text{private}}$ :  $s \in (\mathbb{Z}_p)^n$  chosen at random;
  - $K_{\text{public}}$ :  $V \leftarrow (\mathbf{a}_i, u_i \mid i = 1, \dots, m)$ , where  $\mathbf{a}_i \in (\mathbb{Z}_p)^n$  chosen uniformly at random,  $u_i \in \mathbb{Z}_p$  given by  $u_i \leftarrow \langle \mathbf{a}_i, s \rangle + e_i$  with  $e_i \in \mathbb{Z}_p$  chosen at random with distribution  $\chi$ .



# Regev Cryptosystem (II)

- Encryption of a *single* bit  $b$ :
  - Choose a random subset  $S \subseteq [1..m]$ .
  - The ciphertext is  $(\sum_{i \in S} \mathbf{a}_i, \lfloor p/2 \rfloor + \sum_{i \in S} u_i)$ .
- Decryption of a pair  $(\mathbf{a}, u)$ :
  - Set  $b \leftarrow 0$  if  $u - \langle \mathbf{a}, \mathbf{s} \rangle$  is closer to 0 than to  $\lfloor p/2 \rfloor \bmod p$ , otherwise set  $b \leftarrow 1$ .
- The size of  $V$  is  $O(mn \log p) = \tilde{O}(n^2)$ , and the encryption expands the message by a factor  $O(n \log p) = \tilde{O}(n)$ .



# Regev Cryptosystem (II)

- The security of Regev's 2nd cryptosystem is based on SIVP and the general SVP (not merely the unique-SVP).
- Breaking it implies an efficient *quantum* algorithm for approximating SIVP and SVP to within  $\tilde{O}(n^{3/2})$ .
- No quantum algorithm is known to solve the SIVP or the SVP more efficiently than classical ones.



# Lattice-Based Signatures

- Public key: basis  $B$  of a lattice  $L \subseteq \mathbb{Z}^n$ .
- Secret key: basis  $S = (s_1, \dots, s_n)$  of  $L$  with short vectors.
- Given a vector  $u \in \mathbb{Z}^n$ , knowledge of  $S$  allows the computation of  $z \in L$  that is close to  $u$ : if  $u = \sum_i u_i s_i$ , then  $z = \sum_i \lfloor u_i \rfloor s_i$ .

# NTRU

- ❑ Peculiar “lattice-based” cryptosystem (not purely so, as it involves finite field and ring arithmetic).
- ❑ NTRUEncrypt, NTRUSign.
- ❑ Many modifications to NTRU were necessary since its introduction in 1996.
- ❑ *I regret to say that there is no undisputed evidence that NTRU is secure on a classical computer (let alone a quantum one)...*

❑ Yet it's being standardized by IEEE!





# NTRU

## □ Setup:

- Choose a prime  $N$  (e.g. 521 for  $2^{128}$  security) a small prime  $p$  (typically 3), and a large prime  $q \gg p$  (typically 251).
- Let  $R = \mathbb{Z}[X]/(X^N - 1)$ ,  $R_p = \mathbb{Z}_p[X]/(X^N - 1)$ , and  $R_q = \mathbb{Z}_q[X]/(X^N - 1)$ .

## □ Key generation:

- Randomly generate small polynomials  $f, g \in R$ .
- Invert  $f$  in  $R_q$  to obtain  $f_q^{-1}$ , invert  $f$  in  $R_p$  to obtain  $f_p^{-1}$ , and check that  $g$  is invertible in  $R_q$ .
- $K_{\text{public}}: h \leftarrow p * g * f_q^{-1} \pmod{q}$ .  $K_{\text{private}}: (f, f_p^{-1})$ .



# NTRU

## □ Encryption of a message $m$ :

- Randomly select a small polynomial  $r \in R$ .
- Compute the ciphertext  $c \leftarrow r * h + m \pmod{q}$ .

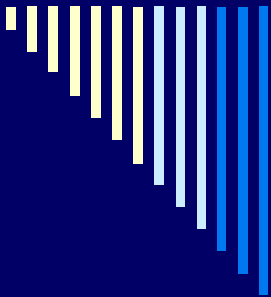
## □ Decryption of a ciphertext $c$ :

- Compute  $a \leftarrow \text{center}(f * e)$ , where function `center` reduces its argument to the interval  $[A..A+q-1]$  for certain constant  $A$ .
- Recover the message  $m \leftarrow f_p * a \pmod{p}$ .

# NTRU

- Compact keys:  $O(n)$  instead of  $O(n^2)$  as other lattice-based schemes.
- Fast processing:  $O(n^2)$  instead of  $O(n^3)$  as RSA or ECC.
- Basic NTRUSign leaks information about the private key, which can be recovered by observing  $O(n)$  signatures (Nguyen 2006).
- NTRUEncrypt might still be useful.





# Syndrome Decoding



# Syndrome Decoding

- The (Hamming) *weight*  $w(u)$  of  $u \in (\mathbb{F}_2)^n$  is the number of nonzero components of  $u$ .
- The distance between  $u, v \in (\mathbb{F}_2)^n$  is  $\text{dist}(u, v) \equiv w(u \oplus v)$ .
  
- Let  $q = 2^m$  for some  $m$ . A *binary*  $(n, k)$ -code  $\mathcal{C}$  over  $\mathbb{F}_q$  is a  $k$ -dimensional vector subspace of  $(\mathbb{F}_2)^n$ .

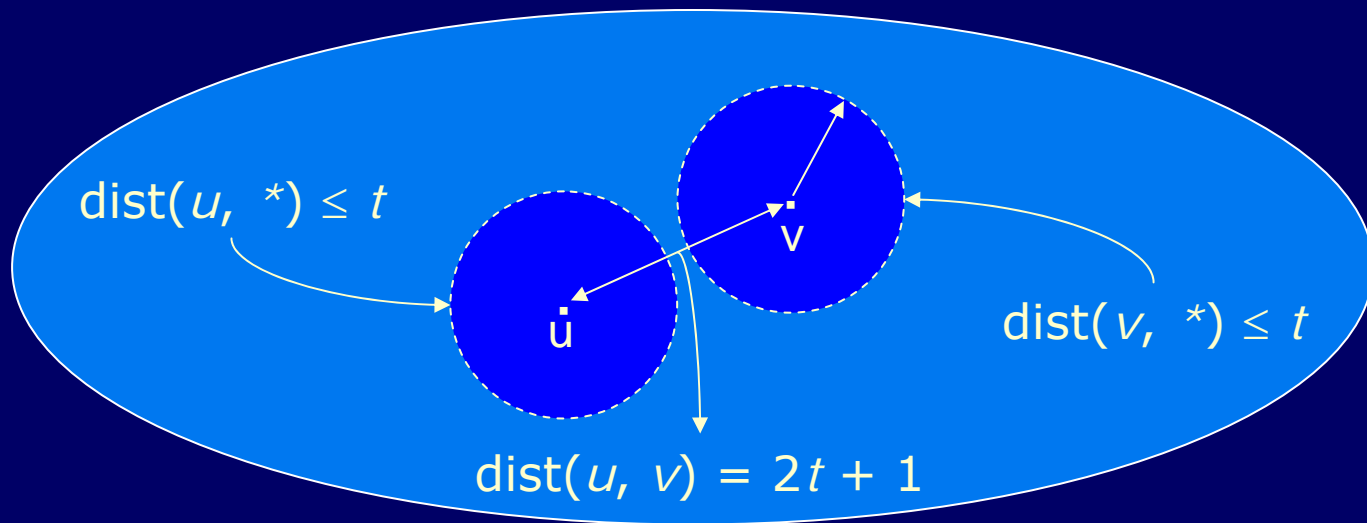


# Syndrome Decoding

- Syndrome Decoding Problem (SDP): given a binary  $(n, k)$ -code  $\mathcal{C}$  and  $c \in (\mathbb{F}_2)^n$ , find  $v \in \mathcal{C}$  such that  $\text{dist}(v, c)$  is minimal.
- In general, the SDP is *NP*-hard ☹.
- In certain cases, knowledge of a certain privileged information allows for efficiently solving the SDP 😊.

# Syndrome Decoding

- Let  $d = \min\{\text{dist}(u, v) \mid u, v \in \mathcal{C}\}$ . If  $v, e \in (\mathbb{F}_2)^n$  and  $w(e) \leq \lfloor (d-1)/2 \rfloor \equiv t$ , the SDP has a unique solution for  $c = v \oplus e$ .





# Syndrome Decoding

- Determining the minimum distance of a linear code is *NP*-hard.
- Bounded Distance Decoding Problem (BDDP):
  - Given a binary  $(n, k)$ -code  $\mathcal{C}$  with known minimum distance  $d$  and  $c \in (\mathbb{F}_2)^n$ , find  $v \in \mathcal{C}$  such that  $\text{dist}(v, c) = d$ .
- $\therefore$  BDDP is SDP with knowledge of  $d$ .
- BDDP is *believed* (but not known for sure) to be intractable.



# Goppa Codes

□ Let  $g(x) = \sum_{i=0}^t g_i x^i$

be a monic ( $g_t = 1$ ) irreducible polynomial in  $\mathbb{F}_q[x]$ .

□ Properties:

- $\mathbb{F}_{q^t} = \mathbb{F}[x]/g(x)$ .

- $g(u) \neq 0, \forall u \in \mathbb{F}_q$ .

□ Let  $L = (L_0, \dots, L_{n-1}) \in (\mathbb{F}_q)^n$  (all distinct).



# Goppa Codes

- The *syndrome function* is the linear map  $S: (\mathbb{F}_2)^n \rightarrow \mathbb{F}_{q^t}$  defined by:

$$S(c) = \sum_{i=0}^{n-1} \frac{c_i}{x - L_i} = \sum_{c_j=1} \frac{1}{x - L_j}.$$

- The *Goppa code*  $\Gamma(L, g)$  is the kernel of the syndrome function, i.e.  $\Gamma = \{c \in (\mathbb{F}_2)^n \mid S(c) = 0\}$ .



# Goppa Codes

- Determining whether a bit vector is a code word amounts to computing its syndrome.
- The syndrome function is linear and can be written in matrix form:  $S(c) = Hc^T$ , where  $H$  is the *parity check matrix* of the code.
- Easy to compute  $H$  from  $L$  and  $g$  (deriving the formula is boring, though).



# Goppa Codes

□ Parity check matrix:  $H_{t \times n} = C_{t \times t} V_{t \times n} D_{n \times n}$   
where:

$$C = \begin{bmatrix} g_t & 0 & \dots & 0 \\ g_{t-1} & g_t & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ g_1 & g_2 & \dots & g_t \end{bmatrix}, \quad V = \begin{bmatrix} 1 & 1 & \dots & 1 \\ L_0 & L_1 & \dots & L_{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ L_0^{t-1} & L_1^{t-1} & \dots & L_{n-1}^{t-1} \end{bmatrix},$$

$$D = \begin{bmatrix} 1/g(L_0) & 0 & \dots & 0 \\ 0 & 1/g(L_1) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1/g(L_{n-1}) \end{bmatrix}.$$



# Goppa Codes

- A Goppa code  $\Gamma$  is a  $k$ -dimensional subspace of  $(\mathbb{F}_2)^n$  for some  $k$ .
- A generator matrix for  $\Gamma$  is a matrix  $G_{k \times n}$  whose rows form a basis of  $\Gamma$ .
- $G$  defines a mapping  $(\mathbb{F}_2)^k \rightarrow (\mathbb{F}_2)^n$  such that  $uG \in \Gamma, \forall u \in (\mathbb{F}_2)^k$ .
- Therefore  $H(uG)^T = 0$ , i.e.  $HG^T = 0$ .
- N.B.:  $H$  can be interpreted as an  $mt \times n$  binary matrix, and  $k \geq n - mt$ .



# Goppa Codes

- A Goppa code  $\Gamma$  has minimum distance  $d = 2t + 1$ , hence can correct up to  $t$  errors.
- Efficient decoding procedure for known  $g$  and  $L$  via *error locator polynomial*:

$$\sigma(x) \equiv \prod_{e_j=1} (x - L_j) \in \mathbb{F}_q[x]/g(x).$$

- Property:  $\sigma(L_j) = 0 \Leftrightarrow e_j = 1$ .



# Error Correction for Goppa Codes

- Let  $m \in \Gamma$ , let  $e \in (\mathbb{F}_2)^n$  be an error vector of weight  $w(e) \leq t$ , and  $c = m \oplus e$ .
- Compute the syndrome of  $e$  through the relation  $S(e) = S(c)$ .
- Compute the error locator polynomial  $\sigma$  from the syndrome.
- Determine which  $L_i$  are zeroes of  $\sigma$ , thus retrieving  $e$  and recovering  $m$ .

# Error Correction for Goppa Codes

- Let  $s(x) \leftarrow S(e)$ . If  $s(x) \equiv 0$ , nothing to do (no error), otherwise  $s(x)$  is invertible.
- Property #1:  $\sigma(x) = a(x)^2 + xb(x)^2$ .
- Property #2:  $d\sigma(x)/dx = b(x)^2$ .
- Property #3:  $d\sigma(x)/dx = \sigma(x)s(x)$ .
- Thus  $b(x)^2 = (a(x)^2 + xb(x)^2)s(x)$ , hence  $a(x) = b(x)z(x)$  where  $z(x) = \sqrt{x + 1/s(x)}$

Extended Euclid!





# McEliece Cryptosystem

- Key generation:
- Choose a  $t$ -error correcting Goppa code  $\Gamma(L, g)$  and compute for it a  $k \times n$  binary generator matrix  $G$ .
- Choose a random  $k \times k$  binary nonsingular matrix  $S$  and a random  $n \times n$  permutation matrix  $P$ , and compute  $E = SGP$ .
- $K_{\text{private}} = (P^{-1}, \Gamma, S^{-1})$ ;  $K_{\text{public}} = (E, t)$ .



# McEliece Cryptosystem

- Encryption of  $k$ -bit plaintext  $m$ :
  - Choose a random binary error vector  $e$  of length  $n$  and weight  $w(e) = t$ .
  - Compute the ciphertext  $c = mE + e$ .
- Decryption of  $n$ -bit ciphertext  $c$ :
  - Compute  $c' = cP^{-1}$ . Notice that  $c' = (mSGP + e)P^{-1} = mSG + eP^{-1} = m'G + e'$ , with  $m' = mS$  and  $e' = eP^{-1}$ .
  - Correct  $e'$  and decode  $c'$  to  $m'$  ( $n$  bits  $\rightarrow k$  bits) using the decoding algorithm for  $\Gamma$ ;
  - Compute the plaintext  $m = m'S^{-1}$ .



# Niederreiter Cryptosystem

- Key generation:
- Choose a  $t$ -error correcting Goppa code  $\Gamma(L, g)$  and compute for it an  $(n-k) \times n$  binary parity check matrix  $H$ .
- Choose a random  $(n-k) \times (n-k)$  binary nonsingular matrix  $S$  and a random  $n \times n$  permutation matrix  $P$ , and compute  $E = SHP$ .
- $K_{\text{private}} = (S^{-1}, \Gamma, P^{-1})$ ;  $K_{\text{public}} = (E, t)$ .



# Niederreiter Cryptosystem

## □ Encryption of plaintext $m$ :

- Represent  $m$  as an  $n$ -bit vector  $u$  of weight  $w(u) = t$  via permutation unranking.
- Compute the  $(n-k)$ -bit ciphertext  $c = Eu^T$ .

## □ Decryption of ciphertext $c$ :

- Compute  $c' = S^{-1}c$ . Notice that  $c' = S^{-1}(SHPu^T) = H(Pu^T) = Hu'$ , with  $u' = Pu^T$ .
- Decode the syndrome  $c'$  to  $u'$  using the decoding algorithm for  $\Gamma$ ;
- Compute  $u = P^{-1}u'$  and obtain the plaintext  $m$  via permutation ranking.



# Niederreiter Cryptosystem

- In principle, the security of McEliece and Niederreiter are exactly equivalent, even though McEliece includes randomization and Niederreiter is deterministic.
- Original Niederreiter setting: generalized Reed-Solomon (GRS) codes instead of the more restricted class of Goppa codes.



# Choosing Parameters

## □ Original McEliece setting:

- $m = 10$ ,  $n = 2^m = 1024$  (hence  $L$  spans  $\mathbb{F}_{2^m}$ ),  $t = 50$ ,  
 $k = n - mt = 524$  (minimum possible), security  $\approx 2^{54}$ ,  
key size = 65.5 KiB.

## □ Other choices:

$m$	$n$	$t$	$k$	security	key size
11	1600	64	896	$2^{78}$	175 KiB
12	2560	64	1792	$2^{114}$	560 KiB
12	2880	80	1920	$2^{130}$	675 KiB
12	4096	128	2560	$2^{186}$	1280 KiB



# Choosing the Code

- Most syndrome-based cryptosystems can be instantiated with general  $(n, k)$ -codes.
- Not all choices of code are secure.
  - McEliece with maximum rank distance (MRD) or Gabidulin codes is insecure (Gibson 1995, 1996).
  - Niederreiter with GRS codes is insecure (Sidelnikov-Shestakov 1992).
- Goppa seems to be OK.
  - Easy to generate (Gao-Panario 1997).
  - Distinguishing a permuted Goppa code from a random code of the same length and distance (Sendrier 2000):  $O(t n^{t-2} \log^2 n)$ .



# CFS Signatures

- Security based on the BDDP assumption.
- Represent the message as a decodable syndrome, then decode the syndrome to produce the error vector as the signature.
- Verify the signature by matching it to the syndrome of the message.



# CFS Signatures

## □ System setup:

- Choose  $m, t \leq m$  and  $n = 2^m$ .
- Choose a hash function  $h: \{0, 1\}^* \times \mathbb{N} \rightarrow (\mathbb{F}_2)^{n-k}$ .

## □ Key generation:

- Choose a  $t$ -error correcting Goppa code  $\Gamma(L, g)$  and compute for it an  $(n-k) \times n$  binary parity check matrix  $H$  in Niederreiter (i.e. disguised) form.
- $K_{\text{private}} = \Gamma; K_{\text{public}} = (H, t)$ .



# CFS Signatures

- Signing a message  $m$ :
  - Find the smallest  $i \in \mathbb{N}$  such that  $c \leftarrow h(m, i)$  is a codeword of  $\Gamma$ .
  - Compute the error vector  $e$  of weight  $t$  such that  $c = He^T$  using the decoding algorithm for  $\Gamma$ . The signature is  $(e, i)$ .
- Verifying a signature  $(e, i)$ :
  - Compute  $c \leftarrow He^T$ .
  - Accept the signature iff  $c = h(m, i)$  for the smallest possible  $i$ .



# CFS Signatures

- The number of possible hash values is  $2^{n-k} = 2^{mt} = n^t$  and the number of syndromes decodable to codewords of weight  $t$  is

$$\binom{n}{t} \approx \frac{n^t}{t!}$$

- ∴ The probability of finding a codeword of weight  $t$  is  $\approx 1/t!$ , and the expected value of hash queries is  $\approx t!$ .



# CFS Signatures

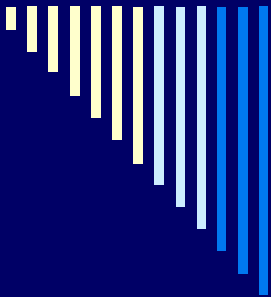
- If the  $n$ -bit error  $e$  of weight  $t$  is encoding via permutation ranking, the signature length is  $\approx \lg(n^t/t!) + \lg(t!) = t \lg n$ .
- Public key is huge:  $mtn$  bits.
- Default recommendation:  $m = 16$ ,  $t = 9$ ,  $n = 2^{16}$ , signature length = 144 bits, key size = 1152 KiB, security  $\approx 2^{72}$ .



# CFS Signatures

- Shortest signatures known (down to 81 bits).
- Trick: “lossy compression” (omit some error bits when signing, guess them when verifying).
- Possible for other signatures, very effective CFS ( $\approx m$ -bit decrease for each omitted error bit).
- Claim: security level remains at  $\approx 2^{81}$ .
  
- Fact: only  $2^{41}$  security (birthday paradox).





# Merkle Signatures

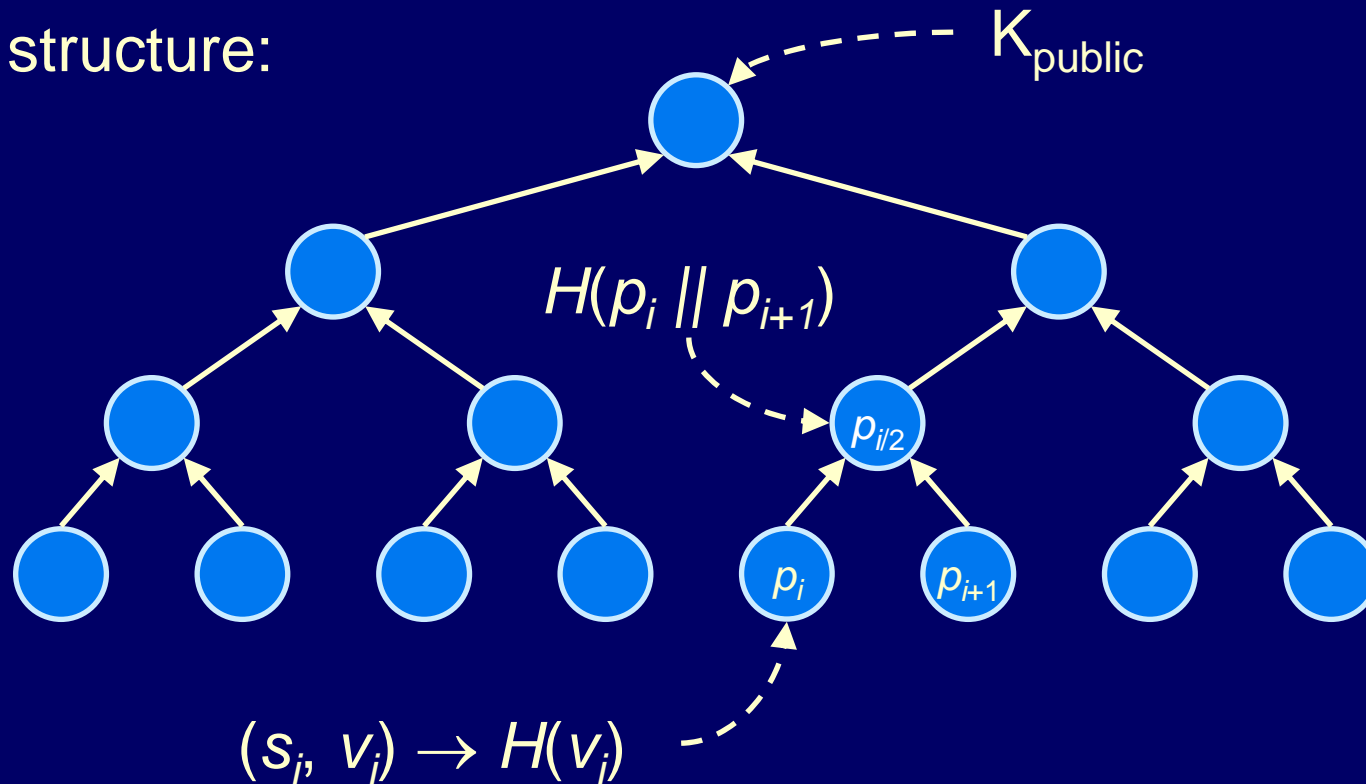


# Merkle Signatures

- Tree-structured, hash-based construction.
- Fixed maximum number of signatures:  $2^N$ .
- Underlying one-time signature scheme  $(R, U, S, \text{sign}, \text{verify})$ :
  - $R$ : private key space
  - $U$ : public key space
  - $S$ : signature space
  - $\text{sign}: R \times \{0,1\}^* \rightarrow S$
  - $\text{verify}: U \times \{0,1\}^* \times S \rightarrow \{0,1\}$
- Condition:  $\forall (s, v) \in R \times U, \forall m \in \{0,1\}^*:$   
 $\text{sign}(s, m) = h \Leftrightarrow \text{verify}(v, m, h) = 1.$

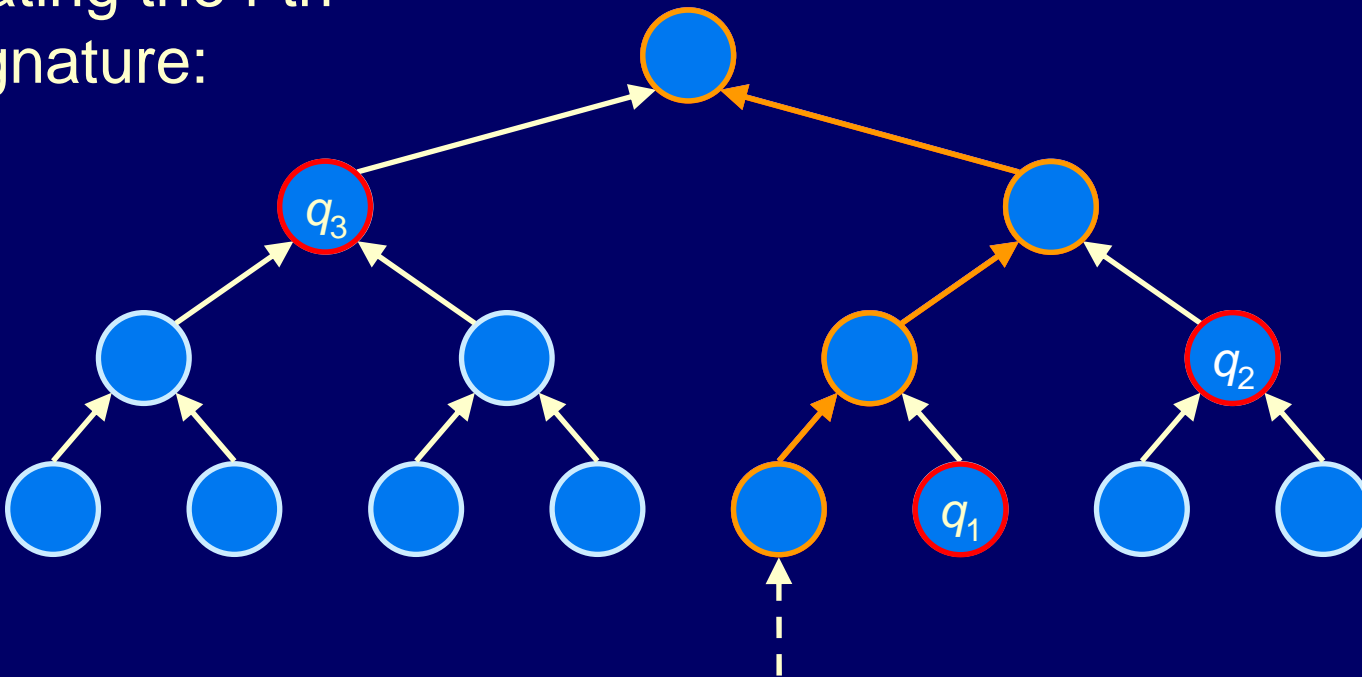
# Merkle Signatures

Key structure:



# Merkle Signatures

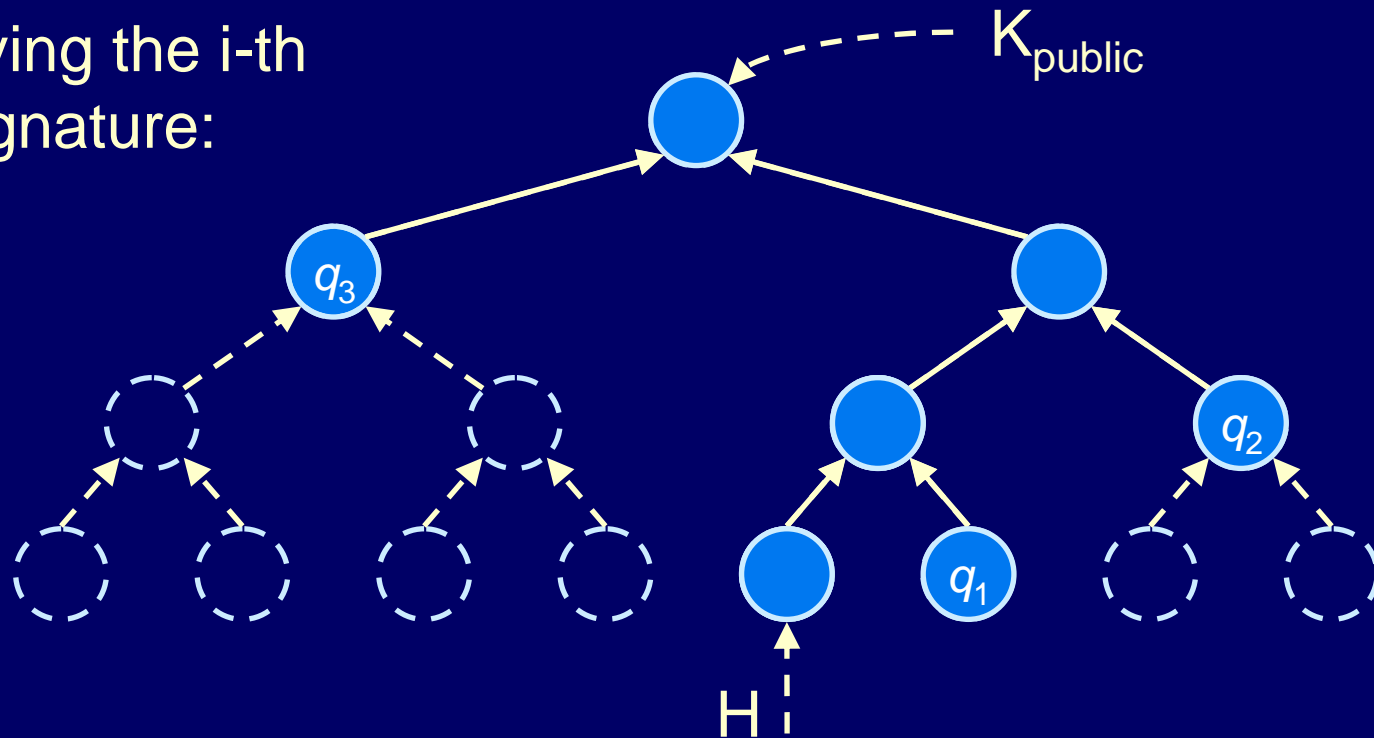
Generating the  $i$ -th signature:



$$\sigma = (\text{sign}(s_i, m), v_i, q_1, \dots, q_{N-1})$$

# Merkle Signatures

Verifying the  $i$ -th signature:



$$\sigma = (\text{sign}(s_i, m), v_i, q_1, \dots, q_{N-1})$$



# Merkle Signatures

- Very fast signing and verification, but very slow key generation:  $O(2^M)$  steps.
- Small keys (if private keys derived from a seed), long  $O(M)$  signatures.
- Several recent improvements, especially key generation (Buchmann et al. 2007).
- Depends on the availability of secure hash functions.

---



# Epilog

What if  $NP \subset BQP$ ?



# Assessment

- Even if  $P = NP$ ,  $NP$ -hard problems not in  $NP$  could remain intractable.
- In any case, there might be theoretically infeasible (strictly exponential) problems that are tractable in practice.
- If all else fails, try using *non-recursive decision problems*, making cryptosystems secure against computationally *unbounded* attackers!) (Myasnikov-Shpilrain-Ushakov 2006)



# Strictly Exponential Cryptosystems

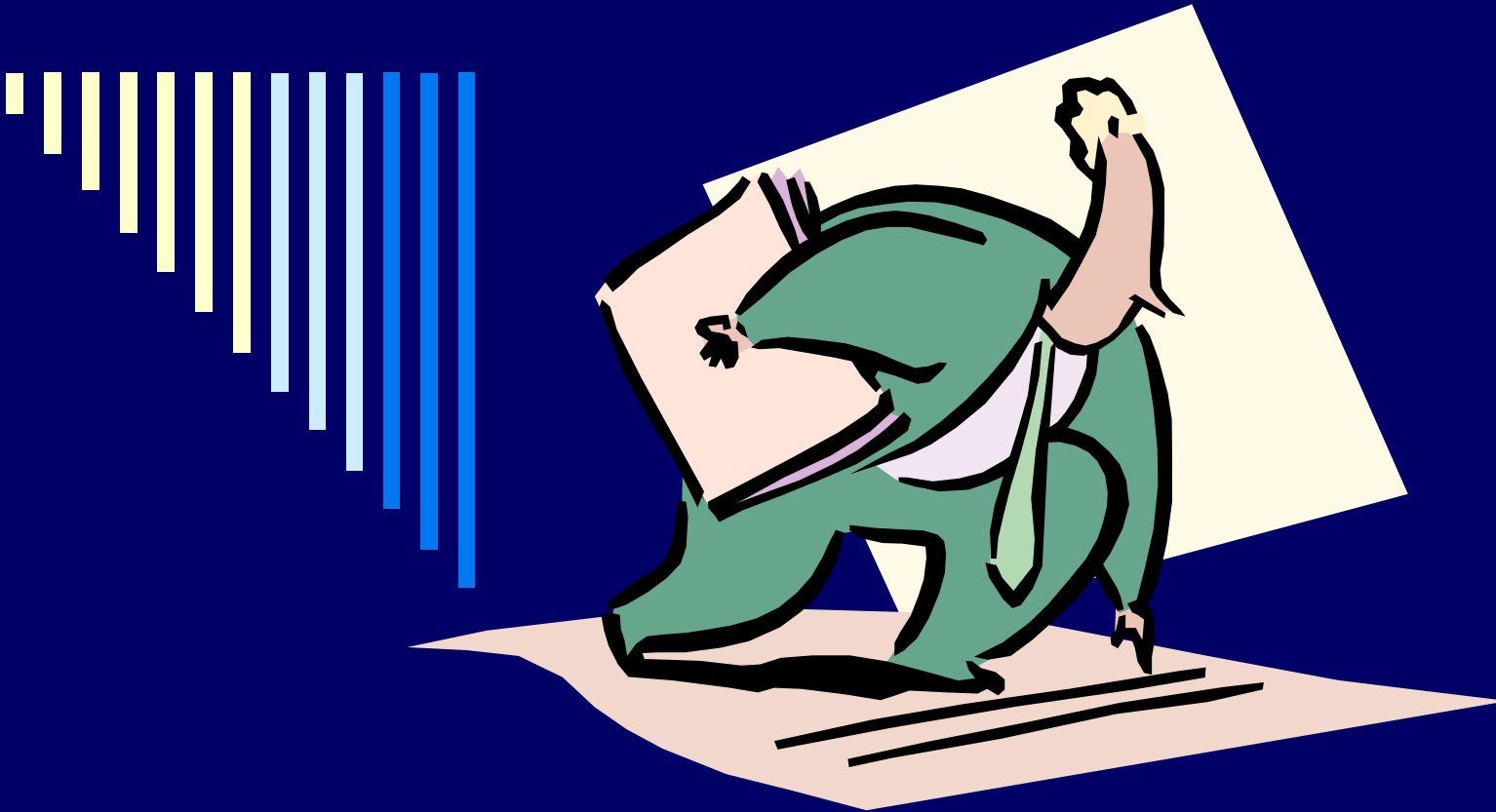
- Security parameters:  $s, n (\gg s)$ .
- $F: S \rightarrow M$  such that:
  - $F$  computable with cost  $\Theta(2^s)$ ,
  - $F^{-1}$  computable with cost  $\Theta(2^s)$  if privileged information is known, otherwise  $\Omega(2^n)$ .
- Example:
  - Key pair: private  $F^{-1}$ ; public  $F$ .
  - Signing:  $c = F^{-1}(m)$ .
  - Verification:  $m = F(c)$ .



---

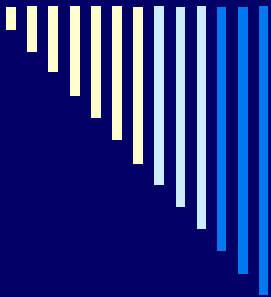
# Conclusions

- To be sure, we *still* don't know what PQC is, do we?
- There are lots of fascinating if heterodox proposals, and many of them may turn out to deserve the "post-quantum" label.
- What is best: vast, excellent opportunities for new research.



Thanks!

---



# Appendix

# A Note On What Is Technically Possible

- Bacon showed that “a quantum computer which has access to closed timelike curve qubits can solve NP-complete problems with only a polynomial number of quantum gates” (Bacon 2004).
- “Closed timelike curve” is the technical term for a time machine!





# Extended Euclidean Algorithm

- Given:  $z(X), g(X) \in \mathbb{F}[X]$
- Find:  $a(X), b(X), f(X) \in \mathbb{F}[X]$
- Where:  $b(X)z(X) + f(X)g(X) = a(X)$
- Thus  $a(X) = b(X)z(X) \bmod g(X)$ , i.e.  
 $a(X) = b(X)z(X)$  in  $\mathbb{F}[X]/g(X)$ .
- Conditions:  $\deg(a) \leq \lfloor t/2 \rfloor$ ,  $\deg(b) \leq \lfloor (t-1)/2 \rfloor$ .



# Extended Euclidean Algorithm

```
A ← z, a ← g, B ← 1, b ← 0, X ← 0, x ← 1
while (deg(a) ≥ ⌊(t + 1)/2⌋) {
  if (A < a) {
    A ↔ a, B ↔ b
  }
  λ ← ⌊A/a⌋
  A ← A - λa, B ← B - λb
}
```





# Ranking and Unranking Permutations

- Let  $P(n,t) = \{u \in (\mathbb{F}_2)^n \mid w(u) = t\}$ , with cardinality

$$r = \binom{n}{t} \approx \frac{n^t}{t!}$$

- A *ranking function* is a mapping  $rank: P(n,t) \rightarrow [1..r]$  which associates a unique index in  $[1..r]$  to each element in  $P(n,t)$ . Its inverse is called the *unranking function*.
- Ranking and unranking can be done in  $O(n)$  time (Myrvold-Ruskey 2001).



# Birthday paradox for $\ell$ -bit CFS signatures

- Generate  $2^{\ell/2}$  syndromes  $c_j \leftarrow h(m_j, i_j)$ ;
- Compute the syndromes  $c_k \leftarrow He_k^T$  of the error vectors in a collection of  $2^{\ell/2}$  valid signatures  $(e_k, i_k)$ ;
- Find a collision  $c_j = c_k$  and forge the signature  $(e_k, i_j)$  for  $m_j$ .
- Security level:  $2^{\ell/2}$ .





# KLCHKP Cryptosystem

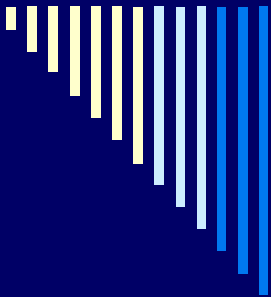
- Symmetric cipher  $E: G \times \{0,1\}^* \rightarrow \{0,1\}^*$ .
- Setup:
  - Choose  $G$  and two disjoint, commuting subgroups  $LG, RG \leq G$ , e.g.  $G = \langle \sigma_1, \dots, \sigma_{2n-1} \rangle$ ,  $LG = \langle \sigma_1, \dots, \sigma_{n-1} \rangle$ ,  $RG = \langle \sigma_{n+1}, \dots, \sigma_{2n-1} \rangle$ .
  - Choose a “sufficiently complicated” word  $g$  in the generators of  $LG$  and  $RG$ .
- Key generation:
  - Choose  $s \in LG$  uniformly at random and compute  $v \leftarrow s^{-1}gs$ .
  - $K_{\text{private}} = s$ ;  $K_{\text{public}} = v$ .



# KLCHKP Cryptosystem

- Encryption of a message  $m$ :
  - Choose  $x \in \text{RG}$  uniformly at random.
  - Compute the nonce  $r \leftarrow x^{-1}gx$  and the key  $k \leftarrow x^{-1}vx$ .
  - Compute the ciphertext  $(r, E(k, m))$ .
  
- Decryption of a ciphertext  $(r, c)$ :
  - Compute the key  $k \leftarrow s^{-1}rs$ .
  - N.B.  $s^{-1}rs = s^{-1}x^{-1}gx s = x^{-1}s^{-1}g s x = x^{-1}vx$ .
  - Recover the message  $m \leftarrow E^{-1}(k, c)$ .





# References



# References

- [Anshel-Anshel-Goldfeld 1999] I. Anshel, M. Anshel, D. Goldfeld, "An Algebraic Method for Public-Key Cryptography", Math. Research Letters 6, 1999.
- [Anshel-Anshel-Fisher-Goldfeld 2001] I. Anshel, M. Anshel, B. Fisher, D. Goldfeld, "New Key Agreement Protocols in Braid Group Cryptography", CT-RSA, LNCS 2020, 2001.
- [Bacon 2004] D. Bacon, "Quantum computational complexity in the presence of closed timelike curves", Phys. Rev. A, 2004.
- [Birman-Ko-Lee 1998] J. S. Birman, K. H. Ko, S. J. Lee, "A new approach to the word and conjugacy problems in the braid groups", Adv. Math. 139 (1998).
- [Bridson-Howie 2003] M. R. Bridson, J. Howie, "Conjugacy of finite subsets in hyperbolic groups", IJAC 15(1), 2005.
- [Buchmann et al. 2004] J. Buchmann, C. Coronado, M. Döring, D. Engelbert, C. Ludwig, R. Overbeck, A. Schmidt, U. Vollmer, R.-P. Weinmann, "Post-Quantum Signatures", 2004.



# References

- [Buchmann et al. 2007] “Merkle Signatures with Virtually Unlimited Signature Capacity”, ACSN’2007, LNCS 4521, 2007.
- [Cha et al. 2001] J. C. Cha, K. H. Ko, S. J. Lee, J. W. Han, J. H. Cheon, “An Efficient Implementation of Braid Groups”, Asiacrypt’2001, LNCS 2248.
- [Cheon-Jun 2003] J. H. Cheon, B. Jun, “A Polynomial Time Algorithm for the Braid Diffie-Hellman Conjugacy Problem”, Crypto’2003, LNCS 2729, 2003.
- [Dubois et al. 2007] V. Dubois, P.-A. Fouque, A. Shamir, J. Stern, “Practical Cryptanalysis of SFLASH”, Crypto’2007, LNCS 4622, 2007.
- [Eick-Kahrobaei 2004] B. Eick, D. Kahrobaei, “Polycyclic groups: A new platform for cryptology?”, arXiv:math/0411077v1, 2004.
- [van Emde-Boas 1981] P. van Emde Boas, “Another NP-complete partition problem and the complexity of computing short vectors in a lattice”, Tech. Report 81-04, University of Amsterdam, Netherlands, 1981.



# References

- [Gao-Panario 1997] S. Gao, D. Panario, "Tests and Constructions of Irreducible Polynomials over Finite Fields", FoCM'97, 1997.
- [Gibson 1995] J. K. Gibson, "Severely denting the Gabidulin version of the McEliece public key cryptosystem", Designs, Codes and Cryptography 6, 1995.
- [Gibson 1996] J. K. Gibson, "The Security of the Gabidulin Public Key Cryptosystem", Eurocrypt'1996, LNCS 1070, 1996.
- [Goldreich-Goldwasser-Halevi 1997] O. Goldreich, S. Goldwasser, S. Halevi. "Public-key cryptosystems from lattice reduction problems", Crypto'1997, LNCS 1294, 1997.
- [Hofheinz-Steinwandt 2003] D. Hofheinz, R. Steinwandt, "A Practical Attack on Some Braid Group Based Cryptographic Primitives", PKC 2003, LNCS 2567, 2003.
- [Hughes 2002] J. Hughes, "A Linear Algebraic Attack on the AAFG1 Braid Group Cryptosystem", ACISP'2002, LNCS 2384, 2002.



# References

- [Knudsen-Meier 1999] L. R. Knudsen, W. Meier, "Cryptanalysis of an Identification Scheme Based on the Permuted Perceptron Problem", Eurocrypt'1999, LNCS 1592, 1999.
- [Ko et al. 2000] K. H. Ko, S. J. Lee, J. H. Cheon, J. W. Han, J.-S. Kang, C. Park, "New Public-Key Cryptosystem Using Braid Groups", Crypto'2000, LNCS 1880.
- [Lee-Lee 2002] S. J. Lee, E. Lee, "Potential Weaknesses of the Commutator Key Agreement Protocol Based On Braid Groups", Eurocrypt'2002, LNCS 2332, 2002.
- [Ludwig 2004] C. Ludwig, "The security and efficiency of Micciancio's cryptosystem", Cryptology ePrint Archive, Report 2004/209.
- [Micciancio 1998] D. Micciancio, "The shortest vector problem is NP-hard to approximate within some constant", IEEE FOCS, 1998.



# References

- [Micciancio 2001] D. Micciancio, "Improving lattice based cryptosystems using the Hermite normal form", CaLC'2001, LNCS 2146, 2001.
- [Myasnikov-Shpilrain-Ushakov 2006] A. G. Myasnikov, V. Shpilrain, A. Ushakov, "Random subgroups of braid groups: an approach to cryptanalysis of a braid group based cryptographic protocol", PKC'2006, LNCS 3958, 2006.
- [Myasnikov-Ushakov 2007] A. D. Myasnikov, A. Ushakov, "Length Based Attack and Braid Groups: Cryptanalysis of Anshel-Anshel-Goldfeld Key Exchange Protocol", PKC 2007, LNCS 4450, 2007.
- [Myrvold-Ruskey 2001] W. Myrvold, F. Ruskey, "Ranking and unranking permutations in linear time", IPL 79(6), 2001.
- [Nguyen 1999] P. Nguyen, "Cryptanalysis of the Goldreich-Goldwasser-Halevi from Crypto'97", Crypto'1999, LNCS 1666, 1999.



# References

- [Pointcheval 1995] D. Pointcheval, "A new identification scheme based on the perceptrons problem", Eurocrypt'1995, LNCS 921, 1995.
- [Sendrier 2000] N. Sendrier, "Finding the permutation between equivalent linear codes: the support splitting algorithm", IEEE Trans. Inf. Theory 46, 2000.
- [Shamir 1989] A. Shamir, "An Efficient Identification Scheme Based on Permuted Kernels", Crypto'1989, LNCS 435, 1990.
- [Shpilrain-Ushakov 2004] V. Shpilrain, A. Ushakov, "The conjugacy search problem in public key cryptography: unnecessary and insufficient", 2004.
- [Sidelnikov-Shestakov 1992] V. Sidelnikov, S. Shestakov, "On cryptosystems based on generalized Reed-Solomon codes", Diskretnaya Mat. 4(3), 1992.