Code-Based Cryptography

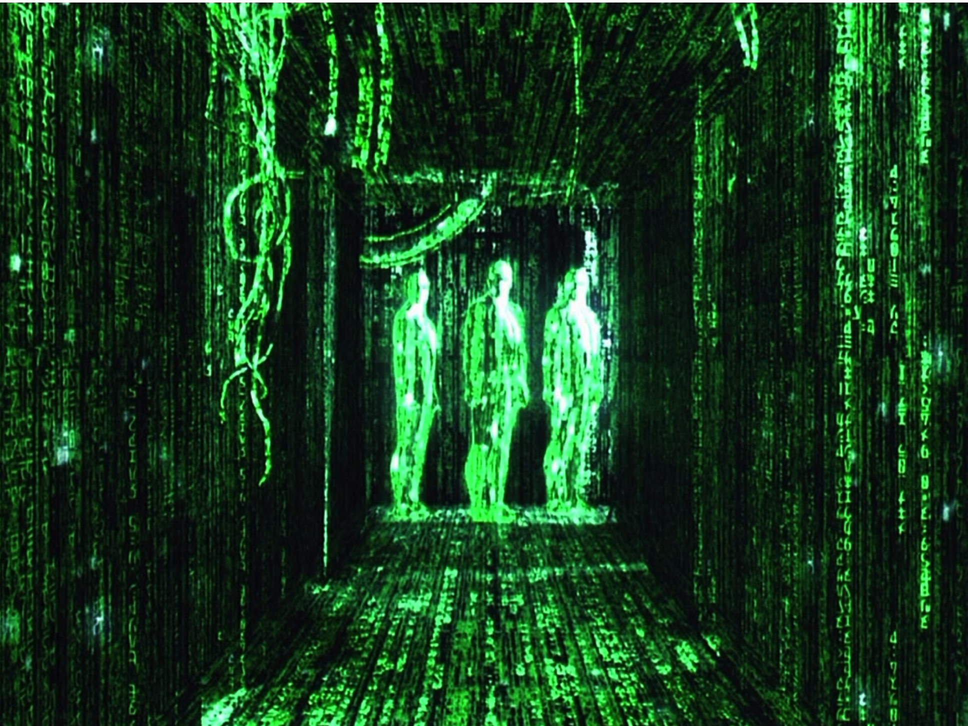
Tanja Lange
with some slides by Tung Chou and Christiane Peters

Technische Universiteit Eindhoven

Mastermath course
28 February 2017

Error correction

- Digital media is exposed to memory corruption.
- Many systems check whether data was corrupted in transit:
  - ISBN numbers have check digit to detect corruption.
  - ECC RAM detects up to two errors and can correct one error. 64 bits are stored as 72 bits: extra 8 bits for checks and recovery.
- In general, k bits of data get stored in n bits, adding some redundancy.
- If no error occurred, these n bits satisfy $n-k$ parity check equations; else can correct errors from the error pattern.
- Good codes can correct many errors without blowing up storage too much;
  offer guarantee to correct t errors (often can correct or at least detect more).
- To represent these check equations we need a matrix.



Hamming code

Parity check matrix ($n = 7, k = 4$):

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

An error-free string of 7 bits $\mathbf{b} = (b_0, b_1, b_2, b_3, b_4, b_5, b_6)$ satisfies these three equations:

$$\begin{array}{cccccccccc} b_0 & +b_1 & & +b_3 & +b_4 & & & = & 0 \\ b_0 & & +b_2 & +b_3 & & +b_5 & & = & 0 \\ & b_1 & +b_2 & +b_3 & & & +b_6 & = & 0 \end{array}$$

If one error occurred at least one of these equations will not hold.
Failure pattern uniquely identifies the error location,
e.g., $1, 0, 1$ means

Hamming code

Parity check matrix ($n = 7, k = 4$):

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

An error-free string of 7 bits $\mathbf{b} = (b_0, b_1, b_2, b_3, b_4, b_5, b_6)$ satisfies these three equations:

$$\begin{array}{cccccccccc} b_0 & +b_1 & & +b_3 & +b_4 & & & = & 0 \\ b_0 & & +b_2 & +b_3 & & +b_5 & & = & 0 \\ & b_1 & +b_2 & +b_3 & & & +b_6 & = & 0 \end{array}$$

If one error occurred at least one of these equations will not hold. Failure pattern uniquely identifies the error location,
e.g., $1, 0, 1$ means b_1 flipped.

Hamming code

Parity check matrix ($n = 7, k = 4$):

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

An error-free string of 7 bits $\mathbf{b} = (b_0, b_1, b_2, b_3, b_4, b_5, b_6)$ satisfies these three equations:

$$\begin{array}{cccccccccc} b_0 & +b_1 & & +b_3 & +b_4 & & & = & 0 \\ b_0 & & +b_2 & +b_3 & & +b_5 & & = & 0 \\ & b_1 & +b_2 & +b_3 & & & +b_6 & = & 0 \end{array}$$

If one error occurred at least one of these equations will not hold.
Failure pattern uniquely identifies the error location,
e.g., $1, 0, 1$ means b_1 flipped.
In math notation, the failure pattern is $H \cdot \mathbf{b}$.

Coding theory

- Names: code word $\mathbf{c}$, error vector $\mathbf{e}$, received word $\mathbf{b} = \mathbf{c} + \mathbf{e}$.
- Very common to transform the matrix so that the right part has just 1 on the diagonal (no need to store that).

$$H = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix} \rightsquigarrow \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix}$$

- Many special constructions discovered in 65 years of coding theory:
  - Large matrix H.
  - Fast decoding algorithm to find $\mathbf{e}$ given $\mathbf{s} = H \cdot (\mathbf{c} + \mathbf{e})$, whenever $\mathbf{e}$ does not have too many bits set.
- Given large H, usually very hard to find fast decoding algorithm.
- Use this difference in complexities for encryption.

Code-based encryption

- 1971 Goppa: Fast decoders for many matrices H.
- 1978 McEliece: Use Goppa codes for public-key cryptography.
  - Original parameters designed for 2^{64} security.
  - 2008 Bernstein–Lange–Peters: broken in $\approx 2^{60}$ cycles.
  - Easily scale up for higher security.
- 1986 Niederreiter: Simplified and smaller version of McEliece.
  - Public key: H with 1's on the diagonal.
  - Secret key: the fast Goppa decoder.
  - Encryption: Randomly generate $\mathbf{e}$ with t bits set.
    Send $H \cdot \mathbf{e}$.
  - Use hash of $\mathbf{e}$ to encrypt message with symmetric crypto (with 256 bits key).

Security analysis

- Some papers studying algorithms for attackers:
  1962 Prange; 1981 Omura; 1988 Lee–Brickell; 1988 Leon;
  1989 Krouk; 1989 Stern; 1989 Dumer; 1990 Coffey–Goodman;
  1990 van Tilburg; 1991 Dumer; 1991 Coffey–Goodman–Farrell;
  1993 Chabanne–Courteau; 1993 Chabaud; 1994 van Tilburg;
  1994 Canteaut–Chabanne; 1998 Canteaut–Chabaud;
  1998 Canteaut–Sendrier; 2008 Bernstein–Lange–Peters;
  2009 Bernstein–Lange–Peters–van Tilborg;
  2009 Bernstein (**post-quantum**); 2009 Finiasz–Sendrier;
  2010 Bernstein–Lange–Peters; 2011 May–Meurer–Thomae;
  2011 Becker–Coron–Joux; 2012 Becker–Joux–May–Meurer;
  2013 Bernstein–Jeffery–Lange–Meurer (**post-quantum**);
  2015 May–Ozerov.

Security analysis

- Some papers studying algorithms for attackers:
  1962 Prange; 1981 Omura; 1988 Lee–Brickell; 1988 Leon;
  1989 Krouk; 1989 Stern; 1989 Dumer; 1990 Coffey–Goodman;
  1990 van Tilburg; 1991 Dumer; 1991 Coffey–Goodman–Farrell;
  1993 Chabanne–Courteau; 1993 Chabaud; 1994 van Tilburg;
  1994 Canteaut–Chabanne; 1998 Canteaut–Chabaud;
  1998 Canteaut–Sendrier; 2008 Bernstein–Lange–Peters;
  2009 Bernstein–Lange–Peters–van Tilborg;
  2009 Bernstein (**post-quantum**); 2009 Finiasz–Sendrier;
  2010 Bernstein–Lange–Peters; 2011 May–Meurer–Thomae;
  2011 Becker–Coron–Joux; 2012 Becker–Joux–May–Meurer;
  2013 Bernstein–Jeffery–Lange–Meurer (**post-quantum**);
  2015 May–Ozerov.
- 256 KB public key for 2^{146} pre-quantum security.
- 512 KB public key for 2^{187} pre-quantum security.
- 1024 KB public key for 2^{263} pre-quantum security.

Security analysis

- Some papers studying algorithms for attackers:
  1962 Prange; 1981 Omura; 1988 Lee–Brickell; 1988 Leon;
  1989 Krouk; 1989 Stern; 1989 Dumer; 1990 Coffey–Goodman;
  1990 van Tilburg; 1991 Dumer; 1991 Coffey–Goodman–Farrell;
  1993 Chabanne–Courteau; 1993 Chabaud; 1994 van Tilburg;
  1994 Canteaut–Chabanne; 1998 Canteaut–Chabaud;
  1998 Canteaut–Sendrier; 2008 Bernstein–Lange–Peters;
  2009 Bernstein–Lange–Peters–van Tilborg;
  2009 Bernstein (**post-quantum**); 2009 Finiasz–Sendrier;
  2010 Bernstein–Lange–Peters; 2011 May–Meurer–Thomae;
  2011 Becker–Coron–Joux; 2012 Becker–Joux–May–Meurer;
  2013 Bernstein–Jeffery–Lange–Meurer (**post-quantum**);
  2015 May–Ozerov.
- 256 KB public key for 2^{146} pre-quantum security.
- 512 KB public key for 2^{187} pre-quantum security.
- 1024 KB public key for 2^{263} pre-quantum security.
- Post-quantum (Grover): below 2^{263}, above 2^{131}.

Next slide:
Initial recommendations of long-term secure post-quantum systems

Daniel Augot, Lejla Batina, Daniel J. Bernstein, Joppe Bos,
Johannes Buchmann, Wouter Castryck, Orr Dunkelman,
Tim Güneysu, Shay Gueron, Andreas Hülsing,
Tanja Lange, Mohamed Saied Emam Mohamed,
Christian Rechberger, Peter Schwabe, Nicolas Sendrier,
Frederik Vercauteren, Bo-Yin Yang

Work done as part of
https://pqcrypto.eu.org
PQCRYPTO EU project

Initial recommendations

- **Symmetric encryption** Thoroughly analyzed, 256-bit keys:
  - AES-256
  - Salsa20 with a 256-bit key

  Evaluating: Serpent-256, ...
- **Symmetric authentication** Information-theoretic MACs:
  - GCM using a 96-bit nonce and a 128-bit authenticator
  - Poly1305
- **Public-key encryption** McEliece with binary Goppa codes:
  - length $n = 6960$, dimension $k = 5413$, $t = 119$ errors

  Evaluating: QC-MDPC, Stehlé-Steinfeld NTRU, ...
- **Public-key signatures** Hash-based (minimal assumptions):
  - XMSS with any of the parameters specified in CFRG draft
  - SPHINCS-256

  Evaluating: HFEv-, ...

Linear Codes

A binary linear code C of length n and dimension k is a k-dimensional subspace of $\mathbb{F}_2^n$.
C is usually specified as

- the row space of a generating matrix $G \in \mathbb{F}_2^{k\times n}$

$$C = \{\mathbf{m}G | \mathbf{m} \in \mathbb{F}_2^k\}$$

- the kernel space of a parity-check matrix $H \in \mathbb{F}_2^{(n-k)\times n}$

$$C = \{\mathbf{c} | H\mathbf{c}^\mathsf{T} = 0,\ \ \mathbf{c} \in \mathbb{F}_2^n\}$$

Leaving out the $^\mathsf{T}$ from now on.

Example:

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

$\mathbf{c} = (111)G = (10011)$ is a codeword.

Systematic form, Hamming weight and distance

- 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 $H = (-Q^\intercal | I_{n-k})$.
- The Hamming weight of a word is the number of nonzero coordinates.

$$\mathrm{wt}(1,0,0,1,1) = 3$$

- The Hamming distance between two words in $\mathbb{F}_2^n$ is the number of coordinates in which they differ.

$$d((1,1,0,1,1),(1,0,0,1,1)) =$$

Systematic form, Hamming weight and distance

- 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 $H = (-Q^\intercal | I_{n-k})$.
- The Hamming weight of a word is the number of nonzero coordinates.
$$\mathrm{wt}(1,0,0,1,1) = 3$$
- The Hamming distance between two words in $\mathbb{F}_2^n$ is the number of coordinates in which they differ.
$$d((1,1,0,1,1),(1,0,0,1,1)) = 1$$

Systematic form, Hamming weight and distance

- 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 $H = (-Q^\intercal|I_{n-k})$.
- The Hamming weight of a word is the number of nonzero coordinates.
$$\mathrm{wt}(1,0,0,1,1) = 3$$
- The Hamming distance between two words in $\mathbb{F}_2^n$ is the number of coordinates in which they differ.
$$d((1,1,0,1,1),(1,0,0,1,1)) = 1$$
The Hamming distance between $\mathbf{x}$ and $\mathbf{y}$ equals the Hamming weight of $\mathbf{x}+\mathbf{y}$:
$$d((1,1,0,1,1),(1,0,0,1,1)) = \mathrm{wt}(0,1,0,0,0).$$

Decoding problem

- The minimum distance of a linear code C is the smallest Hamming weight of a nonzero codeword in C.

$$d = \min_{0 \neq \mathbf{c} \in C} \{\mathrm{wt}(\mathbf{c})\} = \min_{\mathbf{b} \neq \mathbf{c} \in C} \{d(\mathbf{b}, \mathbf{c})\}$$

- In code with minimum distance $d = 2t + 1$, any vector $\mathbf{x} = \mathbf{c} + \mathbf{e}$ with $\mathrm{wt}(\mathbf{e}) \leq t$ is uniquely decodable to $\mathbf{c}$; there is no closer code word.

Decoding problem

- The minimum distance of a linear code C is the smallest Hamming weight of a nonzero codeword in C.

$$d = \min_{0 \neq \mathbf{c} \in C} \{\mathrm{wt}(\mathbf{c})\} = \min_{\mathbf{b} \neq \mathbf{c} \in C} \{d(\mathbf{b}, \mathbf{c})\}$$

- In code with minimum distance $d = 2t + 1$, any vector $\mathbf{x} = \mathbf{c} + \mathbf{e}$ with $\mathrm{wt}(\mathbf{e}) \leq t$ is uniquely decodable to $\mathbf{c}$; there is no closer code word.

Decoding problem: find the closest codeword $\mathbf{c} \in C$ to a given $\mathbf{x} \in \mathbb{F}_2^n$, assuming that there is a unique closest codeword. Let $\mathbf{x} = \mathbf{c} + \mathbf{e}$. Note that finding $\mathbf{e}$ is an equivalent problem.

- If $\mathbf{c}$ is t errors away from $\mathbf{x}$, i.e., the Hamming weight of $\mathbf{e}$ is t, this is called a t-error correcting problem.
- There are lots of code families with fast decoding algorithms, e.g., Reed–Solomon codes, Goppa codes/alternant codes, etc.
- However, the general decoding problem is hard: Information-set decoding (see later) takes exponential time.

The Niederreiter cryptosystem I

Developed in 1986 by Harald Niederreiter as a variant of the McEliece cryptosystem. This is the schoolbook version.

- Use $n \times n$ permutation matrix P and $n-k \times n-k$ invertible matrix S.
- Public Key: a scrambled parity-check matrix $K = SHP \in \mathbb{F}_2^{(n-k)\times n}$.
- Encryption: The plaintext $\mathbf{e}$ is an n-bit vector of weight t. The ciphertext $\mathbf{s}$ is the $(n-k)$-bit vector
$$\mathbf{s} = K\mathbf{e}.$$
- Decryption: Find a n-bit vector $\mathbf{e}$ with $\mathrm{wt}(\mathbf{e}) = t$ such that $\mathbf{s} = K\mathbf{e}$.
- The passive attacker is facing a t-error correcting problem for the public key, which seems to be random.

The Niederreiter cryptosystem II

- Public Key: a scrambled parity-check matrix $K = SHP$.
- Encryption: The plaintext $\mathbf{e}$ is an n-bit vector of weight t. The ciphertext $\mathbf{s}$ is the $(n-k)$-bit vector
$$\mathbf{s} = K\mathbf{e}.$$
- Decryption using secret key: Compute
$$\begin{aligned} S^{-1}\mathbf{s} &= S^{-1}K\mathbf{e} = S^{-1}(SHP)\mathbf{e} \\ &= H(P\mathbf{e}) \end{aligned}$$
and observe that $\mathrm{wt}(P\mathbf{e}) = 1$, because P permutes. Use efficient decoder for H to find $\mathbf{e}' = P\mathbf{e}$ and thus $\mathbf{e} = P^{-1}\mathbf{e}'$.
- KEM/DEM version: pick random $\mathbf{e}$ of weight t, use hash($\mathbf{e}$) as secret key to encrypt and authenticate.

McBits (Bernstein, Chou, Schwabe, CHES 2013)

- Encryption is super fast anyways (just a vector-matrix multiplication).
- Main step in decryption is decoding of Goppa code. The McBits software achieves this in constant time.
- Decoding speed at 2^{128} pre-quantum security: $(n; t) = (4096; 41)$ uses 60493 Ivy Bridge cycles.
- Decoding speed at 2^{263} pre-quantum security: $(n; t) = (6960; 119)$ uses 306102 Ivy Bridge cycles.
- Grover speedup is less than halving the security level, so the latter parameters offer at least 2^{128} post-quantum security.
- More at `https://binary.cr.yp.to/mcbits.html`.

Binary Goppa code

Let $q = 2^m$. A binary Goppa code is often defined by

- a list $L = (a_1, \ldots, a_n)$ of n distinct elements in $\mathbb{F}_q$, called the support.
- a square-free polynomial $g(x) \in \mathbb{F}_q[x]$ of degree t such that $g(a) \neq 0$ for all $a \in L$. $g(x)$ is called the Goppa polynomial.
- E.g. choose $g(x)$ irreducible over $\mathbb{F}_q$.

The corresponding binary Goppa code $\Gamma(L, g)$ is

$$\left\{ \mathbf{c} \in \mathbb{F}_2^n \,\middle|\, S(\mathbf{c}) = \frac{c_1}{x - a_1} + \frac{c_2}{x - a_2} + \cdots + \frac{c_n}{x - a_n} \equiv 0 \bmod g(x) \right\}$$

- This code is linear $S(\mathbf{b} + \mathbf{c}) = S(\mathbf{b}) + S(\mathbf{c})$ and has length n.
- What can we say about the dimension and minimum distance?

Dimension of $\Gamma(L, g)$

- $g(a_i) \neq 0$ implies $\gcd(x - a_i, g(x)) = 1$, thus get polynomials
$$(x - a_i)^{-1} \equiv g_i(x) \equiv \sum_{j=0}^{t-1} g_{i,j} x^j \bmod g(x)$$
via XGCD. All this over $\mathbb{F}_q = \mathbb{F}_{2^m}$.
- In this form, $S(\mathbf{c}) \equiv 0 \bmod g(x)$ means
$$\sum_{i=1}^{n} c_i \left(\sum_{j=0}^{t-1} g_{i,j} x^j \right) = \sum_{j=0}^{t-1} \left(\sum_{i=1}^{n} c_i g_{i,j} \right) x^j = 0,$$
meaning that for each $0 \leq j \leq t - 1$:
$$\sum_{i=1}^{n} c_i g_{i,j} = 0.$$
- These are t conditions over $\mathbb{F}_q$, so tm conditions over $\mathbb{F}_2$. Giving an $tm \times n$ parity check matrix over $\mathbb{F}_2$.
- Some rows might be linearly dependent, so $k \geq n - tm$.

Nice parity check matrix

Assume $g(x) = \sum_{i=0}^{t} g_i x^i$ monic, i.e., $g_t = 1$.

$$H = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ g_{t-1} & 1 & 0 & \dots & 0 \\ g_{t-2} & g_{t-1} & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ g_1 & g_2 & g_3 & \dots & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ a_1 & a_2 & a_3 & \cdots & a_n \\ a_1^2 & a_2^2 & a_3^2 & \cdots & a_n^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_1^{t-1} & a_2^{t-1} & a_3^{t-1} & \cdots & a_n^{t-1} \end{pmatrix}$$
$$\cdot \begin{pmatrix} \frac{1}{g(a_1)} & 0 & 0 & \dots & 0 \\ 0 & \frac{1}{g(a_2)} & 0 & \dots & 0 \\ 0 & 0 & \frac{1}{g(a_3)} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \frac{1}{g(a_n)} \end{pmatrix}$$

Minimum distance of $\Gamma(L, g)$. Put $s(x) = S(\mathbf{c})$

$$s(x) \;=\; \sum_{i=1}^{n} c_i/(x - a_i)$$

Minimum distance of $\Gamma(L, g)$. Put $s(x) = S(\mathbf{c})$

$$\begin{aligned} s(x) &= \sum_{i=1}^{n} c_i/(x - a_i) \\ &= \left(\sum_{i=1}^{n} c_i \prod_{j \neq i} (x - a_j) \right) / \prod_{i=1}^{n} (x - a_i) \equiv 0 \bmod g(x). \end{aligned}$$

- $g(a_i) \neq 0$ implies $\gcd(x - a_i, g(x)) = 1$, so $g(x)$ divides $\sum_{i=1}^{n} c_i \prod_{j \neq i} (x - a_j)$.
- Let $\mathbf{c} \neq 0$ have small weight $\mathrm{wt}(\mathbf{c}) = w \leq t = \deg(g)$. For all i with $c_i = 0$, $x - a_i$ appears in every summand.

Minimum distance of $\Gamma(L, g)$. Put $s(x) = S(\mathbf{c})$

$$\begin{aligned} s(x) &= \sum_{i=1}^{n} c_i/(x - a_i) \\ &= \left(\sum_{i=1}^{n} c_i \prod_{j \neq i} (x - a_j) \right) / \prod_{i=1}^{n} (x - a_i) \equiv 0 \bmod g(x). \end{aligned}$$

- $g(a_i) \neq 0$ implies $\gcd(x - a_i, g(x)) = 1$,
  so $g(x)$ divides $\sum_{i=1}^{n} c_i \prod_{j \neq i} (x - a_j)$.
- Let $\mathbf{c} \neq 0$ have small weight $\mathrm{wt}(\mathbf{c}) = w \leq t = \deg(g)$.
  For all i with $c_i = 0$, $x - a_i$ appears in every summand.
  Cancel out those $x - a_i$ with $c_i = 0$.
- The denominator is now $\prod_{i, c_i \neq 0} (x - a_i)$, of degree w.
- The numerator now has degree $w - 1$ and $\deg(g) > w - 1$
  implies that the numerator is $= 0$ (without reduction mod g),
  which is a contradiction to $\mathbf{c} \neq 0$, so $\mathrm{wt}(\mathbf{c}) = w \geq t + 1$.

Better minimum distance for $\Gamma(L, g)$

- Let $\mathbf{c} \neq 0$ have small weight $\mathrm{wt}(\mathbf{c}) = w$.
- Put $f(x) = \prod_{i=1}^{n}(x - a_i)^{c_i}$ with $c_i \in \{0, 1\}$.
- Then the derivative $f'(x) = \sum_{i=1}^{n} c_i \prod_{j \neq i}(x - a_i)^{c_i}$.
- Thus $s(x) = f'(x)/f(x) \equiv 0 \bmod g(x)$.
- As before this implies $g(x)$ divides the numerator $f'(x)$.
- Note that over $\mathbb{F}_{2^m}$:

$$(f_{2i+1}x^{2i+1})' = f_{2i+1}x^{2i}, \quad (f_{2i}x^{2i})' = 0 \cdot f_{2i}x^{2i-1} = 0,$$

  thus $f'(x)$ contains only terms of even degree and $\deg(f') \leq w - 1$. Assume w odd, thus $\deg(f') = w - 1$.
- Note that over $\mathbb{F}_{2^m}$: $(x + 1)^2 = x^2 + 1$

Better minimum distance for $\Gamma(L, g)$

- Let $\mathbf{c} \neq 0$ have small weight $\mathrm{wt}(\mathbf{c}) = w$.
- Put $f(x) = \prod_{i=1}^{n} (x - a_i)^{c_i}$ with $c_i \in \{0, 1\}$.
- Then the derivative $f'(x) = \sum_{i=1}^{n} c_i \prod_{j \neq i} (x - a_i)^{c_i}$.
- Thus $s(x) = f'(x)/f(x) \equiv 0 \bmod g(x)$.
- As before this implies $g(x)$ divides the numerator $f'(x)$.
- Note that over $\mathbb{F}_{2^m}$:

  $$(f_{2i+1}x^{2i+1})' = f_{2i+1}x^{2i}, \quad (f_{2i}x^{2i})' = 0 \cdot f_{2i}x^{2i-1} = 0,$$

  thus $f'(x)$ contains only terms of even degree and $\deg(f') \leq w - 1$. Assume w odd, thus $\deg(f') = w - 1$.
- Note that over $\mathbb{F}_{2^m}$: $(x + 1)^2 = x^2 + 1$ and in general

  $$f'(x) = \sum_{i=0}^{(w-1)/2} F_{2i}x^{2i} = \left(\sum_{i=0}^{(w-1)/2} F_{2i}x^{i} \right)^2 = F^2(x).$$

- Since $g(x)$ is square-free, $g(x)$ divides $F(x)$, thus $w \geq 2t + 1$.

Decoding of in $\Gamma(L, g)$

- Decoding works with polynomial arithmetic.
- Fix $\mathbf{e}$. Let $\sigma(x) = \prod_{i, e_i \neq 0}(x - a_i)$. Same as $f(x)$ before.
- $\sigma(x)$ is called error locator polynomial. Given $\sigma(x)$ can factor it to retrieve error positions, $\sigma(a_i) = 0 \Leftrightarrow$ error in i.
- Split into odd and even terms: $\sigma(x) = a^2(x) + xb^2(x)$.
- Note as before $s(x) = \sigma'(x)/\sigma(x)$ and $\sigma'(x) = b^2(x)$.
- Thus

$$b^2(x) \equiv \sigma(x)s(x) \equiv (a^2(x) + xb^2(x))s(x) \bmod g(x)$$
$$b^2(x)(x + 1/s(x)) \equiv a^2(x) \bmod g(x)$$

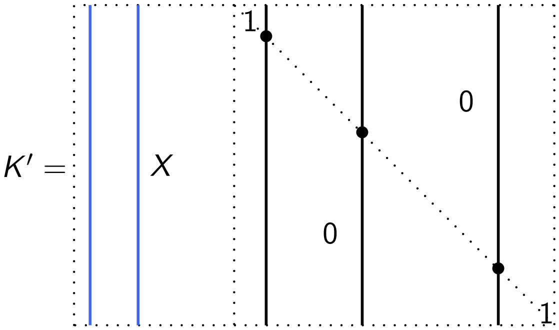
- Put $v(x) \equiv \sqrt{x + 1/s(x)} \bmod g(x)$, then $a(x) \equiv b(x)v(x) \bmod g(x)$.
- Can compute $v(x)$ from $s(x)$.
- Use XGCD on v and g, stop part-way when

$$a(x) = b(x)v(x) + h(x)g(x),$$

with $\deg(a) \leq \lfloor t/2 \rfloor, \deg(b) \leq \lfloor (t-1)/2 \rfloor$.

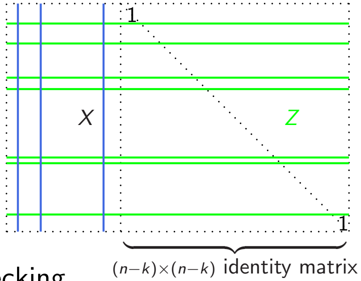
Generic attack: Information-set decoding

1988 Lee, Brickell. Reminder $\mathbf{s} = K\mathbf{e}$.



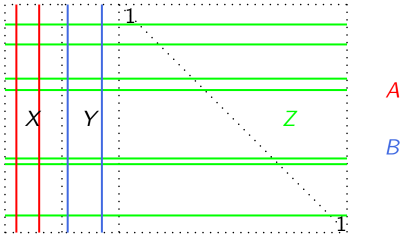
1. Permute K and bring to systematic form $K' = (X|I_{n-k})$. (If this fails, repeat with other permutation).
2. For small p, pick p of the k columns on the left, compute their sum $X\mathbf{p}$. ($\mathbf{p}$ is the vector of weight p).
3. If $\mathrm{wt}(\mathbf{s} + X\mathbf{p}) = t - p$ then put $\mathbf{e}' = \mathbf{p}||(\mathbf{s} + X\mathbf{p})$. Output unpermuted version of $\mathbf{e}'$.
4. Else return to 2 or return to 1 to rerandomize.

Leon's attack



- Setup similar to Lee-Brickell's attack.
- Random combinations of p vectors will be dense, so have $\mathrm{wt}(X\mathbf{p}) \sim k/2$.
- Idea: Introduce early abort by checking only ℓ positions (selected by set Z, green lines in the picture). This forms $\ell \times k$ matrix X_Z, length-ℓ vector $\mathbf{s}_Z$.
- Inner loop becomes:
  1. Pick $\mathbf{p}$ with $\mathrm{wt}(\mathbf{p}) = p$.
  2. Compute $X_Z\mathbf{p}$.
  3. If $\mathbf{s}_Z + X_Z\mathbf{p} \neq 0$ goto 1.
  4. Else compute $X\mathbf{p}$.
     - 4.1 If $\mathrm{wt}(\mathbf{s} + X\mathbf{p}) = t - p$ then put $\mathbf{e}' = \mathbf{p}||(\mathbf{s} + X\mathbf{p})$. Output unpermuted version of $\mathbf{e}'$.
     - 4.2 Else return to 1 or rerandomize K.
- Note that $\mathbf{s}_Z + X_Z\mathbf{p} = 0$ means that there are no ones in the positions specified by Z. Small loss in success, big speedup.

Stern's attack



- Setup similar to Leon's and Lee-Brickell's attacks.
- Use the early abort trick, so specify set Z.
- Improve chances of finding $\mathbf{p}$ with $X_Z\mathbf{p} = 0$:
    - Split left part of K' into two disjoint subsets X and Y.
    - Let $A = \{\mathbf{a} \in \mathbb{F}_2^{k/2} | \mathrm{wt}(\mathbf{a}) = p\}$, $B = \{\mathbf{b} \in \mathbb{F}_2^{k/2} | \mathrm{wt}(\mathbf{b}) = p\}$.
    - Search for words having exactly p ones in X and p ones in Y and exactly $w - 2p$ ones in the remaining columns.
    - Do the latter part as a collision search:
      Compute $\mathbf{s}_Z + X_Z\mathbf{a}$ for all (many) $\mathbf{a} \in A$, sort.
      Then compute $Y_Z\mathbf{b}$ for $\mathbf{b} \in B$ and look for collisions.
    - Iterate until word with $\mathrm{wt}(\mathbf{s} + X\mathbf{a} + Y\mathbf{b}) = 2p$ is found for some X, Y, Z.
- Select p, ℓ, and the subset of A to minimize overall work.
- Quantum targets: inner or outer loop.

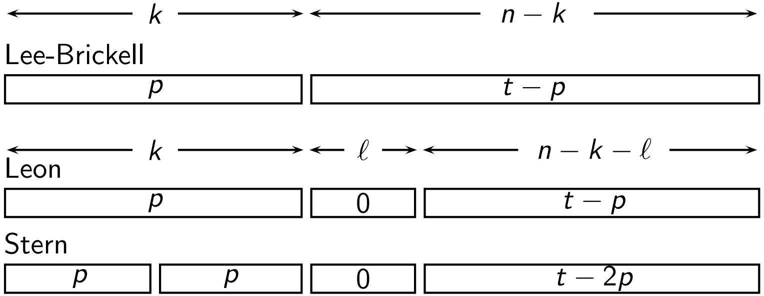
Running time in practice

2008 Bernstein, Lange, Peters.

- Wrote attack software against original McEliece parameters, decoding 50 errors in a $[1024, 524]$ code.
- Lots of optimizations, e.g. cheap updates between $\mathbf{s}_Z + X_Z\mathbf{a}$ and next value for $\mathbf{a}$; optimized frequency of K randomization.
- 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

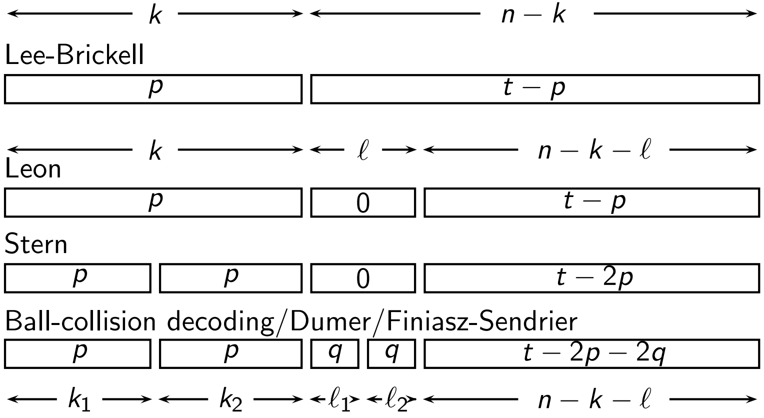
Methods differ in where the errors are allowed to be.



Running time is exponential for Goppa parameters n, k, d.

Information-set decoding

Methods differ in where the errors are allowed to be.



2011 May-Meurer-Thomae and 2012 Becker-Joux-May-Meurer refine multi-level collision search. Running time still exponential for Goppa parameters n, k, d; exponent is minimally smaller compared to Stern.

More exciting codes

- Niederreiter actually proposed to use generalized Reed-Solomon codes, this was broken in 1992 by Sidelnikov and Shestakov.
- In general we distinguish between generic attacks (such as information-set decoding) and structural attacks (that use the structure of the code).
- Gröbner basis computation is a generally powerful tool for structural attacks. (See talk by Ludovic Perret.)
- Cyclic codes need to store only top row of matrix, rest follows by shifts. Quasi-cyclic: multiple cyclic blocks.
- QC Goppa: too exciting, too much structure.
- Interesting candidate: Quasi-cyclic Moderate-Density Parity-Check (QC-MDPC) codes, due to Misoczki, Tillich, Sendrier, and Barreto (2012). Check out PQCrypto-2016 Maurich-Heberle-Güneysu (tomorrow morning) and Tung Chou's talk in the hot topics session.
- Hermitian codes, general algebraic geometry codes.
- Please help us update `https://pqcrypto.org/code.html`.