Post-Quantum Cryptography

Tanja Lange

Technische Universiteit Eindhoven

tanja@hyperelliptic.org

30 July 2009

Security

- Confidentiality: A wants to be sure that only B can read message m.
- Integrity: Data cannot be modified without authorization.
  - B wants to be sure that A sent this message m: It really was A and it exactly was message m.
  - A wants to be sure that B will accept the message only if m is unaltered.
- Non-repudiation: B wants to prove to an unlimited number of third parties C that message m is from A.
- Availability: A wants to be sure that communication will reach B.

Approaches differ considerably depending on whether or not A and B have a shared secret.

Security

- Confidentiality: A wants to be sure that only B can read message m.
- Integrity: Data cannot be modified without authorization.
  - B wants to be sure that A sent this message m: It really was A and it exactly was message m.
  - A wants to be sure that B will accept the message only if m is unaltered.
- Non-repudiation: B wants to prove to an unlimited number of third parties C that message m is from A.
- Availability: A wants to be sure that communication will reach B.

Approaches differ considerably depending on whether or not A and B have a shared secret.

Confidentiality

- Long preshared secret: Use one-time pad.
- Short preshared secret:
  Use symmetric cipher – block cipher or stream cipher depending on type of data, e.g. AES-128 or eSTREAM finalists.
- No preshared secret:
  Use public-key encryption in a possibly hybrid setting, e.g. use public-key encryption to encrypt a key that is subsequently used in symmetric cipher.

  An example is ECC-512 (elliptic curve with 512 bits) to determine shared key used with AES-256.

  Current recommendation (security and performance): ECC-256 combined with AES-128.

Integrity

- Long preshared secret: Use Wegman-Carter, e.g. Poly1305.
- Short preshared secret:
  Use Message Authentication Codes (MACs), e.g. HMAC-SHA-1 or Poly1305-AES, and send $m||\text{MAC}(m)$.
- No preshared secret:
  Use public key signatures, e.g. ECDSA with ECC-256 and SHA-256, and send $m||\text{Sig}(m)$.

Important difference:
Signatures additionally provide non-repudiation. If B wants to prove to the world that A sent a particular message, public-key signatures are needed. Non-repudiation is inherently linked to public availability of verification data. To achieve non-repudiation no benefit in preshared secret.

No preshared secrets were harmed for this talk.

Diffie-Hellman key exchange

Parameters

$P, \ell = |\langle P \rangle|$

some finite group

Alice

1. secretly generates $a < |\langle P \rangle|$
2. computes $Q_1 = [a]P$
3. transmits Q_1
4. computes $[a]Q_2$

Bob

1. secretly generates $b < |\langle P \rangle|$
2. computes $Q_2 = [b]P$
3. transmits Q_2
4. computes $[b]Q_1$

$$[a]Q_2 = [ab]P = [b]Q_1$$

Common Key: the group element $k = [(ab)]P \in G$ can be used in symmetric encryption.

Some groups I

Consider $(\mathbb{F}_p, +)$: integers modulo prime p with operation $+$. Nobody would use this!

- Take generator, e.g. 42 then compute $Q_1 \equiv a \cdot 42 \bmod p$. This takes $O(\log^{1+\varepsilon} p)$ operations.
- Break it: compute $42^{-1} \bmod p$ and obtain $a \equiv Q_1 \cdot 42^{-1} \bmod p$. This takes $O(\log^{1+\varepsilon} p)$ operations.
- Encryption and break have same complexity, system is broken!
- The constants hidden in the O are different, so it can offer security for some (short) time; could amplify time needed for the attacker by concurrently using 1000000 instances. But: we want better security!

Some groups II

Consider $(\mathbb{F}_p^*, \cdot)$: integers modulo prime p with operation multiplication.

- Set up: Choose p with large prime ℓ dividing $p-1$. Find an element $g \in \mathbb{F}_p^*$ with order ℓ, i.e. $g^\ell \equiv 1 \bmod p$.
- Public key: Compute $Q_1 \equiv g^a \bmod p$.
  This takes $O(\log^{2+\varepsilon} p)$ operations.
- Break it: Best known method to obtain a is index calculus which takes
  $$O(\exp(c(\log p)^{1/3}(\log\log p)^{2/3})).$$
- Encryption and break have different complexity, i.e. security grows faster than execution time.
- To obtain security of 2^k (attacker needs 2^k operations) choose $p \approx 2^{k^3}$.

Some groups III

Consider $E(\mathbb{F}_p)$: elliptic curve defined over $\mathbb{F}_p$ with elliptic curve addition and neutral element $\mathcal{O}$.

- Set up: Choose elliptic curve so that group order has large prime divisor $\ell \sim p$, find point $P \in E(\mathbb{F}_p)$ with order ℓ, i.e. $[\ell]P = \mathcal{O}$.
- Public key: Compute $Q_1 = [a]P$.
  This takes $O(\log^{2+\varepsilon} p)$ operations.
- Break it: Best known method to obtain a is Pollard's rho method which takes $O(\sqrt{\ell})$.
- Encryption and break have different complexity, i.e. security grows faster than execution time.
- To obtain security of 2^k (attacker needs 2^k operations) choose curve with $p, \ell \approx 2^{2k}$.

(Asymptotic) comparison

- $(\mathbb{F}_p, +)$: For security 2^k needs p of size 2^{2^k} and takes time 2^k to execute. Useless.
- $(\mathbb{F}_p, \cdot)$: For security 2^k needs p of size 2^{k^3} and takes time k^6 to execute. Useful.
- $E(\mathbb{F}_p)$: For security 2^k needs ℓ of size 2^{2k} and takes time k^2 to execute. More useful.
- Given the constants there are regions in which $(\mathbb{F}_p, \cdot)$ performs better than $E(\mathbb{F}_p)$ for a fixed security level. Asymptotically (and in practice) elliptic curves win.
- These are our usual considerations when studying public key primitives; new ones must be faster than ECC at same security level. Speed and size matter!
- This all changes for really long-term security!

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

Quantum computer

Compensate for Grover by $\approx$ 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”

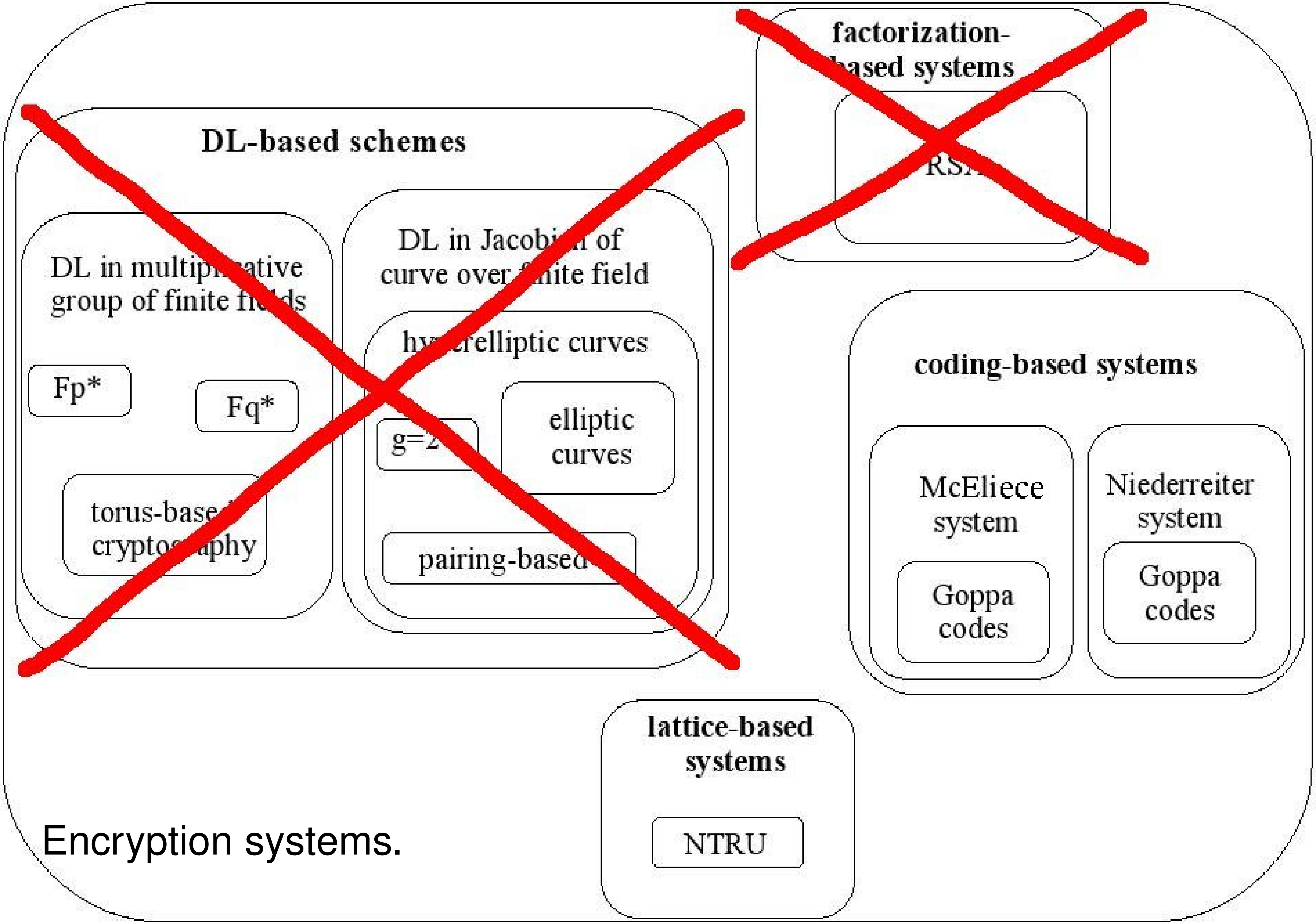
Public-key encryption:

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

Public-key signatures:

- Multivariate-quadratic-equations signatures (e.g. HFEv-)
- Hash-based signatures (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.


factorization-
DL-based schemes
Fp*
Fq*
elliptic curves
pairing-based
coding-based systems
McEliece system
Goppa codes
Niederreiter system
Goppa codes
lattice-based systems
NTRU
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.
- ECC-encrypted long-term-confidential documents (e.g. health records, state secrets) become readable once quantum computers are available.
- ECC-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 need better study of parameters, faster implementations, and better embedding into protocols.

Post-quantum cryptography

- We study cryptographic systems that run on conventional computers, are secure against attacks with conventional computers, and remain secure under attacks with quantum computers.
- Could develop cryptosystems for quantum computers – but there won't be a market too soon (attacker has big quantum computer long before wide-spread use).
- 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.

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

Hash-based signatures

- Evolved from Lamport one-time signature scheme.
- Standardize a $4k$-bit hash function H.
- Signer's secret key X: $8k$ strings $x_1[0], x_1[1], \ldots, x_{4k}[0], x_{4k}[1]$, each $4k$ bits. Total: $32k^2$ bits.
- Signer's public key Y: $8k$ strings $y_1[0], y_1[1], \ldots, y_{4k}[0], y_{4k}[1]$, each $4k$ bits, computed as $y_i[b] = H(x_i[b])$. Total: $32k^2$ bits.
- Signature $S(X, r, m)$ of a message m: $r, x_1[h_1], \ldots, x_{4k}[h_{4k}]$ where $H(r, m) = (h_1, \ldots, h_{4k})$.
- Must never use secret key more than once.
- Collision resistance not required from H.
- 1979 Merkle extends to more signatures.

4-time Merkle hash tree

Four Lamport one-time keys Y_1, Y_2, Y_3, Y_4 with corresponding X_1, X_2, X_3, X_4

$$\begin{tikzcd}
X_1 \arrow[d] & & X_2 \arrow[d] & & X_3 \arrow[d] & & X_4 \arrow[d] \\
Y_1 \arrow[dr] & & Y_2 \arrow[dl] & & Y_3 \arrow[dr] & & Y_4 \arrow[dl] \\
& Y_5 = H(Y_1, Y_2) \arrow[drr] & & & & Y_6 = H(Y_3, Y_4) \arrow[dll] & \\
& & & Y_7 = H(Y_5, Y_6) & & &
\end{tikzcd}$$

Merkle public key is Y_7.

4-time Merkle hash tree

First message m_1: signature $S(X_1, r_1, m_1), Y_1, Y_2, Y_5, Y_6$.

$$
\begin{tikzcd}
X_1 \arrow[d] & X_2 \arrow[d] & X_3 \arrow[d] & X_4 \arrow[d] \\
Y_1 \arrow[dr] & Y_2 \arrow[d] & Y_3 \arrow[dr] & Y_4 \arrow[d] \\
 & Y_5 = H(Y_1, Y_2) \arrow[dr] & & Y_6 = H(Y_3, Y_4) \arrow[dl] \\
 & & Y_7 = H(Y_5, Y_6) &
\end{tikzcd}
$$

Receiver's verification:
$Y_7 = H(Y_5, Y_6)$;
$Y_5 = H(Y_1, Y_2)$;
Y_1 matches one-time signature $S(X_1, r_1, m_1)$.

4-time Merkle hash tree

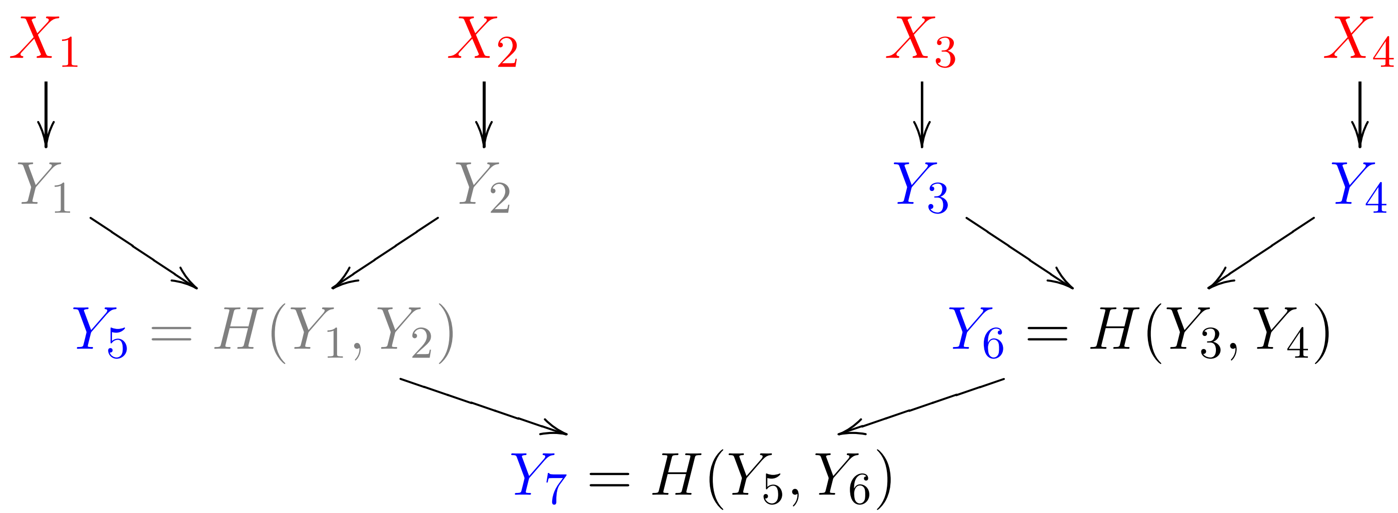
Second message m_2: signature $S(X_2, r_2, m_2), Y_2, Y_1, Y_5, Y_6$.

$$
\begin{tikzcd}
X_1 \arrow[d] & & X_2 \arrow[d] & & X_3 \arrow[d] & & X_4 \arrow[d] \\
Y_1 \arrow[dr] & & Y_2 \arrow[dl] & & Y_3 \arrow[dr] & & Y_4 \arrow[dl] \\
& Y_5 = H(Y_1, Y_2) \arrow[drr] & & & & Y_6 = H(Y_3, Y_4) \arrow[dll] & \\
& & & Y_7 = H(Y_5, Y_6) & & &
\end{tikzcd}
$$

Receiver's verification:
$Y_7 = H(Y_5, Y_6)$;
$Y_5 = H(Y_1, Y_2)$ (after trying $H(Y_2, Y_1)$);
Y_2 matches one-time signature $S(X_2, r_2, m_2)$.

4-time Merkle hash tree

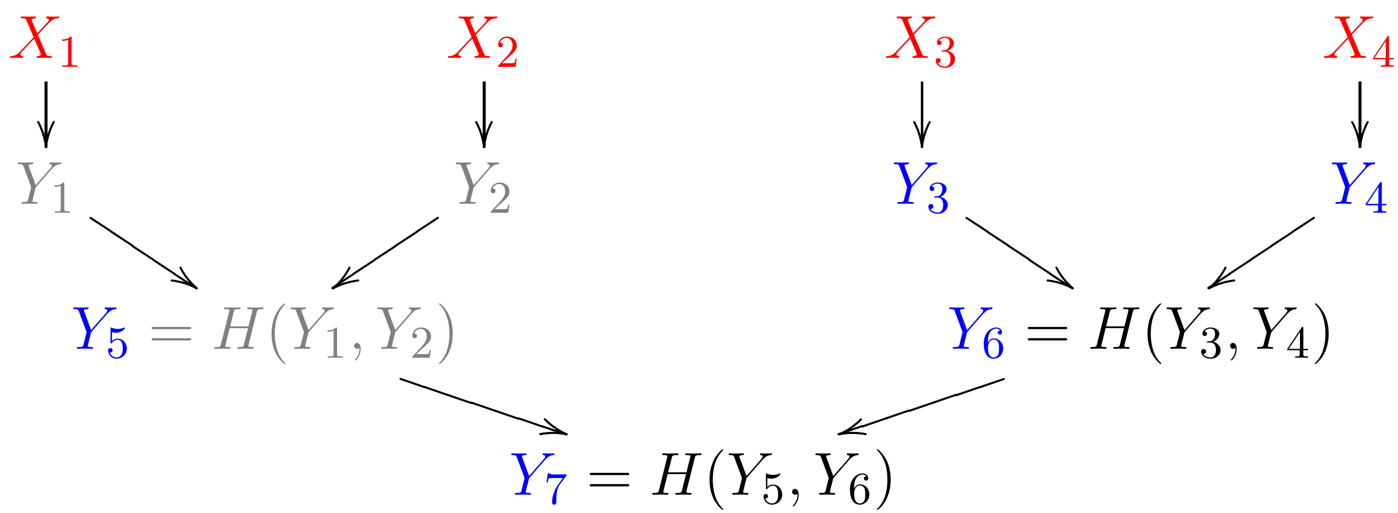
Third message m_3: signature $S(X_3, r_3, m_3), Y_3, Y_4, Y_6, Y_5$.



Receiver's verification:
$Y_7 = H(Y_5, Y_6)$;
$Y_6 = H(Y_3, Y_4)$;
Y_3 matches one-time signature $S(X_3, r_3, m_3)$.

4-time Merkle hash tree

Fourth message m_4: signature $S(X_4, r_4, m_4), Y_4, Y_3, Y_6, Y_5$.



Receiver's verification:

$Y_7 = H(Y_5, Y_6)$;

$Y_6 = H(Y_3, Y_4)$;

Y_4 matches one-time signature $S(X_4, r_4, m_4)$.

Multivariate signatures

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

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

  is hard, where the q_i are quadratic equation 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 $f(x) = y$ easier.
- Hide structure by applying linear transformations, removing some equations; adding extra variables.

Linear codes

We 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 decodeable codes.

The McEliece cryptosystem I

- Proposed by McEliece in 1978; system and parameter sizes proposed back then need 2^{60} operations to break; done last year (Bernstein-L.-Peters).
- 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: $n = 1024$, $k = 524$, $t = 50$).
- 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 $G' = SGP$.

The McEliece cryptosystem II

- McEliece encryption: Compute $\mathbf{m}G'$ and add a random error vector $\mathbf{e}$ of weight t and length n.
- Encryption of a message $\mathbf{m}$ of length k: Compute $\mathbf{m}G'$ and add a random error vector $\mathbf{e}$ of weight t and length n. Send $\mathbf{y} = \mathbf{m}G' + \mathbf{e}$.
- McEliece decryption using secret key: Compute $\mathbf{y}P^{-1} = \mathbf{m}G'P^{-1} + \mathbf{e}P^{-1} = \mathbf{m}SG + \mathbf{e}P^{-1}$.
  Use decoding algorithm to find $\mathbf{m}S$ and thereby $\mathbf{m}$.
- 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.
- For codes other than Goppa codes often original code could be reconstructed from G' allowing faster decoding.

Attacks on the McEliece PKC

- Most effective attack against the McEliece cryptosystem (with binary Goppa code) is information-set decoding.
- Many variants:
  McEliece (1978), Leon (1988), Lee and Brickell (1988), Stern (1989), van Tilburg (1990), Canteaut and Chabanne (1994), Canteaut and Chabaud (1998), Canteaut and Sendrier (1998), Bernstein-L.-Peters, and Bernstein-L.-Peters-van Tilborg.
- Quantum attacks (Grover applies) by Overbeck and Sendrier.
- Many attacks on code-based crypto when codes other than Goppa codes are used. Some alternatives available that give shorter keys.

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 2010 in Darmstadt, Germany, May 25-28. Dates directly before Eurocrypt (Nice, France). Location close to Frankfurt airport.