# Code-based cryptography

Tanja Lange

Technische Universiteit Eindhoven

tanja@hyperelliptic.org

13 July 2011

joint work with Daniel J. Bernstein and Christiane Peters

# 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 (massive for VSH)

Compensate for Grover by $\approx$ 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 (massive for VSH)

Sorry,
no picture
available
quite yet.

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"

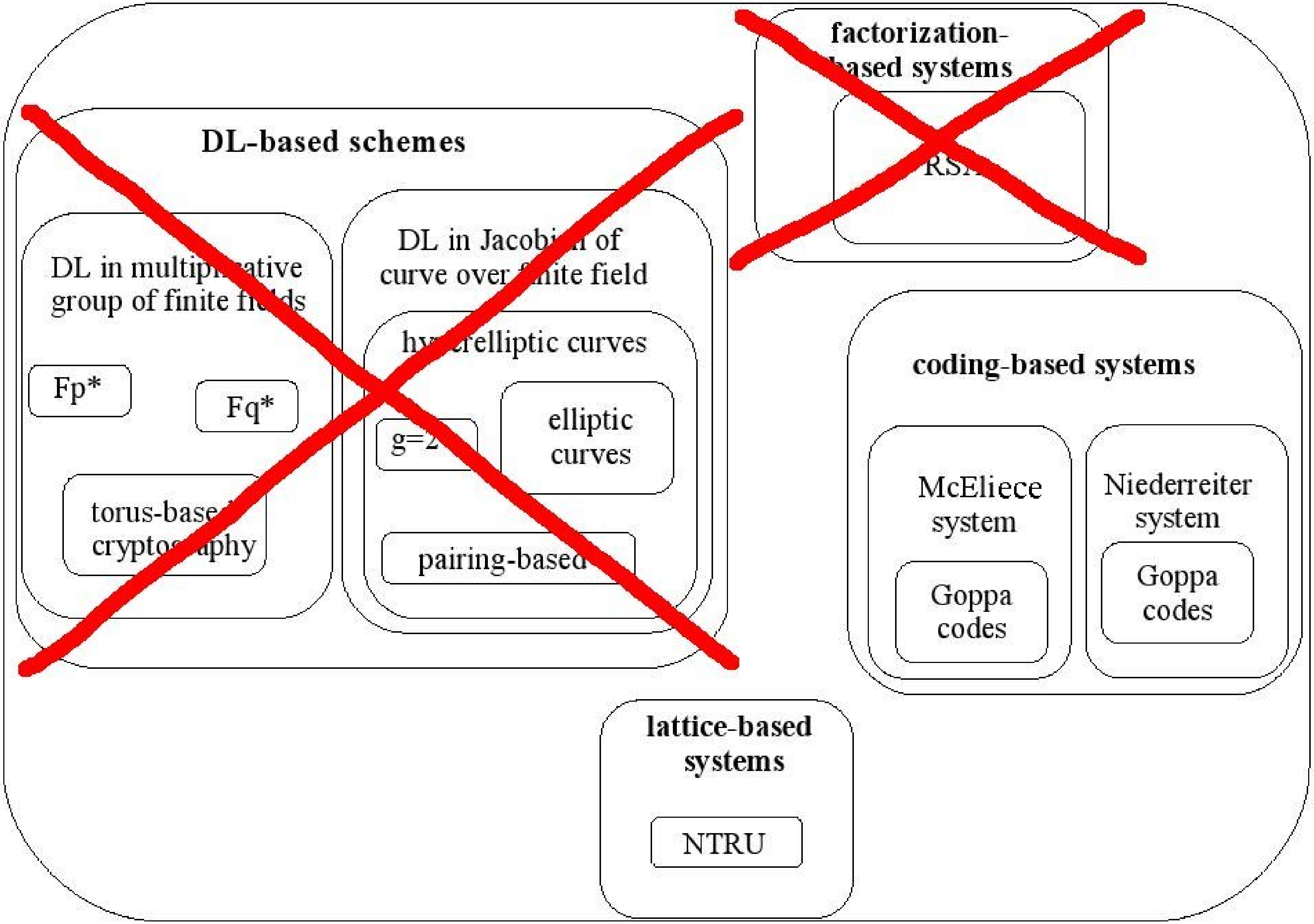
Public-key encryption:

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

Public-key signatures:

- Multivariate-quadratic-equations cryptography (e.g. $\text{HFE}^-$)
- 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.
- 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 need better study of parameters, faster implementations, and better embedding into protocols.

# 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.
- Could potentially develop cryptosystems for quantum computers – but there won't be a market too soon.

# Warnings

- The following describes “text-book” versions of the cryptosystems.
- E.g. the plaintexts are not randomized, so an attacker could easily check his guess: Did Alice say ’yes’ or ’no’?
- There exist CCA2-secure versions. They ensure that breaking the system boils down to breaking the mathematical problem.
- But they clutter the exposition somewhat.
- Main differences:
  - encryption randomizes plaintext;
  - effective length of plaintext is reduced (make space for randomness).
- Make sure to use them when deploying the system!

# Linear codes – notation

- A generator matrix of an $[n, k]$ code $C$ is a $k \times n$ matrix $G$ such that $C = \{\mathbf{x}\, G : \mathbf{x} \in \mathbb{F}_q^k\}$.
- The matrix $G$ corresponds to a map $\mathbb{F}_q^k \rightarrow \mathbb{F}_q^n$ 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 $C = \{\mathbf{c} \in \mathbb{F}_q^n : H\, \mathbf{c}^T = 0\}$.
- 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 $H = (-Q^T|I_{n-k})$.
- The minimum distance of a linear code $C$ is the smallest Hamming weight of a nonzero codeword in $C$.

# The McEliece cryptosystem

First part of talk focuses on generic attacks, so no special properties assumed about the code.

- Let $C$ be an $[n, k]$ code with minimum distance $\geq 2w + 1$ given by a generator matrix $G$. The McEliece public key is the $k \times n$ matrix $G$ and $w$.
- The McEliece secret key consists of an efficient $w$-error correcting decoding algorithm for $C$.
- McEliece encryption of a message $\mathbf{m}$ of length $k$: Compute $\mathbf{m}G$ and add a random error vector $\mathbf{e}$ of weight $w$ and length $n$. Send $\mathbf{y} = \mathbf{m}G + \mathbf{e}$.
- Finding $\mathbf{e}$ means finding $\mathbf{m}$.

# The Niederreiter cryptosystem

- Let $C$ be an $[n, k]$ code with minimum distance $\geq 2w + 1$ given by a parity-check matrix $H$. The Niederreiter public key is the $(n - k) \times n$ matrix $H$ and $w$.
- The Niederreiter secret key consists of an efficient $w$-error correcting decoding algorithm for $C$.
- Niederreiter encryption of a message $m$, uniquely encoded as a vector $\mathbf{m}$ of weight $w$ and length $n$: Compute and send $\mathbf{s} = H\mathbf{m}^T$.
- Problems are related: McEliece's $\mathbf{y}$ gives syndrome $H\mathbf{y}^T = H(\mathbf{m}G + \mathbf{e})^T = H\mathbf{e}^T$.
- Message in Niederreiter corresponds to error term in McEliece.

# Attacks on the code-based systems

- Encryption of message $m$ is $\mathbf{y} = \mathbf{m}G + \mathbf{e}$ or $\mathbf{s} = H\mathbf{m}^T$.
- Attacker is faced with decoding $\mathbf{y}$ to nearest codeword $\mathbf{m}G$ or syndrome-decoding $\mathbf{s}$. This is decoding a generic code.
- Most effective attack against these cryptosystems (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, Lange, and Peters (2008), and Finiasz and Sendrier (2009).
- New result: Ball-collision decoding with proper analysis and exponential improvement over previous results.

# Search for low weight words vs. decoding

- McEliece ciphertext $\mathbf{y} \in \mathbb{F}_q^n$ has distance $w$ from a unique closest codeword $\mathbf{c} = \mathbf{m}G$ in a code $C$ which has minimum distance at least $2w + 1$.
- Find $\mathbf{e}$ of weight $w$ such that $\mathbf{c} = \mathbf{y} - \mathbf{e}$:
  - append $\mathbf{y}$ to the list of generators
  - and form a generator matrix for $C + \{0, \mathbf{y}\}$.

  Then

  $$\mathbf{e} = (\mathbf{m}, 1) \left( \frac{G}{\mathbf{m}G + \mathbf{e}} \right)$$

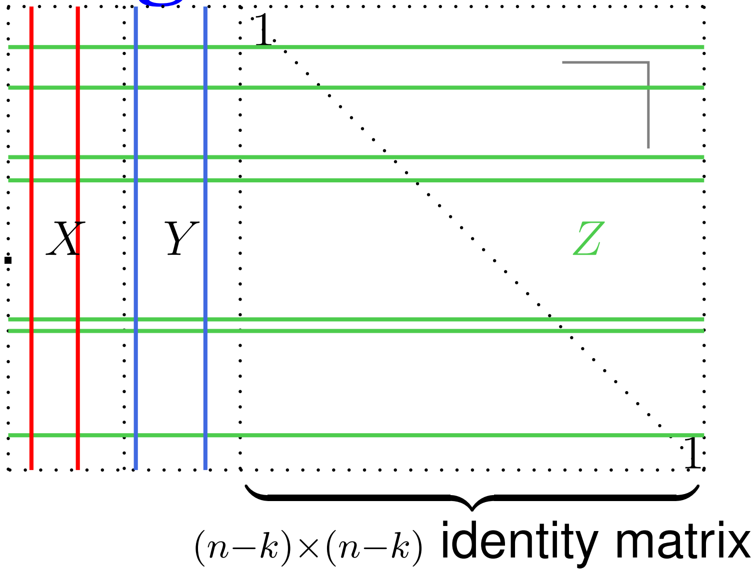
  is a codeword in $C + \{0, \mathbf{y}\}$; and it is the only weight-$w$ word.
- Bottleneck in all of these attacks is finding the weight-$w$ codeword in $C + \{0, \mathbf{y}\}$. This code has dimension $k + 1$.

# Stern's attack

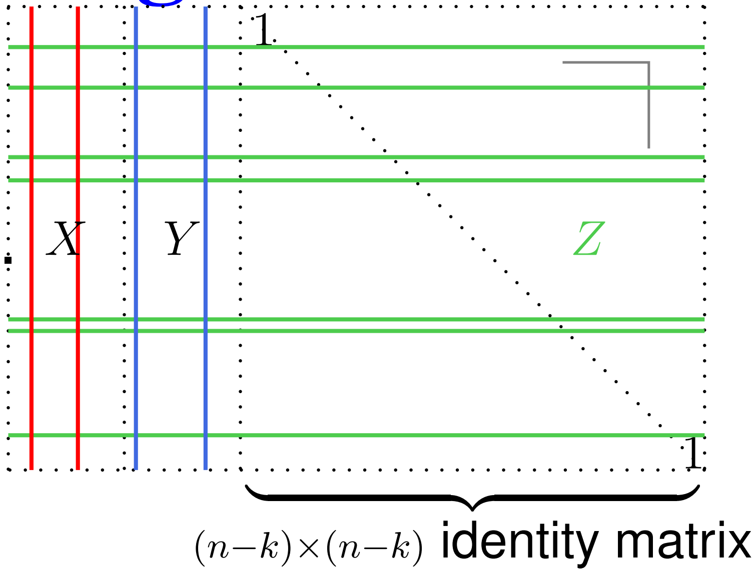
- Focus on binary codes, i.e. codes over $\mathbb{F}_2$, here. Generalization (and much more) done by Peters (2010).
- Given $w \geq 0$ and an $(n-k) \times n$ parity check matrix $H$ for a binary $[n, k]$ code $C$. Goal is to find $\mathbf{c} \in C$ of weight $w$.
- Construct $\mathbf{c}$ by looking for exactly $w$ columns of $H$ which add up to $0$.
  - Select $n-k$ columns and transfer these into identity matrix.
  - Choose two disjoint subsets $X, Y$ among the other columns of $H$ and a subset $Z$ among the first.
  - Search for words having exactly $p, p, 0$ ones in those column sets $X, Y, Z$ and exactly $w-2p$ ones in the remaining columns.
  - Iterate until such a word is found for some $X, Y, Z$.

# One iteration of Stern's algorithm

- Select $n-k$ linearly independent columns; apply elementary row operations to get the identity matrix.
- Form a set $Z$ of $\ell$ rows (green).
- Divide remaining $k$ columns into two subsets $X$ and $Y$.

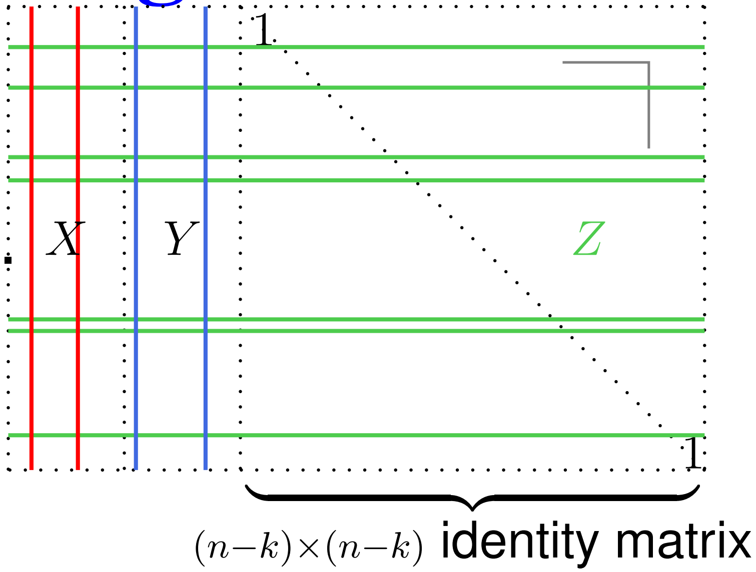


# One iteration of Stern's algorithm




- Select $n-k$ linearly independent columns; apply elementary row operations to get the identity matrix.
- Form a set $Z$ of $\ell$ rows (green).
- Divide remaining $k$ columns into two subsets $X$ and $Y$.
- For every size-$p$ subset $A$ of $X$ compute the $\ell$-bit vector $\pi(A)$ by adding up the columns of $H' = (H_{i,j})_{i\in Z, j\in A}$ (red). Similarly, compute $\pi(B)$ (blue).
- For each collision $\pi(A) = \pi(B)$ compute the sum of the $2p$ columns in $A \cup B$. This sum is an $(n-k)$-bit vector.

# One iteration of Stern's algorithm



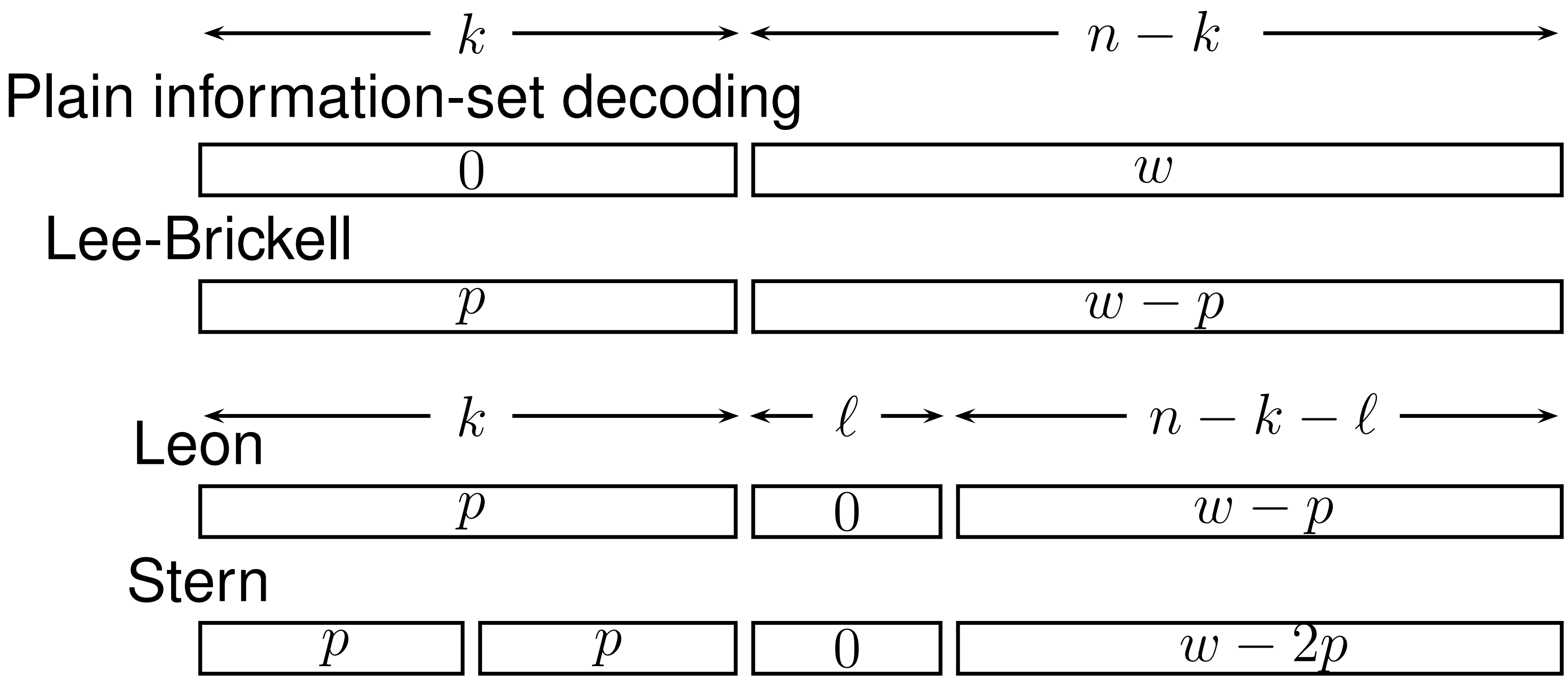

- Select $n-k$ linearly independent columns; apply elementary row operations to get the identity matrix.
- Form a set $Z$ of $\ell$ rows (green).
- Divide remaining $k$ columns into two subsets $X$ and $Y$.
- For every size-$p$ subset $A$ of $X$ compute the $\ell$-bit vector $\pi(A)$ by adding up the columns of $H' = (H_{i,j})_{i\in Z, j\in A}$ (red). Similarly, compute $\pi(B)$ (blue).
- For each collision $\pi(A) = \pi(B)$ compute the sum of the $2p$ columns in $A \cup B$. This sum is an $(n-k)$-bit vector.
- **If** the sum has weight $w-2p$, we obtain $0$ by adding the corresponding $w-2p$ columns in the $(n-k)\times(n-k)$ submatrix. **Else** select $n-k$ new columns.

# Running time in practice

- Wrote attack software against original McEliece parameters, decoding 50 errors in a $[1024, 524]$ code.
- Lots of improvements stated in 2008 paper.
- 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.
- About 200 computers involved, with about 300 cores.
- Most of the cores put in far fewer than 90 days of work; some of which were considerably slower than a Core 2.
- Computation used about 8000 core-days.
- Error vector found by Walton cluster at SFI/HEA Irish Centre of High-End Computing (ICHEC).

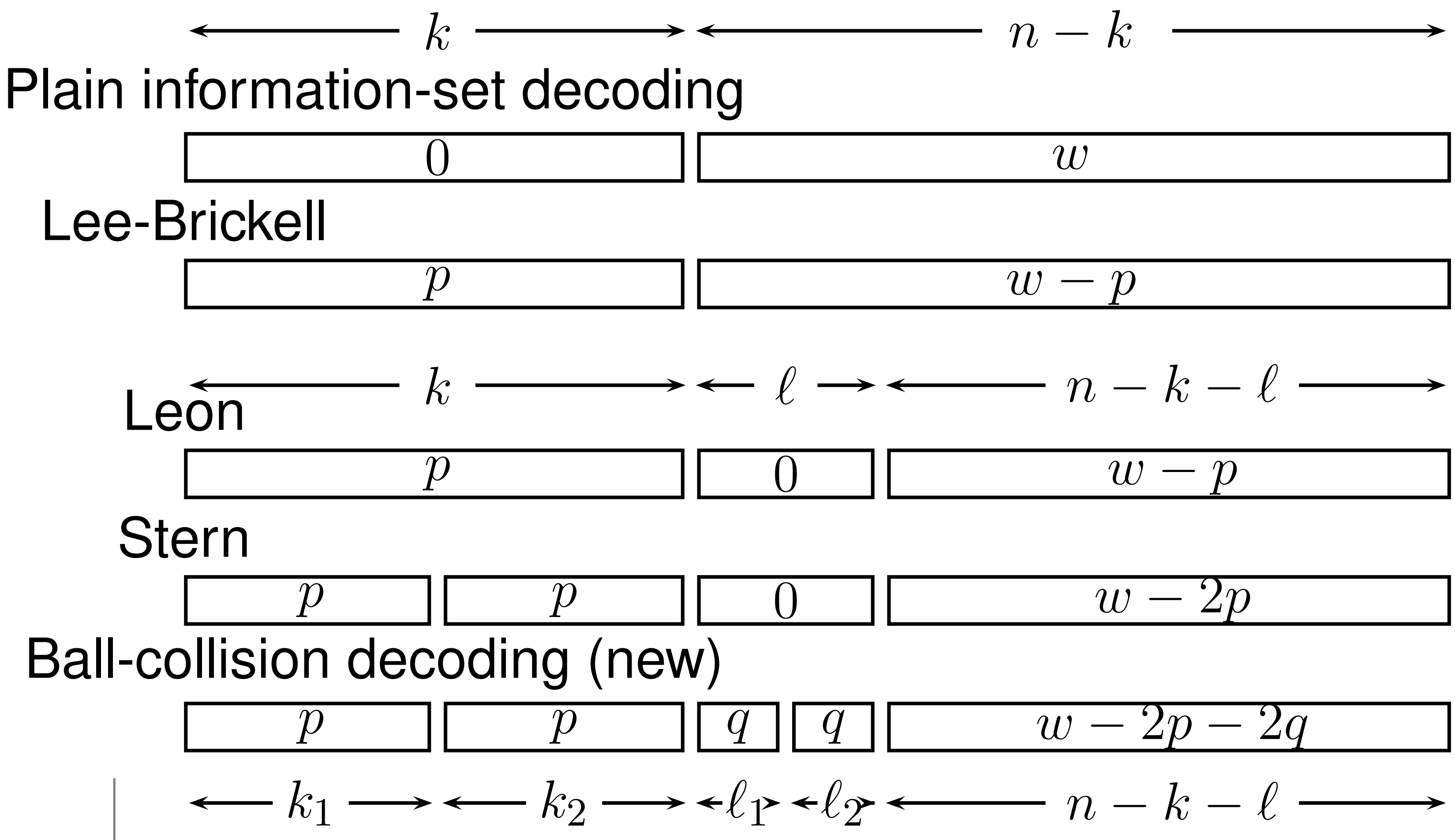
# Information-set decoding

Methods differ in where the errors are allowed to be.



# Information-set decoding

Methods differ in where the errors are allowed to be.



# Ball-collision decoding

- Do not insist on having 0 errors in $Z$ but allow $2q$ errors in this stretch.
- This helps! If $\ell$ is optimized in Stern's algorithm (and relatives) it grows with $n$, so Stern is successful if an increasingly large stretch of the vector is error free. In hindsight, that's a bad idea.
- This algorithm is inspired by supercode decoding (Barg, Krouk, van Tilborg; 1999). Supercode decoding enforces an increasingly detailed error pattern on the code but the analysis is wrong (in the appendix).
- Decoding of generic codes takes time $2^{(\alpha(R,W)+o(1))n}$, for rate $k/n \to R$ and error-rate $w/n \to W$ for $n \to \infty$.
- Ball-collision decoding achieves a smaller $\alpha$, i.e. an exponential speedup over Stern's algorithm!

# What codes are behind this?

- McEliece suggested to use classical Goppa codes $\Gamma$.
- If you know algebraic geometry Goppa codes, these are included as genus-0 codes.
- These codes are efficiently decodeable – so how can there be a secret?
- The McEliece secret key consists of a generator matrix $G$ for $\Gamma$, an efficient $t$-error correcting decoding algorithm for $\Gamma$; an $n \times n$ permutation matrix $P$ and a nonsingular $k \times k$ matrix $S$.
- The McEliece public key is the $k \times n$ matrix $G' = SGP$. Systematic key size $k(n-k)$ elements in $\mathbb{F}_q$.
- $n, k, t$ are public; but $\Gamma$, $P$, $S$ are randomly generated secrets. $G'$ efficiently hides the structure of $G$.

# Goppa codes

- Fix a prime power $q$; a positive integer $m$; a positive integer $n \leq q^m$; an integer $t < n/m$;
- fix distinct elements $a_1, \ldots, a_n$ in $\mathbb{F}_{q^m}$;
- fix polynomial $g(x)$ in $\mathbb{F}_{q^m}[x]$, $\deg(g) = t$ s.t. $g(a_i) \neq 0$.
- Denote by $\Gamma_{q^m}(a_1, \ldots, a_n, g)$ the linear code consisting of all words $c = (c_1, \ldots, c_n)$ in $\mathbb{F}_{q^m}^n$ satisfying

$$\sum_{i=1}^{n} \frac{c_i}{x - a_i} \equiv 0 \pmod{g(x)}$$

- The Goppa code $\Gamma_q(a_1, \ldots, a_n, g)$ with Goppa polynomial $g(x)$ and support $a_1, \ldots, a_n$ is the restriction of $\Gamma_{q^m}(a_1, \ldots, a_n, g)$ to $\mathbb{F}_q$.
- Has dimension $\geq n - mt$ and minimum distance $\geq t + 1$.

# Dependence on $q$

- Choose $n, m, t$ so that decoding in $G'$ is hard; larger $n$ and $m$ mean harder to decode codes – but also larger keys.
- For $q = 2$ the minimum distance is at least $2t + 1$. Patterson's algorithm decodes efficiently up to $t$ errors.
- For larger $q$ no such construction known, so can correct only up to $t/2$ errors.
- Generalizations of information-set decoding to $q > 2$ by Peters show that key sizes are larger for $2 < q < 31$ than for $q = 2$ at the same level of security (hardness of decoding $G'$).
- For $q \geq 31$ one can achieve smaller public keys.
- Haven't done implementation yet, so don't have speed comparison for $q = 2$ vs. $q = 31$.

# Wild Goppa Codes

- Theorem by Sugiyama, Kasahara, Hirasawa, and Namekawa:
  Let $q$ be a prime power. Let $m$ be a positive integer. Let $n$ be an integer with $1 \leq n \leq q^m$. Let $a_1, a_2, \ldots, a_n$ be distinct elements of $\mathbb{F}_{q^m}$. Let $g$ be a monic squarefree polynomial in $\mathbb{F}_{q^m}[x]$ coprime to $(x - a_1) \cdots (x - a_n)$.
  Then $\Gamma_q(a_1, a_2, \ldots, a_n, g^{q-1}) = \Gamma_q(a_1, a_2, \ldots, a_n, g^q)$.
- We call these codes wild Goppa codes in analogy to wild ramification in extension fields.
- Put $\deg(g) = t$. Code with polynomial $g^{q-1}$ is the same as code with $g^q$, so it has minimum distance $\geq qt + 1$ instead of $\geq (q-1)t + 1$.
- This covers well-known result for $q = 2$; gives an important improvement for $q = 3, q = 4, \ldots$

# Proof

$\Leftarrow$ If $\sum_i c_i/(x-a_i) \equiv 0 \bmod g^q$ then $\sum_i c_i/(x-a_i) \equiv 0 \bmod g^{q-1}$.

# Proof

$\Leftarrow$ If $\sum_i c_i/(x - a_i) \equiv 0 \bmod g^q$ then $\sum_i c_i/(x - a_i) \equiv 0 \bmod g^{q-1}$.

$\Rightarrow$ Conversely, let $\mathbf{c} \in \mathbb{F}_q^n$ with $\sum_i c_i/(x - a_i) = 0$ in $\mathbb{F}_{q^m}[x]/g^{q-1}$.
Let $g$ split completely in $k[x] \supseteq \mathbb{F}_{q^m}[x]$.

# Proof

$\Leftarrow$ If $\sum_i c_i/(x - a_i) \equiv 0 \bmod g^q$ then $\sum_i c_i/(x - a_i) \equiv 0 \bmod g^{q-1}$.

$\Rightarrow$ Conversely, let $\mathbf{c} \in \mathbb{F}_q^n$ with $\sum_i c_i/(x - a_i) = 0$ in $\mathbb{F}_{q^m}[x]/g^{q-1}$.
Let $g$ split completely in $k[x] \supseteq \mathbb{F}_{q^m}[x]$. Then
$\sum_i c_i/(x - a_i) = 0$ in $k[x]/(x - r)^{q-1}$ for each factor $x - r$ of $g$.

# Proof

$\Leftarrow$ If $\sum_i c_i/(x-a_i) \equiv 0 \bmod g^q$ then $\sum_i c_i/(x-a_i) \equiv 0 \bmod g^{q-1}$.

$\Rightarrow$ Conversely, let $\mathbf{c} \in \mathbb{F}_q^n$ with $\sum_i c_i/(x-a_i) = 0$ in $\mathbb{F}_{q^m}[x]/g^{q-1}$.

Let $g$ split completely in $k[x] \supseteq \mathbb{F}_{q^m}[x]$. Then

$\sum_i c_i/(x-a_i) = 0$ in $k[x]/(x-r)^{q-1}$ for each factor $x-r$ of $g$.

Use $\frac{1}{x-a_i} = -\frac{1}{a_i-r} - \frac{x-r}{(a_i-r)^2} - \frac{(x-r)^2}{(a_i-r)^3} - \cdots$ to see

$$\sum_i \frac{c_i}{a_i-r} + (x-r)\sum_i \frac{c_i}{(a_i-r)^2} + (x-r)^2 \sum_i \frac{c_i}{(a_i-r)^3} + \cdots = 0$$

in $k[x]/(x-r)^{q-1}$;

# Proof

$\Leftarrow$ If $\sum_i c_i/(x-a_i) \equiv 0 \bmod g^q$ then $\sum_i c_i/(x-a_i) \equiv 0 \bmod g^{q-1}$.

$\Rightarrow$ Conversely, let $\mathbf{c} \in \mathbb{F}_q^n$ with $\sum_i c_i/(x-a_i) = 0$ in $\mathbb{F}_{q^m}[x]/g^{q-1}$.
Let $g$ split completely in $k[x] \supseteq \mathbb{F}_{q^m}[x]$. Then
$\sum_i c_i/(x-a_i) = 0$ in $k[x]/(x-r)^{q-1}$ for each factor $x-r$ of $g$.
Use $\frac{1}{x-a_i} = -\frac{1}{a_i-r} - \frac{x-r}{(a_i-r)^2} - \frac{(x-r)^2}{(a_i-r)^3} - \cdots$ to see

$$\sum_i \frac{c_i}{a_i-r} + (x-r)\sum_i \frac{c_i}{(a_i-r)^2} + (x-r)^2 \sum_i \frac{c_i}{(a_i-r)^3} + \cdots = 0$$

in $k[x]/(x-r)^{q-1}$; i.e., $\sum_i c_i/(a_i-r) = 0$, $\sum_i c_i/(a_i-r)^2 = 0$, $\ldots$, $\sum_i c_i/(a_i-r)^{q-1} = 0$.
Take the $q$th power of the equation $\sum_i c_i/(a_i-r) = 0$, and use $c_i \in \mathbb{F}_q$, to obtain $\sum_i c_i/(a_i-r)^q = 0$. Work backwards to see that $\sum_i c_i/(x-a_i) = 0$ in $k[x]/(x-r)^q$.

# Proof

$\Leftarrow$ If $\sum_i c_i/(x-a_i) \equiv 0 \bmod g^q$ then $\sum_i c_i/(x-a_i) \equiv 0 \bmod g^{q-1}$.

$\Rightarrow$ Conversely, let $\mathbf{c} \in \mathbb{F}_q^n$ with $\sum_i c_i/(x-a_i) = 0$ in $\mathbb{F}_{q^m}[x]/g^{q-1}$.
Let $g$ split completely in $k[x] \supseteq \mathbb{F}_{q^m}[x]$. Then
$\sum_i c_i/(x-a_i) = 0$ in $k[x]/(x-r)^{q-1}$ for each factor $x-r$ of $g$.
Use $\frac{1}{x-a_i} = -\frac{1}{a_i-r} - \frac{x-r}{(a_i-r)^2} - \frac{(x-r)^2}{(a_i-r)^3} - \cdots$ to see

$$\sum_i \frac{c_i}{a_i-r} + (x-r)\sum_i \frac{c_i}{(a_i-r)^2} + (x-r)^2 \sum_i \frac{c_i}{(a_i-r)^3} + \cdots = 0$$

in $k[x]/(x-r)^{q-1}$; i.e., $\sum_i c_i/(a_i-r) = 0$, $\sum_i c_i/(a_i-r)^2 = 0$, $\ldots$, $\sum_i c_i/(a_i-r)^{q-1} = 0$.
Take the $q$th power of the equation $\sum_i c_i/(a_i-r) = 0$, and use $c_i \in \mathbb{F}_q$, to obtain $\sum_i c_i/(a_i-r)^q = 0$. Work backwards to see that $\sum_i c_i/(x-a_i) = 0$ in $k[x]/(x-r)^q$.
All $x-r$ are distinct, thus $\sum_i c_i/(x-a_i) = 0$ in $k[x]/g^q$.

# Generalization of wild Goppa Codes

- Even more wild?

# Generalization of wild Goppa Codes

- Even more wild? Hidden wildness? Wild incognito :–)
- Potential problem: degree of $g$ (when optimized for key size) decreases with growing $q$; starting with $q = 11$ searching through all polynomials to find $g$ is faster than information-set decoding for security level $2^{128}$.
- Only “potential” problem since support would still be unknown and we can choose that so large that search is harder.
- Have published challenges to see what is easier to break. `pqcrypto.org/wild-challenges.html`
- Nevertheless: suggest to use

$$\Gamma_q(a_1, a_2, \ldots, a_n, g^{q-1} f) = \Gamma_q(a_1, a_2, \ldots, a_n, g^q f)$$

- Also offers better granularity in choosing the degrees.

# 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 2011 in Taipei, Taiwan, Nov 29 – Dec 2. Dates directly before Asiacrypt (Seoul, Korea) and Indocrypt (Chennai, India).