Post-Quantum Cryptography

Tanja Lange

Technische Universiteit Eindhoven

tanja@hyperelliptic.org

23 January 2013

Threat of quantum computers

Shor's algorithm makes polynomial time:

- integer factorization
- DLP in finite fields
- DLP on elliptic curves
- DLP in general class groups

Grover's algorithm brings faster simultaneous search in data

- some security loss in symmetric crypto (block and stream ciphers)
- some security loss in hash functions (if not VSH)

Compensate for Grover by doubling key size.

Threat of quantum computers

Shor's algorithm makes polynomial time:

- integer factorization
- DLP in finite fields
- DLP on elliptic curves
- DLP in general class groups

Grover's algorithm brings faster simultaneous search in data

- some security loss in symmetric crypto (block and stream ciphers)
- some security loss in hash functions (if not VSH)

Sorry,
no picture
available

Compensate for Grover by doubling key size.

. . . but 15 years from now . . .

Large quantum computers might be reality. Then

- RSA is dead.
- DH key exchange is dead.
- DSA is dead.
- XTR is dead.
- ECDSA is dead.
- ECC is dead.
- HECC is dead.

... but 15 years from now ...

Large quantum computers might be reality. Then

- RSA is dead.
- DH key exchange is dead.
- DSA is dead.
- XTR is dead.
- ECDSA is dead.
- ECC is dead.
- HECC is dead.
- all public key cryptography

. . . but 15 years from now . . .

Large quantum computers might be reality. Then

- RSA is dead.
- DH key exchange is dead.
- DSA is dead.
- XTR is dead.
- ECDSA is dead.
- ECC is dead.
- HECC is dead.
- all public key cryptography is dead?

... but 15 years from now ...

Large quantum computers might be reality. Then

- RSA is dead.
- DH key exchange is dead.
- DSA is dead.
- XTR is dead.
- ECDSA is dead.
- ECC is dead.
- HECC is dead.
- all public key cryptography is dead?
- Actually there are a few more public-key cryptosystems.

The “survivors”

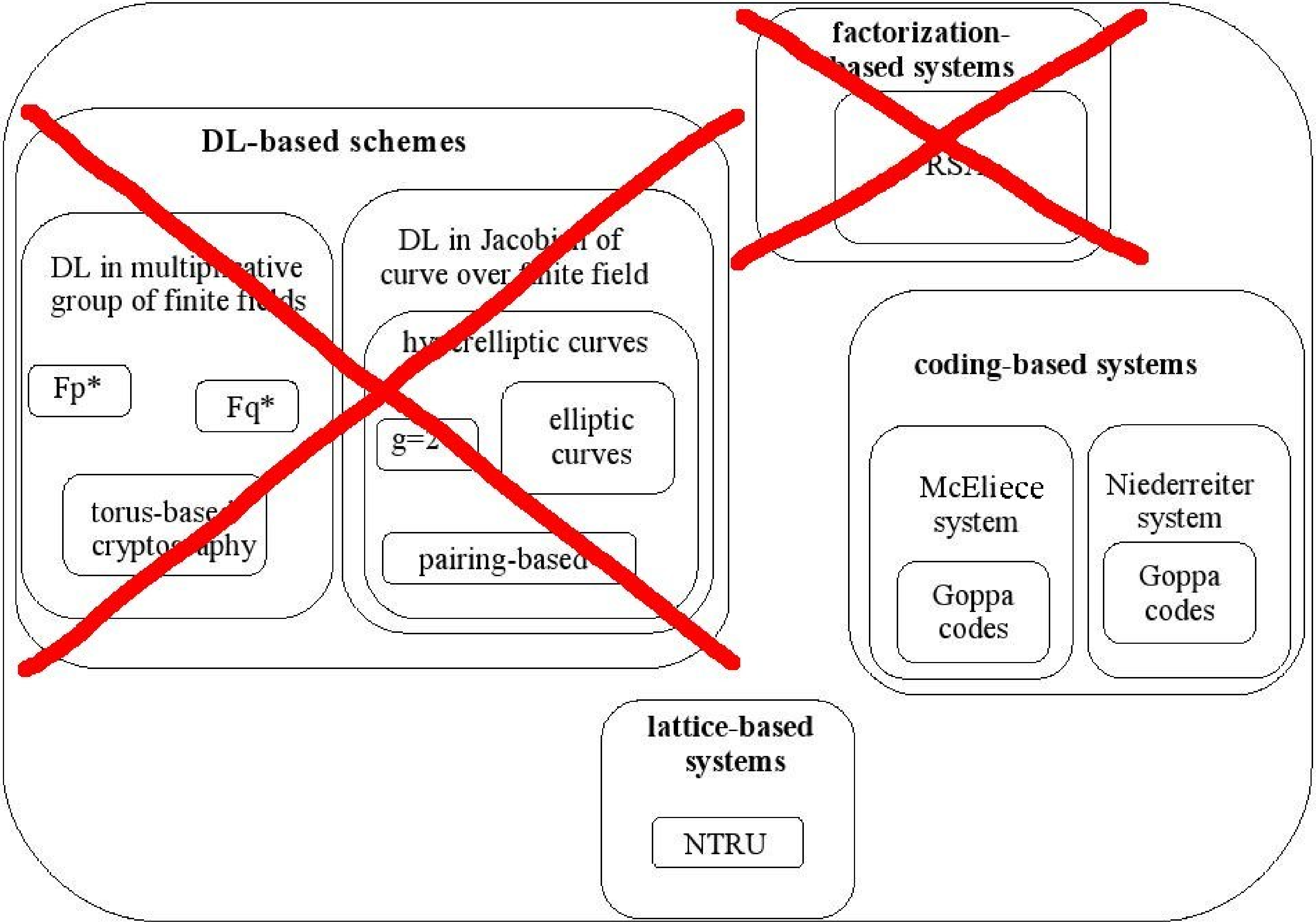
Public-key encryption:

- Lattice-based cryptography (e.g. NTRU)
- Code-based cryptography (e.g. McEliece, Niederreiter)

Public-key signatures:

- Multivariate-quadratic-equations cryptography (e.g. HFE^-)
- Hash based cryptography (e.g. Merkle’s hash-trees signatures)

For these systems no efficient usage of Shor’s algorithm is known. Grover’s algorithm has to be taken into account when choosing key sizes.
Some more possibilities with less confidence.



Encryption systems.

Why care about this now?

15 years might seem a long time. But

- There is no guarantee that it takes at least 15 years.
- Long-term confidential documents (e.g. health records, state secrets) become readable once quantum computers are available. Attacker can store all of today's encrypted data to read later.
- Electronic signatures on long-term commitments (e.g. last wishes, contracts) can be forged once quantum computers are available.
- Nobody will inform you if a secret agency made a breakthrough in constructing a quantum computer.
- The systems mentioned before remain secure – but are inefficient in time or size or both and need better embedding into protocols.

How about quantum cryptography?

- Quantum cryptography expands a short shared key into an effectively infinite shared stream.
- Requires Alice and Bob to know some (e.g. 256) unpredictable secret key bits.
- Result of quantum cryptography is that Alice and Bob both know a stream of some more (e.g. 10^{12}) unpredictable secret bits.
- Length of the output stream increases linearly with the amount of time.

How about quantum cryptography?

- Quantum cryptography expands a short shared key into an effectively infinite shared stream.
- Requires Alice and Bob to know some (e.g. 256) unpredictable secret key bits.
- Result of quantum cryptography is that Alice and Bob both know a stream of some more (e.g. 10^{12}) unpredictable secret bits.
- Length of the output stream increases linearly with the amount of time.
- Sounds like a stream cipher to you? Not exactly ...

Differences from stream ciphers

- Quantum cryptography uses physical techniques instead of mathematical function of the input key.
- Quantum cryptography needs direct connection between the quantum cryptography hardware (distance is an issue), eavesdropping interrupts the communication. Conventional cryptography can use standard channels; eavesdropping fails because the encrypted information is incomprehensible.
- Security of quantum cryptography follows from quantum mechanics instead of being merely conjectural.

Differences from stream ciphers

- Quantum cryptography uses physical techniques instead of mathematical function of the input key.
- Quantum cryptography needs direct connection between the quantum cryptography hardware (distance is an issue), eavesdropping interrupts the communication. Conventional cryptography can use standard channels; eavesdropping fails because the encrypted information is incomprehensible.
- Security of quantum cryptography follows from quantum mechanics instead of being merely conjectural.
- A stream cipher can be implemented on conventional CPUs and generates GB of stream per second on a $200 CPU. Quantum cryptography generates kB of stream per second on special hardware costing $50000.

How about quantum cryptography?

- Quantum cryptography expands a short shared key into an effectively infinite shared stream.
- Requires Alice and Bob to know some (e.g. 256) unpredictable secret key bits.
- Result of quantum cryptography is that Alice and Bob both know a stream of some more (e.g. 10^{12}) unpredictable secret bits.
- Length of the output stream increases linearly with the amount of time.

How about quantum cryptography?

- Quantum cryptography expands a short shared key into an effectively infinite shared stream.
- Requires Alice and Bob to know some (e.g. 256) unpredictable secret key bits.
- Result of quantum cryptography is that Alice and Bob both know a stream of some more (e.g. 10^{12}) unpredictable secret bits.
- Length of the output stream increases linearly with the amount of time.
- More serious problem: how to get the initial secret?! Secret meeting of agents and key exchange – or public-key cryptography.

How about quantum cryptography?

- Quantum cryptography expands a short shared key into an effectively infinite shared stream.
- Requires Alice and Bob to know some (e.g. 256) unpredictable secret key bits.
- Result of quantum cryptography is that Alice and Bob both know a stream of some more (e.g. 10^{12}) unpredictable secret bits.
- Length of the output stream increases linearly with the amount of time.
- More serious problem: how to get the initial secret?! Secret meeting of agents and key exchange – or public-key cryptography.
- And there was no problem in symmetric cryptography in the first place.

Post-quantum cryptography

- Cryptographic systems that run on conventional computers, are secure against attacks with conventional computers, and remain secure under attacks with quantum computers are called post-quantum cryptosystems.
- Post-quantum cryptography deals with
  - the design of such systems;
  - cryptanalysis of such systems;
  - the analysis of suitable parameters depending on different threat models;
  - design of protocols using the secure primitives.
- ECRYPT story: started PQCrypto series, several research retreats and smaller conferences; most recently Lorentz Center workshop.

Warnings

- The following describes text-book versions.
- There exist e.g. CCA2 secure versions, versions with better efficiency, other finite fields . . . .

Hash-based signatures

- 1979 Lamport one-time signature scheme.
- Fix a k-bit one-way function $G : \{0,1\}^k \to \{0,1\}^k$ and hash function $H : \{0,1\}^* \to \{0,1\}^k$.
- Signer's secret key X: $2k$ strings $x_1[0], x_1[1], \ldots, x_k[0], x_k[1]$, each k bits. Total: $2k^2$ bits.
- Signer's public key Y: $2k$ strings $y_1[0], y_1[1], \ldots, y_k[0], y_k[1]$, each k bits, computed as $y_i[b] = G(x_i[b])$. Total: $2k^2$ bits.
- Signature $S(X, r, m)$ of a message m: $r, x_1[h_1], \ldots, x_k[h_k]$ where $H(r, m) = (h_1, \ldots, h_k)$.
- Must never use secret key more than once.
- Usually choose $G = H$ (restricted to k bits).
- 1979 Merkle extends to more signatures.

8-time Merkle hash tree

Eight Lamport one-time keys $Y_1, Y_2, \ldots, Y_8$ with corresponding $X_1, X_2, \ldots, X_8$, where $X_i = (x_{i,1}[0], x_{i,1}[1], \ldots, x_{i,k}[0], x_{i,k}[1])$ and $Y_i = (y_{i,1}[0], y_{i,1}[1], \ldots, y_{i,k}[0], y_{i,k}[1])$.

$$
\begin{array}{cccccccc}
X_1 & X_2 & X_3 & X_4 & X_5 & X_6 & X_7 & X_8 \\
\downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\
Y_1 & Y_2 & Y_3 & Y_4 & Y_5 & Y_6 & Y_7 & Y_8
\end{array}
$$

$$Y_9 = H(Y_1, Y_2) \quad Y_{10} = H(Y_3, Y_4) \quad Y_{11} = H(Y_5, Y_6) \quad Y_{12} = H(Y_7, Y_8)$$

$$Y_{13} = H(Y_9, Y_{10}) \qquad\qquad Y_{14} = H(Y_{11}, Y_{12})$$

$$Y_{15} = H(Y_{13}, Y_{14})$$

Merkle public key is Y_{15}.

Signature in 8-time Merkle hash tree

First message has signature is $(S({\color{red}X_1}, r, m), Y_1, Y_2, Y_{10}, Y_{14})$.

$$
\begin{tikzcd}
\boxed{{\color{red}X_1}} \arrow[d] & X_2 \arrow[d] & X_3 \arrow[d] & X_4 \arrow[d] & X_5 \arrow[d] & X_6 \arrow[d] & X_7 \arrow[d] & X_8 \arrow[d] \\
\boxed{Y_1} \arrow[d] & \boxed{Y_2} \arrow[dl] & Y_3 \arrow[d] & Y_4 \arrow[dl] & Y_5 \arrow[d] & Y_6 \arrow[dl] & Y_7 \arrow[d] & Y_8 \arrow[dl] \\
Y_9 = H(Y_1, Y_2) \arrow[dr] & & \boxed{Y_{10} = H(Y_3, Y_4)} \arrow[dl] & & Y_{11} = H(Y_5, Y_6) \arrow[dr] & & Y_{12} = H(Y_7, Y_8) \arrow[dl] & \\
& Y_{13} = H(Y_9, Y_{10}) \arrow[drr] & & & & \boxed{Y_{14} = H(Y_{11}, Y_{12})} \arrow[dll] & & \\
& & & \boxed{{\color{blue}Y_{15} = H(Y_{13}, Y_{14})}} & & & &
\end{tikzcd}
$$

Signature in 8-time Merkle hash tree

First message has signature is $(S(X_1, r, m), Y_1, Y_2, Y_{10}, Y_{14})$.

$$
\begin{array}{cccccccc}
\boxed{X_1} & X_2 & X_3 & X_4 & X_5 & X_6 & X_7 & X_8 \\
\downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\
\boxed{Y_1} & \boxed{Y_2} & Y_3 & Y_4 & Y_5 & Y_6 & Y_7 & Y_8
\end{array}
$$

$$
Y_9 = H(Y_1, Y_2) \quad \boxed{Y_{10} = H(Y_3, Y_4)} \quad Y_{11} = H(Y_5, Y_6) \quad Y_{12} = H(Y_7, Y_8)
$$

$$
Y_{13} = H(Y_9, Y_{10}) \qquad \boxed{Y_{14} = H(Y_{11}, Y_{12})}
$$

$$
\boxed{Y_{15} = H(Y_{13}, Y_{14})}
$$

Verify by checking signature $S(X_1, r, m)$ on m against Y_1.
Link Y_1 against public key Y_{15} by computing $Y'_9 = H(Y_1, Y_2)$, $Y'_{13} = H(Y'_9, Y_{10})$, and comparing $H(Y'_{13}, Y_{14})$ with Y_{15}.

Problems and improvements

- Signature as presented is stateful – signer needs to know which X_i's have been used (and never reuse!).
- Depth of tree determines length of signature & number of signatures that can be done with public key.

Problems and improvements

- Signature as presented is stateful – signer needs to know which X_i's have been used (and never reuse!).
- Depth of tree determines length of signature & number of signatures that can be done with public key.
- Can have tree of trees to balance length of public key and signature length.
- No need to have H collision resistant.
- Winternitz signatures are more compact than Lamport signatures: To sign values in $[0, 2^k - 1]$ pick random k-bit X_0 and Y_0, compute $X_{i+1} = H(X_i), Y_{i+1} = H(Y_i)$, publish (X_{2^k-1}, Y_{2^k-1}) as key. Signature on j is (X_j, Y_{2^k-1-j}).
  Verify $H^{2^k-1-j}(X_j) \stackrel{?}{=} X_{2^k-1}$, $H^j(Y_{2^k-1-j}) \stackrel{?}{=} Y_{2^k-1}$.
- Can have stateless signatures at larger key size.

Code-based cryptography

Here only consider binary codes, i.e. codes over $\mathbb{F}_2$.

- A generator matrix of an $[n, k]$ code C is a $k \times n$ matrix G such that $C = \{\mathbf{x}\,G : \mathbf{x} \in \mathbb{F}_2^k\}$.
- The matrix G corresponds to a map $\mathbb{F}_2^k \to \mathbb{F}_2^n$ sending a message of length k to an n-bit string.
- A parity-check matrix of an $[n, k]$ code C is an $(n-k) \times n$ matrix H such that $C = \{\mathbf{c} \in \mathbb{F}_2^n : H\,\mathbf{c}^T = 0\}$.
- A systematic generator matrix is a generator matrix of the form $(I_k|Q)$ where I_k is the $k \times k$ identity matrix and Q is a $k \times (n-k)$ matrix (redundant part).
- Easy to get parity-check matrix from systematic generator matrix, use $H = (Q^T|I_{n-k})$.

Decoding problem

- The Hamming distance between two words in $\mathbb{F}_2^n$ is the number of coordinates where they differ. The Hamming weight of a word is the number of non-zero coordinates.
- The minimum distance of a linear code C is the smallest Hamming weight of a nonzero codeword in C.
- Classical decoding problem: find the closest codeword $\mathbf{x} \in C$ to a given $\mathbf{y} \in \mathbb{F}_2^n$, assuming that there is a unique closest codeword.
- In particular: Decoding a generic binary code of length n and without knowing anything about its structure requires about $2^{(0.5+o(1))n/\log_2(n)}$ binary operations (assuming a rate $\approx 1/2$)
- Coding theory deals with efficiently decodable codes.

The McEliece cryptosystem I

- Let C be a length-n binary Goppa code Γ of dimension k with minimum distance $2t+1$ where $t \approx (n-k)/\log_2(n)$; original parameters (1978) $n = 1024$, $k = 524$, $t = 50$.
- The McEliece secret key consists of a generator matrix G for Γ, an efficient t-error correcting decoding algorithm for Γ; an $n \times n$ permutation matrix P and a nonsingular $k \times k$ matrix S.
- n, k, t are public; but Γ, P, S are randomly generated secrets.
- The McEliece public key is the $k \times n$ matrix $G' = SGP$.
- Encrypt: Compute $\mathbf{m}G'$ and add a random error vector $\mathbf{e}$ of weight t and length n. Send $\mathbf{y} = \mathbf{m}G' + \mathbf{e}$.
- Decrypt: Compute $\mathbf{y}P^{-1} = \mathbf{m}G'P^{-1} + \mathbf{e}P^{-1} = \mathbf{m}SG + \mathbf{e}P^{-1}$. Use fast decoding to find $\mathbf{m}S$ and $\mathbf{m}$.

The McEliece cryptosystem II

- Encrypt: Compute $\mathbf{m}G'$ and add a random error vector $\mathbf{e}$ of weight t and length n. Send $\mathbf{y} = \mathbf{m}G' + \mathbf{e}$.
- Decrypt: Compute $\mathbf{y}P^{-1} = \mathbf{m}G'P^{-1} + \mathbf{e}P^{-1} = \mathbf{m}SG + \mathbf{e}P^{-1}$. Use fast decoding to find $\mathbf{m}S$ and $\mathbf{m}$.
- Attacker is faced with decoding $\mathbf{y}$ to nearest codeword $\mathbf{m}G'$ in the code generated by G'. This is general decoding if G' does not expose any structure.
- Wrote attack software against original McEliece parameters, decoding 50 errors in a $[1024, 524]$ code.
- Attack on a single computer with a 2.4GHz Intel Core 2 Quad Q6600 CPU would need, on average, 1400 days (2^{58} CPU cycles) to complete the attack.
- Running the software on 200 such computers would reduce the average time to one week.

Improvements

- Increase n: The most obvious way to defend McEliece's cryptosystem is to increase the code length n.
- Allow values of n between powers of 2: Get considerably better optimization of (e.g.) the McEliece public-key size.
- Use list decoding to increase t: Unique decoding is ensured by CCA2-secure variants.
- Decrease key size by using fields other than $\mathbb{F}_2$ (wild McEliece).
- Decrease key size & be faster by using other codes. Needs security analysis: some codes have too much structure.

Lattice-based crypto (1996)

- A lattice L is a discrete subgroup of $\mathbb{R}^n$: Let $B = \{\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n\}$ be a basis $L = \left\{\sum_{i=1}^n x_i \mathbf{b}_i | x_i \in \mathbb{Z}\right\}$. There are many different bases to represent this set.
- In basis close to orthogonal can easily determine closest vector to given point in $\mathbb{R}^n$.
- For a general basis finding the closest vector is hard.
- Secret key: basis with short, close to orthogonal vectors B.
- Public key: skewed basis BU, where U is unimodular matrix.
- Simplest lattice schemes look like code schemes – just using different domains for the message and error.
- Most efficient versions (NTRU, Ring-LWE) use ideal lattices; need more cryptanalysis.

Multivariate signatures (1982)

- Idea: Given $y_0, \ldots, y_{n-1} \in \mathbb{F}_2$ finding $x_0, \ldots, x_{n-1} \in \mathbb{F}_2$ with

$$\begin{aligned} y_0 &= q_0(x_0, x_1, \ldots, x_{n-1}), \\ y_1 &= q_1(x_0, x_1, \ldots, x_{n-1}), \\ &\vdots \\ y_{n-1} &= q_{n-1}(x_0, x_1, \ldots, x_{n-1}), \end{aligned}$$

  is hard, where the q_i are quadratic equations over $\mathbb{F}_2$.
- Signature: preimage of $(y_0, \ldots, y_{n-1}) = H(r, m)$ (if exists).
- Build in trapdoor by constructing the polynomials from a hidden polynomial $q(x)$ over $\mathbb{F}_{2^n} \cong \mathbb{F}_2[t]/f(t)$, using $x = \sum x_i t^i$ and sorting by powers of t. Finding $x \in \mathbb{F}_{2^n}$ with $q(x) = y$ easier.
- Hide structure by applying linear transformations, removing some equations; adding extra variables.

Advertisement

- Visit `www.pqcrypto.org` for much more material, in particular references in
  - Quantum computing
  - Hash-based cryptography
  - Code-based cryptography
  - Lattice-based cryptography
  - MQ cryptography
- Help us complete the bibliography.
- Next conference on post-quantum cryptography: PQCrypto 2013 in Limoges, France, Jun 4 – 7. Dates directly after Eurocrypt (Athens, Greece).
- Deadline is January 25 (this Friday!)