Code-Based Cryptography

Tung Chou
with some slides by Tanja Lange and Christiane Peters

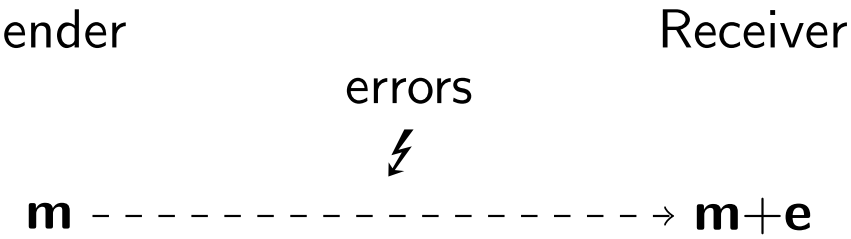
Academia Sinica

PQCRYPTO Mini-School 2020
20 July, 2020

Basics of coding theory

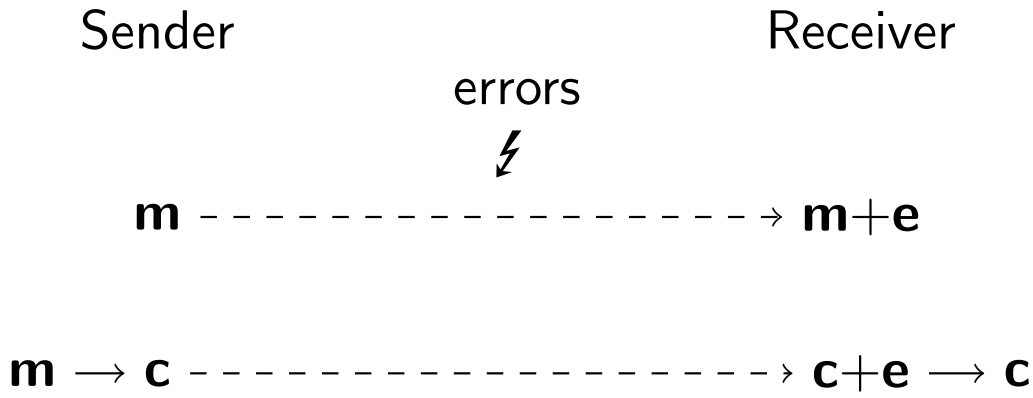
Error correction

- Goal: protect against errors in a noisy channel.



Error correction

- Goal: protect against errors in a noisy channel.



- The sender transforms a length-k message $\mathbf{m}$ into a length-n codeword $\mathbf{c}$ ($n > k$) by adding redundancy (encoding).
- The channel introduces errors (bitflips), which can be viewed as adding an error vector $\mathbf{e}$ to the data.
- The receiver uses a decoding algorithm to correct the errors. This works as long as there are not many errors.

Linear codes

A linear code C of length n and dimension k is a k-dimensional subspace of $\mathbb{F}_q^n$.

Linear codes

A linear code C of length n and dimension k is a k-dimensional subspace of $\mathbb{F}_q^n$.

C is usually specified as

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

$$C = \left\{ \mathbf{m}G \mid \mathbf{m} \in \mathbb{F}_q^k \right\}$$

Encoding means computing $\mathbf{c} = \mathbf{m}G$.

Linear codes

A linear code C of length n and dimension k is a k-dimensional subspace of $\mathbb{F}_q^n$.

C is usually specified as

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

$$C = \left\{ \mathbf{m}G \mid \mathbf{m} \in \mathbb{F}_q^k \right\}$$

Encoding means computing $\mathbf{c} = \mathbf{m}G$.

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

$$C = \left\{ \mathbf{c} \mid H\mathbf{c}^\intercal = 0, \ \ \mathbf{c} \in \mathbb{F}_q^n \right\}$$

(leaving out the $^\intercal$ and assuming $q = 2$ from now on)

Linear codes

A linear code C of length n and dimension k is a k-dimensional subspace of $\mathbb{F}_q^n$.

C is usually specified as

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

$$C = \left\{ \mathbf{m}G \mid \mathbf{m} \in \mathbb{F}_q^k \right\}$$

Encoding means computing $\mathbf{c} = \mathbf{m}G$.

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

$$C = \left\{ \mathbf{c} \mid H\mathbf{c}^\intercal = 0, \ \ \mathbf{c} \in \mathbb{F}_q^n \right\}$$

(leaving out the $^\intercal$ and assuming $q = 2$ from now on)

- In general generating and parity-check matrices are not unique!

Example

- C with code length $n = 7$ and code dimension $k = 4$.

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

- $\mathbf{c} = (1001)G = (1001001)$ is a codeword.

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

- $Hc = 0$.
- Note that $G = (I|Q)$ and $H = (Q^T|I)$.

Example

- C with code length $n = 7$ and code dimension $k = 4$.

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

- $\mathbf{c} = (1001)G = (1001001)$ is a codeword.

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

- $Hc = 0$.
- Note that $G = (I|Q)$ and $H = (Q^T|I)$.
- Linear codes are linear:

$$\alpha_1\mathbf{c}_1 + \alpha_2\mathbf{c}_2 = \alpha_1\mathbf{m}_1G + \alpha_2\mathbf{m}_2G = (\alpha_1\mathbf{m}_1 + \alpha_2\mathbf{m}_2)G.$$

$$H(\alpha_1\mathbf{c}_1 + \alpha_2\mathbf{c}_2) = \alpha_1H\mathbf{c}_1 + \alpha_2H\mathbf{c}_2 = 0 + 0 = 0.$$

Hamming weight and distance

- 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.

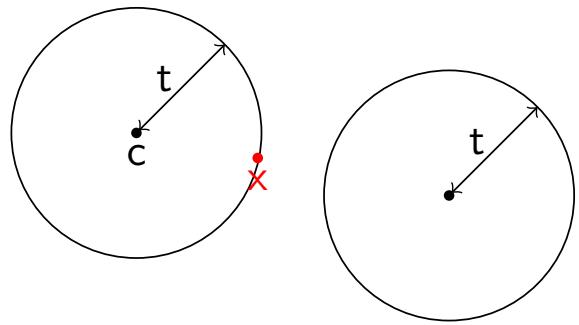
$$d((1,1,0,1,1),(1,0,0,1,1)) = 1$$

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

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

Minimum distance

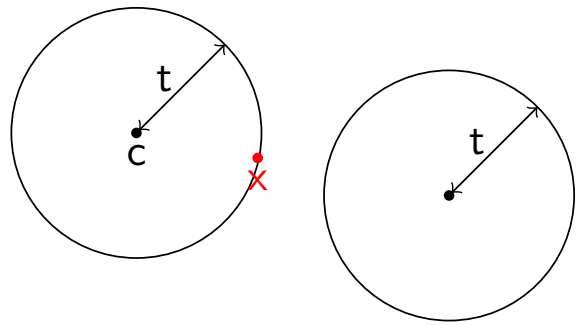
- Minimum distance indicates how many errors can be corrected: any vector $\mathbf{x} = \mathbf{c} + \mathbf{e}$ with $\mathrm{wt}(\mathbf{e}) = t < d/2$ is uniquely decodable to $\mathbf{c}$;



- Equivalently, the code can correct t errors if $d \geq 2t + 1$.
- Equivalently, the code can correct $\lfloor (d-1)/2 \rfloor$ errors.

Minimum distance

- Minimum distance indicates how many errors can be corrected: any vector $\mathbf{x} = \mathbf{c} + \mathbf{e}$ with $\mathrm{wt}(\mathbf{e}) = t < d/2$ is uniquely decodable to $\mathbf{c}$;



- Equivalently, the code can correct t errors if $d \geq 2t + 1$.
- Equivalently, the code can correct $\lfloor (d-1)/2 \rfloor$ errors.

- Having $t < d/2$ does not mean that there is an efficient algorithm for decoding t errors!

Hamming code

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

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

- Note that the columns are binary expansions of $\{1, 2, \ldots, 7\}$.

A codeword $\mathbf{c} = (c_0, c_1, c_2, c_3, c_4, c_5, c_6)$ satisfies these three equations:

$$\begin{array}{ccccccccccc} c_0 & + c_1 & & & + c_3 & + c_4 & & & & = & 0 \\ c_0 & & + c_2 & & + c_3 & & + c_5 & & & = & 0 \\ & c_1 & + c_2 & & + c_3 & & & + c_6 & & = & 0 \end{array}$$

- Minimum distance?

Hamming code

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

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

- Note that the columns are binary expansions of $\{1, 2, \ldots, 7\}$.

A codeword $\mathbf{c} = (c_0, c_1, c_2, c_3, c_4, c_5, c_6)$ satisfies these three equations:

$$\begin{array}{rrrrrrrrr} c_0 & +c_1 & & +c_3 & +c_4 & & & = & 0 \\ c_0 & & +c_2 & +c_3 & & +c_5 & & = & 0 \\ & c_1 & +c_2 & +c_3 & & & +c_6 & = & 0 \end{array}$$

- Minimum distance? $d = 3$.

Hamming code

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

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

- Note that the columns are binary expansions of $\{1, 2, \ldots, 7\}$.

A codeword $\mathbf{c} = (c_0, c_1, c_2, c_3, c_4, c_5, c_6)$ satisfies these three equations:

$$\begin{array}{rrrrrrrrr} c_0 & +c_1 & & +c_3 & +c_4 & & & = & 0 \\ c_0 & & +c_2 & +c_3 & & +c_5 & & = & 0 \\ & c_1 & +c_2 & +c_3 & & & +c_6 & = & 0 \end{array}$$

- Minimum distance? $d = 3$.
- The code can correct $\lfloor (d-1)/2 \rfloor = 1$ error.

Hamming code

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

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

- Note that the columns are binary expansions of $\{1, 2, \ldots, 7\}$.

A codeword $\mathbf{c} = (c_0, c_1, c_2, c_3, c_4, c_5, c_6)$ satisfies these three equations:

$$\begin{array}{rrrrrrrrr} c_0 & +c_1 & & +c_3 & +c_4 & & & = & 0 \\ c_0 & & +c_2 & +c_3 & & +c_5 & & = & 0 \\ & c_1 & +c_2 & +c_3 & & & +c_6 & = & 0 \end{array}$$

- Minimum distance? $d = 3$.
- The code can correct $\lfloor (d-1)/2 \rfloor = 1$ error.
- If there is an error in any c_i, the error position can be identified by $\mathbf{s} = H\mathbf{c}$, e.g. $\mathbf{s} = (1, 0, 1)^T$ means that $c_?$ is flipped.

Decoding problem

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, i.e., the decoding problem for random linear codes, is hard.
  - Theoretically, the problem is NP-complete
  - Pratically, Information-set decoding (see later) takes exponential time.

Different view: syndrome decoding

- The syndrome of $\mathbf{x} \in \mathbb{F}_2^n$ is $\mathbf{s} = H\mathbf{x}$.
  Note $H\mathbf{x} = H(\mathbf{c} + \mathbf{e}) = H\mathbf{c} + H\mathbf{e} = H\mathbf{e}$ depends only on $\mathbf{e}$.

- The syndrome decoding problem is to compute $\mathbf{e} \in \mathbb{F}_2^n$ given $\mathbf{s} \in \mathbb{F}_2^{n-k}$ so that $H\mathbf{e} = \mathbf{s}$ and $\mathbf{e}$ has minimal weight.

- Syndrome decoding and (regular) decoding are equivalent.

Different view: syndrome decoding

- The syndrome of $\mathbf{x} \in \mathbb{F}_2^n$ is $\mathbf{s} = H\mathbf{x}$.
  Note $H\mathbf{x} = H(\mathbf{c} + \mathbf{e}) = H\mathbf{c} + H\mathbf{e} = H\mathbf{e}$ depends only on $\mathbf{e}$.

- The syndrome decoding problem is to compute $\mathbf{e} \in \mathbb{F}_2^n$ given $\mathbf{s} \in \mathbb{F}_2^{n-k}$ so that $H\mathbf{e} = \mathbf{s}$ and $\mathbf{e}$ has minimal weight.

- Syndrome decoding and (regular) decoding are equivalent.
  - To decode $\mathbf{x} = \mathbf{c} + \mathbf{e}$ with syndrome decoder, compute $\mathbf{e}$ from $H\mathbf{x} = H\mathbf{e}$, then $\mathbf{c} = \mathbf{x} + \mathbf{e}$.
  - Given syndrome $\mathbf{s} = H\mathbf{e}$, assume $H = (Q^\mathsf{T} | I_{n-k})$.

Different view: syndrome decoding

- The syndrome of $\mathbf{x} \in \mathbb{F}_2^n$ is $\mathbf{s} = H\mathbf{x}$.
  Note $H\mathbf{x} = H(\mathbf{c} + \mathbf{e}) = H\mathbf{c} + H\mathbf{e} = H\mathbf{e}$ depends only on $\mathbf{e}$.

- The syndrome decoding problem is to compute $\mathbf{e} \in \mathbb{F}_2^n$ given $\mathbf{s} \in \mathbb{F}_2^{n-k}$ so that $H\mathbf{e} = \mathbf{s}$ and $\mathbf{e}$ has minimal weight.

- Syndrome decoding and (regular) decoding are equivalent.
  - To decode $\mathbf{x} = \mathbf{c} + \mathbf{e}$ with syndrome decoder, compute $\mathbf{e}$ from $H\mathbf{x} = H\mathbf{e}$, then $\mathbf{c} = \mathbf{x} + \mathbf{e}$.
  - Given syndrome $\mathbf{s} = H\mathbf{e}$, assume $H = (Q^\mathsf{T} | I_{n-k})$.
    Then $\mathbf{x} = (00\ldots0)||\mathbf{s}$ satisfies $\mathbf{s} = H\mathbf{x}$ and $\mathbf{x} = \mathbf{c} + \mathbf{e}$.

- Note that this $\mathbf{x}$ is not a solution to the syndrome decoding problem, unless it has very low weight.

Binary Goppa Codes

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 degree-t polynomial $g(x) \in \mathbb{F}_q[x]$ such that $g(a) \neq 0$ for all $a \in L$. $g(x)$ is called the Goppa polynomial.

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 degree-t polynomial $g(x) \in \mathbb{F}_q[x]$ such that $g(a) \neq 0$ for all $a \in L$. $g(x)$ is called the Goppa polynomial.

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.

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 degree-t polynomial $g(x) \in \mathbb{F}_q[x]$ such that $g(a) \neq 0$ for all $a \in L$. $g(x)$ is called the Goppa polynomial.

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.

- How about dimension k and minimum distance d?
- How should we represent the code as a matrix?
- Is there an efficient decoding algorithm?

Properties of $\Gamma(L, g)$

1. Dimension $k \geq n - mt$ (usually equality holds).
2. Nice parity-check matrix

$$\begin{pmatrix} 1/g(a_1) & 1/g(a_2) & \cdots & 1/g(a_n) \\ a_1/g(a_1) & a_2/g(a_2) & \cdots & a_n/g(a_n) \\ \vdots & \vdots & \ddots & \vdots \\ a_1^{t-1}/g(a_1) & a_2^{t-1}/g(a_2) & \cdots & a_n^{t-1}/g(a_n) \end{pmatrix}.$$

3. Minimum distance $d \geq t + 1$. $d \geq 2t + 1$ if g is square-free.
   - Square-free: if $g = g_1^{d_1} g_2^{d_2} \cdots g_\ell^{d_\ell}$, then $d_i = 1$ for all i.
   - Irreducible implies square-free .
4. $\Gamma(L, g) = \Gamma(L, g^2)$ if g is square-free
5. There exist efficient t-error decoding algorithms when g is square-free.

Dimension $k \geq n - mt$

- $g(a_i) \neq 0$ implies $\gcd(x - a_i, g(x)) = 1$, thus get polynomials

$$\frac{1}{x - a_i} \equiv f_i(x) = \sum_{j=0}^{t-1} f_{i,j} x^j \bmod g(x), \;\; f_i(x) \in \mathbb{F}_{2^m}[x]$$

- $\sum_{i=1}^{n} c_i f_i(x) \equiv 0 \bmod g(x) \implies \sum_{i=1}^{n} c_i f_i(x) = 0$, in other words,

$$\begin{pmatrix} f_{1,0} & f_{2,0} & \cdots & f_{n,0} \\ f_{1,1} & f_{2,1} & \cdots & f_{n,1} \\ \vdots & \vdots & \ddots & \vdots \\ f_{1,t-1} & f_{2,t-1} & \cdots & f_{n,t-1} \end{pmatrix} \cdot (c_1, \ldots, c_n)^T = (0, \ldots, 0)^T \in \mathbb{F}_{2^m}^t$$

- These are t conditions over $\mathbb{F}_{2^m}$, so tm conditions over $\mathbb{F}_2$.
- Some conditions might be linearly dependent, so $k \geq n - tm$.

Minimum distance $d \geq t + 1$.

- Put $s(x) = S(\mathbf{c}) = \sum_{i=1}^{n} c_i/(x - a_i)$

$$s(x) \quad = \quad \sum_{c_i=1} \frac{1}{x - a_i} = \left(\sum_{c_i=1} \prod_{j \neq i} (x - a_j) \right) / \prod_{c_i=1} (x - a_i)$$

Minimum distance $d \geq t + 1$.

- Put $s(x) = S(\mathbf{c}) = \sum_{i=1}^{n} c_i/(x - a_i)$

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

Note that $\deg(f) = \mathrm{wt}(\mathbf{c}) \implies \deg(f'(x)) \leq \mathrm{wt}(\mathbf{c}) - 1$.

Minimum distance $d \geq t+1$.

- Put $s(x) = S(\mathbf{c}) = \sum_{i=1}^{n} c_i/(x-a_i)$

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

  Note that $\deg(f) = \mathrm{wt}(\mathbf{c}) \implies \deg(f'(x)) \leq \mathrm{wt}(\mathbf{c}) - 1$.

- Multiply both sides of $\equiv$ by $f(x)$ to obtain

$$f'(x) = \sum_{c_i=1} \prod_{j \neq i} (x-a_j) \equiv 0 \bmod g(x)$$

Minimum distance $d \geq t+1$.

- Put $s(x) = S(\mathbf{c}) = \sum_{i=1}^{n} c_i/(x-a_i)$

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

  Note that $\deg(f) = \mathrm{wt}(\mathbf{c}) \implies \deg(f'(x)) \leq \mathrm{wt}(\mathbf{c}) - 1$.

- Multiply both sides of $\equiv$ by $f(x)$ to obtain

$$f'(x) = \sum_{c_i=1} \prod_{j \neq i} (x-a_j) \equiv 0 \bmod g(x)$$

- $f'(x)$ is a multiple of $g(x)$. Is $f'(x) = 0$?

Minimum distance $d \geq t+1$.

- Put $s(x) = S(\mathbf{c}) = \sum_{i=1}^{n} c_i/(x-a_i)$

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

  Note that $\deg(f) = \mathrm{wt}(\mathbf{c}) \implies \deg(f'(x)) \leq \mathrm{wt}(\mathbf{c}) - 1$.

- Multiply both sides of $\equiv$ by $f(x)$ to obtain

$$f'(x) = \sum_{c_i=1} \prod_{j \neq i} (x-a_j) \equiv 0 \bmod g(x)$$

- $f'(x)$ is a multiple of $g(x)$. Is $f'(x) = 0$?
- Note that $\gcd(f'(x), x-a_i) = 1$ if $c_i = 1$, so

$$\gcd(f'(x), f(x)) = 1 \implies f'(x) \neq 0$$

Minimum distance $d \geq t+1$.

- Put $s(x) = S(\mathbf{c}) = \sum_{i=1}^{n} c_i/(x-a_i)$

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

  Note that $\deg(f) = \mathrm{wt}(\mathbf{c}) \implies \deg(f'(x)) \leq \mathrm{wt}(\mathbf{c}) - 1$.

- Multiply both sides of $\equiv$ by $f(x)$ to obtain

$$f'(x) = \sum_{c_i=1} \prod_{j \neq i} (x-a_j) \equiv 0 \bmod g(x)$$

- $f'(x)$ is a multiple of $g(x)$. Is $f'(x) = 0$?
- Note that $\gcd(f'(x), x-a_i) = 1$ if $c_i = 1$, so

$$\gcd(f'(x), f(x)) = 1 \implies f'(x) \neq 0$$

- Therefore $f'(x)$ has degree $\geq t \implies \mathrm{wt}(\mathbf{c}) \geq t+1$

Minimum distance $d \geq 2t + 1$ for square-free $g(x)$

Minimum distance $d \geq 2t + 1$ for square-free $g(x)$

- Recall that $f'(x) \equiv 0 \bmod g(x)$.

Minimum distance $d \geq 2t + 1$ for square-free $g(x)$

- Recall that $f'(x) \equiv 0 \bmod g(x)$.
- Let $\mathrm{wt}(\mathbf{c}) = w$, so $\deg(f) = w$ and $\deg(f') \leq w - 1$

Minimum distance $d \geq 2t + 1$ for square-free $g(x)$

- Recall that $f'(x) \equiv 0 \bmod g(x)$.
- Let $\mathrm{wt}(\mathbf{c}) = w$, so $\deg(f) = w$ and $\deg(f') \leq w - 1$
- Observe 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.

Minimum distance $d \geq 2t + 1$ for square-free $g(x)$

- Recall that $f'(x) \equiv 0 \bmod g(x)$.
- Let $\mathrm{wt}(\mathbf{c}) = w$, so $\deg(f) = w$ and $\deg(f') \leq w - 1$
- Observe 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.
- Note that over $\mathbb{F}_{2^m}$: $x^{2i} + x^{2j} = (x^i + x^j)^2$

Minimum distance $d \geq 2t + 1$ for square-free $g(x)$

- Recall that $f'(x) \equiv 0 \bmod g(x)$.
- Let $\mathrm{wt}(\mathbf{c}) = w$, so $\deg(f) = w$ and $\deg(f') \leq w - 1$
- Observe 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.
- Note that over $\mathbb{F}_{2^m}$: $x^{2i} + x^{2j} = (x^i + x^j)^2$ and in general

$$f'(x) = \sum_{i=0}^{\lfloor (w-1)/2 \rfloor} f_{2i+1}x^{2i} = \left(\sum_{i=0}^{\lfloor (w-1)/2 \rfloor} \sqrt{f_{2i+1}}x^i \right)^2 = F^2(x).$$

Minimum distance $d \geq 2t + 1$ for square-free $g(x)$

- Recall that $f'(x) \equiv 0 \bmod g(x)$.
- Let $\mathrm{wt}(\mathbf{c}) = w$, so $\deg(f) = w$ and $\deg(f') \leq w - 1$
- Observe 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.
- Note that over $\mathbb{F}_{2^m}$: $x^{2i} + x^{2j} = (x^i + x^j)^2$ and in general

$$f'(x) = \sum_{i=0}^{\lfloor (w-1)/2 \rfloor} f_{2i+1}x^{2i} = \left(\sum_{i=0}^{\lfloor (w-1)/2 \rfloor} \sqrt{f_{2i+1}}x^i \right)^2 = F^2(x).$$

- Recall that $f'(x) \neq 0 \implies F(x) \neq 0$

Minimum distance $d \geq 2t+1$ for square-free $g(x)$

- Recall that $f'(x) \equiv 0 \bmod g(x)$.
- Let $\mathrm{wt}(\mathbf{c}) = w$, so $\deg(f) = w$ and $\deg(f') \leq w-1$
- Observe 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.
- Note that over $\mathbb{F}_{2^m}$: $x^{2i} + x^{2j} = (x^i + x^j)^2$ and in general

$$f'(x) = \sum_{i=0}^{\lfloor (w-1)/2 \rfloor} f_{2i+1}x^{2i} = \left(\sum_{i=0}^{\lfloor (w-1)/2 \rfloor} \sqrt{f_{2i+1}}x^i \right)^2 = F^2(x).$$

- Recall that $f'(x) \neq 0 \implies F(x) \neq 0$
- Since $g(x)$ is square-free, $g(x)|F^2(x) \implies g(x)|F(x)$, thus

$$\lfloor (w-1)/2 \rfloor \geq t \implies w \geq 2t+1$$

.

Decoding of $\mathbf{c} + \mathbf{e}$ when g is irreducible (Patterson)

- Fix $\mathbf{e}$. Let $\sigma(x) = \prod_{i, e_i \neq 0}(x - a_i)$. Similar to $f(x)$ before for $\mathbf{c}$.
- $\sigma(x)$ 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

$$A(x) \equiv B(x)v(x) \bmod g(x).$$

- Use XGCD on v and g, stop part-way when

$$A(x) = B(x)v(x) + h(x)g(x),$$

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

Code-based encryption

The McEliece cryptosystem I

- Let C be a length-n binary Goppa code Γ of dimension k with minimum distance $2t+1$: original parameters (1978) $n = 1024$, $k = 524$, $t = 50$.

- The McEliece secret key consists of
  - a generator matrix G for Γ
  - a uniform random permutation matrix $P \in \mathbb{F}_2^{n \times n}$
  - a uniform random nonsingular matrix $S \in \mathbb{F}_2^{k \times k}$.
  - an efficient t-error correcting decoding algorithm for Γ

- n, k, t are public; but Γ, P, S are randomly generated secrets.

- The McEliece public key is the $G' = SGP \in \mathbb{F}_2^{k \times n}$, which can be viewed as a scrambled version of G.

- The (public) key size is kn bits.

The McEliece cryptosystem I

- Let C be a length-n binary Goppa code Γ of dimension k with minimum distance $2t+1$: original parameters (1978) $n = 1024$, $k = 524$, $t = 50$.

- The McEliece secret key consists of
  - a generator matrix G for Γ
  - a uniform random permutation matrix $P \in \mathbb{F}_2^{n \times n}$
  - a uniform random nonsingular matrix $S \in \mathbb{F}_2^{k \times k}$.
  - an efficient t-error correcting decoding algorithm for Γ

- n, k, t are public; but Γ, P, S are randomly generated secrets.

- The McEliece public key is the $G' = SGP \in \mathbb{F}_2^{k \times n}$, which can be viewed as a scrambled version of G.

- The (public) key size is kn bits.

- The idea is to make G' look like a random matrix.

The McEliece cryptosystem II

Encryption

- The message is $\mathbf{m} \in \mathbb{F}_2^k$.
- Generate a uniform random error vector $\mathbf{e} \in \mathbb{F}_2^n$ with $\mathrm{wt}(\mathbf{e}) = t$.
- Send $\mathbf{y} = \mathbf{m}G' + \mathbf{e} \in \mathbb{F}_2^n$.

Decryption

- Compute $\mathbf{y}P^{-1} = \mathbf{m}G'P^{-1} + \mathbf{e}P^{-1} = (\mathbf{m}S)G + \mathbf{e}P^{-1}$.
- Note that P is a permutation matrix, so $\mathrm{wt}(\mathbf{e}P^{-1}) = \mathrm{wt}(\mathbf{e}) = t$.
- Use the decoding algorithm to find $(\mathbf{m}S)G$, $\mathbf{m}S$ and $\mathbf{m}$.

Security

- Finding $\mathbf{m}$ $\iff$ finding $\mathbf{m}G'$, so the 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.

The Niederreiter cryptosystem I

Developed in 1986 by Harald Niederreiter as a variant of the McEliece cryptosystem.

- The secret key consists of
  - a generator matrix H for the code C
  - a uniform random permutation matrix $P \in \mathbb{F}_2^{n \times n}$
  - a uniform random nonsingular matrix $S \in \mathbb{F}_2^{(n-k)\times(n-k)}$.
  - an efficient t-error correcting syndrome decoding algorithm for C

- The public key is $H' = SHP \in \mathbb{F}_2^{(n-k)\times n}$, a scrambled version of H.

- The (public) key size is $(n-k)n$ bits.

The Niederreiter cryptosystem I

Developed in 1986 by Harald Niederreiter as a variant of the McEliece cryptosystem.

- The secret key consists of
  - a generator matrix H for the code C
  - a uniform random permutation matrix $P \in \mathbb{F}_2^{n \times n}$
  - a uniform random nonsingular matrix $S \in \mathbb{F}_2^{(n-k)\times(n-k)}$.
  - an efficient t-error correcting syndrome decoding algorithm for C

- The public key is $H' = SHP \in \mathbb{F}_2^{(n-k)\times n}$, a scrambled version of H.

- The (public) key size is $(n-k)n$ bits.

- The idea is to make H' look like a random matrix.

The Niederreiter cryptosystem II

- Encryption: The plaintext $\mathbf{e}$ is an n-bit vector of weight t. The ciphertext $\mathbf{s}$ is the $(n-k)$-bit vector

$$\mathbf{s} = H'\mathbf{e}.$$

- Decryption using secret key: Compute

$$\begin{aligned} S^{-1}\mathbf{s} &= S^{-1}H'\mathbf{e} = S^{-1}(SHP)\mathbf{e} \\ &= H(P\mathbf{e}) \end{aligned}$$

  and observe that $\mathrm{wt}(P\mathbf{e}) = \mathrm{wt}(\mathbf{e}) = t$.

  Use efficient syndrome decoder for H to find $\mathbf{e}' = P\mathbf{e}$ and thus $\mathbf{e} = P^{-1}\mathbf{e}'$.

- Breaking Niederreiter $\Longleftrightarrow$ Breaking McEliece

Systematic form

- Idea: choose S such that $G' = (I_k|Q)$ for McEliece and $H' = (Q^\intercal|I_{n-k})$ for Niederreiter, where $Q \in \mathbb{F}_2^{k\times(n-k)}$. Now the key size is $(n-k)k$.

Systematic form

- Idea: choose S such that $G' = (I_k|Q)$ for McEliece and $H' = (Q^\intercal|I_{n-k})$ for Niederreiter, where $Q \in \mathbb{F}_2^{k\times(n-k)}$. Now the key size is $(n-k)k$.

- Summary on key and ciphertext sizes

	|key|	|ciphertext|
McEliece	$k \times n$	n
Niederreiter	$(n-k) \times n$	$n-k$
Systematic McEliece	$k \times (n-k)$	n
Systematic Niederreiter	$(n-k) \times k$	$n-k$

Systematic form

- Idea: choose S such that $G' = (I_k|Q)$ for McEliece and $H' = (Q^\intercal|I_{n-k})$ for Niederreiter, where $Q \in \mathbb{F}_2^{k\times(n-k)}$. Now the key size is $(n-k)k$.

- Summary on key and ciphertext sizes

	|key|	|ciphertext|
McEliece	$k \times n$	n
Niederreiter	$(n-k) \times n$	$n-k$
Systematic McEliece	$k \times (n-k)$	n
Systematic Niederreiter	$(n-k) \times k$	$n-k$

- As long as there is a sufficiently high probability to have systematic form, this optimization does not sacrifice security: an attack on the resulting cryptosystem implies an attack on the original system.

Systematic form

- Idea: choose S such that $G' = (I_k|Q)$ for McEliece and $H' = (Q^\intercal|I_{n-k})$ for Niederreiter, where $Q \in \mathbb{F}_2^{k\times(n-k)}$. Now the key size is $(n-k)k$.

- Summary on key and ciphertext sizes

	|key|	|ciphertext|
McEliece	$k \times n$	n
Niederreiter	$(n-k) \times n$	$n-k$
Systematic McEliece	$k \times (n-k)$	n
Systematic Niederreiter	$(n-k) \times k$	$n-k$

- As long as there is a sufficiently high probability to have systematic form, this optimization does not sacrifice security: an attack on the resulting cryptosystem implies an attack on the original system.

- Does not work for all secret keys: might need to retry a few times.

Security analysis

Some papers studying algorithms for attackers:
1962 Prange; 1981 Clark–Cain, crediting 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; 2012 Becker–Joux–May–Meurer; 2013 Hamdaoui–Sendrier; 2015 May–Ozerov; 2016 Canto Torres–Sendrier; 2017 Kachigar–Tillich (**post-quantum**); 2017 Both–May; 2018 Both–May; 2018 Kirshanova (**post-quantum**).

Consequence of security analysis

- The McEliece system (with key-size optimizations) uses $(c_0 + o(1))\lambda^2(\lg \lambda)^2$-bit keys as $\lambda \to \infty$ to achieve 2^λ security against all these attacks.

Consequence of security analysis

- The McEliece system (with key-size optimizations) uses $(c_0 + o(1))\lambda^2(\lg \lambda)^2$-bit keys as $\lambda \to \infty$ to achieve 2^λ security against all these attacks. Here $c_0 \approx 0.7418860694$.

Consequence of security analysis

- The McEliece system (with key-size optimizations) uses $(c_0 + o(1))\lambda^2(\lg \lambda)^2$-bit keys as $\lambda \to \infty$ to achieve 2^λ security against all these attacks. Here $c_0 \approx 0.7418860694$.

- 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.

Consequence of security analysis

- The McEliece system (with key-size optimizations) uses $(c_0 + o(1))\lambda^2(\lg \lambda)^2$-bit keys as $\lambda \to \infty$ to achieve 2^λ security against all these attacks. Here $c_0 \approx 0.7418860694$.

- 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): lower-bounded by square root of the pre-quantum security level.

Note on codes

- McEliece proposed to use binary Goppa codes.
  These are still used today.
- Niederreiter described his scheme using Reed-Solomon codes.
  These were broken in 1992 by Sidelnikov and Chestakov.

Note on codes

- McEliece proposed to use binary Goppa codes.
  These are still used today.
- Niederreiter described his scheme using Reed-Solomon codes.
  These were broken in 1992 by Sidelnikov and Chestakov.
- More corpses on the way: concatenated codes, Reed-Muller codes, several Algebraic Geometry (AG) codes, Gabidulin codes, several LDPC codes, cyclic codes.

Note on codes

- McEliece proposed to use binary Goppa codes.
  These are still used today.
- Niederreiter described his scheme using Reed-Solomon codes.
  These were broken in 1992 by Sidelnikov and Chestakov.
- More corpses on the way: concatenated codes, Reed-Muller codes, several Algebraic Geometry (AG) codes, Gabidulin codes, several LDPC codes, cyclic codes.
- Many recent schemes use QC-MDPC codes and rank-metric codes.
  - Small key size due to structures in the public keys.

Note on codes

- McEliece proposed to use binary Goppa codes.
  These are still used today.
- Niederreiter described his scheme using Reed-Solomon codes.
  These were broken in 1992 by Sidelnikov and Chestakov.
- More corpses on the way: concatenated codes, Reed-Muller codes, several Algebraic Geometry (AG) codes, Gabidulin codes, several LDPC codes, cyclic codes.
- Many recent schemes use QC-MDPC codes and rank-metric codes.
  - Small key size due to structures in the public keys.
  - Is the underlying hard problem well-studied?
  - Decoding failures.

Do not use the schoolbook versions!

Sloppy Alice attacks! 1998 Verheul, Doumen, van Tilborg

- Assume that the decoding algorithm decodes up to t errors, i. e. it decodes $\mathbf{y} = \mathbf{c} + \mathbf{e}$ to $\mathbf{c}$ if $\mathrm{wt}(\mathbf{e}) \leq t$.

Sloppy Alice attacks! 1998 Verheul, Doumen, van Tilborg

- Assume that the decoding algorithm decodes up to t errors,
  i. e. it decodes $\mathbf{y} = \mathbf{c} + \mathbf{e}$ to $\mathbf{c}$ if $\mathrm{wt}(\mathbf{e}) \leq t$.

- Eve intercepts ciphertext $\mathbf{y} = \mathbf{m}G' + \mathbf{e}, \mathrm{wt}(\mathbf{e}) = t$.
  Eve poses as Alice towards Bob and sends him tweaks of $\mathbf{y}$.
  She uses Bob's reactions (success or failure to decrypt) to recover $\mathbf{m}$.

Sloppy Alice attacks! 1998 Verheul, Doumen, van Tilborg

- Assume that the decoding algorithm decodes up to t errors,
  i. e. it decodes $\mathbf{y} = \mathbf{c} + \mathbf{e}$ to $\mathbf{c}$ if $\mathrm{wt}(\mathbf{e}) \leq t$.

- Eve intercepts ciphertext $\mathbf{y} = \mathbf{m}G' + \mathbf{e}, \mathrm{wt}(\mathbf{e}) = t$.
  Eve poses as Alice towards Bob and sends him tweaks of $\mathbf{y}$.
  She uses Bob's reactions (success or failure to decrypt) to recover $\mathbf{m}$.

- Eve sends $\mathbf{y}_i = \mathbf{y} + \mathbf{e}_i$ for $\mathbf{e}_i$ the i-th unit vector.
  If Bob returns error, position i in $\mathbf{e}$ is 0 (so the number of errors has increased to $t + 1$ and Bob fails).
  Else position i in $\mathbf{e}$ is 1.

Sloppy Alice attacks! 1998 Verheul, Doumen, van Tilborg

- Assume that the decoding algorithm decodes up to t errors,
  i. e. it decodes $\mathbf{y} = \mathbf{c} + \mathbf{e}$ to $\mathbf{c}$ if $\mathrm{wt}(\mathbf{e}) \leq t$.

- Eve intercepts ciphertext $\mathbf{y} = \mathbf{m}G' + \mathbf{e}, \mathrm{wt}(\mathbf{e}) = t$.
  Eve poses as Alice towards Bob and sends him tweaks of $\mathbf{y}$.
  She uses Bob's reactions (success or failure to decrypt) to recover $\mathbf{m}$.

- Eve sends $\mathbf{y}_i = \mathbf{y} + \mathbf{e}_i$ for $\mathbf{e}_i$ the i-th unit vector.
  If Bob returns error, position i in $\mathbf{e}$ is 0 (so the number of errors has increased to $t + 1$ and Bob fails).
  Else position i in $\mathbf{e}$ is 1.

- After k steps Eve knows the first k positions of $\mathbf{m}G'$ without error.
  Invert the $k \times k$ submatrix of G' to get $\mathbf{m}$ assuming it is invertible.

More on sloppy Alice

- This attack has Eve send Bob variations of the same ciphertext; so Bob will think that Alice is sloppy.
- Other name: reaction attack.
  (1999 Hall, Goldberg, and Schneier)

More on sloppy Alice

- This attack has Eve send Bob variations of the same ciphertext; so Bob will think that Alice is sloppy.
- Other name: reaction attack.
  (1999 Hall, Goldberg, and Schneier)
- Attack also works on Niederreiter version:
  Bitflip cooresponds to sending $\mathbf{s}_i = \mathbf{s} + K_i$,
  where K_i is the i-th column of K.
- More involved but doable (for McEliece and Niederreiter)
  if decryption requires exactly t errors.

Berson's attack

- Eve knows $\mathbf{y}_1 = \mathbf{m}G' + \mathbf{e}_1$ and $\mathbf{y}_2 = \mathbf{m}G' + \mathbf{e}_2$;
  these have the same $\mathbf{m}$.
- Then $\mathbf{y}_1 + \mathbf{y}_2 = \mathbf{e}_1 + \mathbf{e}_2 = \bar{\mathbf{e}}$. This has weight in $[0, 2t]$.
- If $\mathrm{wt}(\bar{\mathbf{e}}) = 2t$:
  All zero positions in $\bar{\mathbf{e}}$ are error free in both ciphertexts.
  Invert G' in those columns to recover $\mathbf{m}$ as in previous attack.
- Else:

Berson's attack

- Eve knows $\mathbf{y}_1 = \mathbf{m}G' + \mathbf{e}_1$ and $\mathbf{y}_2 = \mathbf{m}G' + \mathbf{e}_2$;
  these have the same $\mathbf{m}$.
- Then $\mathbf{y}_1 + \mathbf{y}_2 = \mathbf{e}_1 + \mathbf{e}_2 = \bar{\mathbf{e}}$. This has weight in $[0, 2t]$.
- If $\mathrm{wt}(\bar{\mathbf{e}}) = 2t$:
  All zero positions in $\bar{\mathbf{e}}$ are error free in both ciphertexts.
  Invert G' in those columns to recover $\mathbf{m}$ as in previous attack.
- Else: ignore the $2w = \mathrm{wt}(\bar{\mathbf{e}}) < 2t$ positions in G' and $\mathbf{y}_1$.
  Solve decoding problem for $k \times (n - 2w)$ generator matrix G'' and vector $\mathbf{y}'_1$ with $t - w$ errors; typically much easier.

Security notions and generic attacks

Formal security notions

- McEliece/Niederreiter are One-Way Encryption (OWE) schemes: the attaker is asked to recover a randomly chosen plaintext given the ciphertext.
  - For McEliece, find randomly chosen $\mathbf{m}$ given $\mathbf{m}G' + \mathbf{e}$.
  - For Niederreiter, find randomly chosen $\mathbf{e}$ given $H'\mathbf{e}$.

Formal security notions

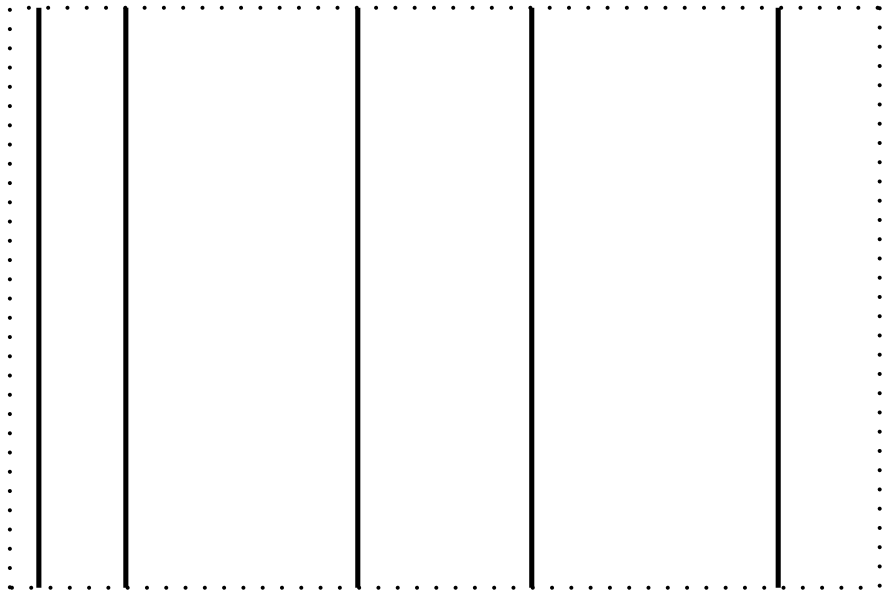
- McEliece/Niederreiter are One-Way Encryption (OWE) schemes: the attaker is asked to recover a randomly chosen plaintext given the ciphertext.
  - For McEliece, find randomly chosen $\mathbf{m}$ given $\mathbf{m}G' + \mathbf{e}$.
  - For Niederreiter, find randomly chosen $\mathbf{e}$ given $H'\mathbf{e}$.
- In practice, we often require a scheme to be CCA–II secure: given challenge ciphertext $\mathbf{y}$, Eve can ask for decryption of anything but $\mathbf{y}$.
  - Given $\mathbf{y} = \mathbf{m}G' + \mathbf{e}$, Eve picks a random code word $\mathbf{c} = \bar{\mathbf{m}}G'$, asks for decryption of $\mathbf{y} + \mathbf{c}$.
  - This is different from challenge $\mathbf{y}$, so Bob answers.
  - Answer is $\mathbf{m} + \bar{\mathbf{m}}$.

Formal security notions

- McEliece/Niederreiter are One-Way Encryption (OWE) schemes: the attaker is asked to recover a randomly chosen plaintext given the ciphertext.
  - For McEliece, find randomly chosen $\mathbf{m}$ given $\mathbf{m}G' + \mathbf{e}$.
  - For Niederreiter, find randomly chosen $\mathbf{e}$ given $H'\mathbf{e}$.
- In practice, we often require a scheme to be CCA–II secure: given challenge ciphertext $\mathbf{y}$, Eve can ask for decryption of anything but $\mathbf{y}$.
  - Given $\mathbf{y} = \mathbf{m}G' + \mathbf{e}$, Eve picks a random code word $\mathbf{c} = \bar{\mathbf{m}}G'$, asks for decryption of $\mathbf{y} + \mathbf{c}$.
  - This is different from challenge $\mathbf{y}$, so Bob answers.
  - Answer is $\mathbf{m} + \bar{\mathbf{m}}$.
- Fix by using CCA2 transformation (e.g. Fujisaki-Okamoto transform) or (easier) KEM/DEM version. These transforms use symmetric primitives such as hash functions.

Generic attack: Brute force

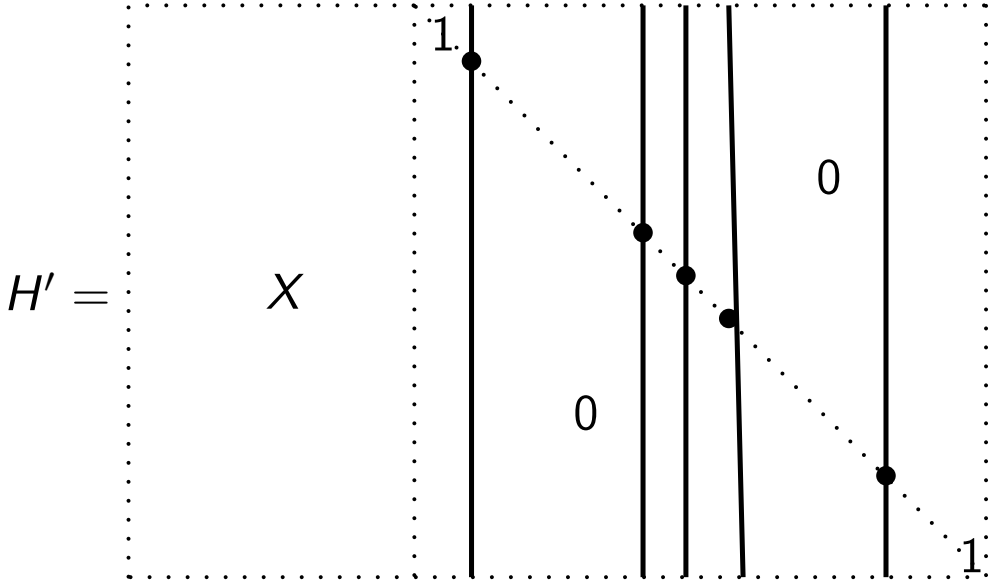
Given $H \in \mathbb{F}_2^{(n-k)\times n}$ and $\mathbf{s} = H\mathbf{e}$, find $\mathbf{e}$ with $\mathrm{wt}(\mathbf{e}) = t$.

$H =$ 

- Pick any group of t columns of H, add them and compare with $\mathbf{s}$.
- Cost: $\binom{n}{t}$ sums of t columns.

  Can do better so that each try costs only 1 column addition (after some initial additions).

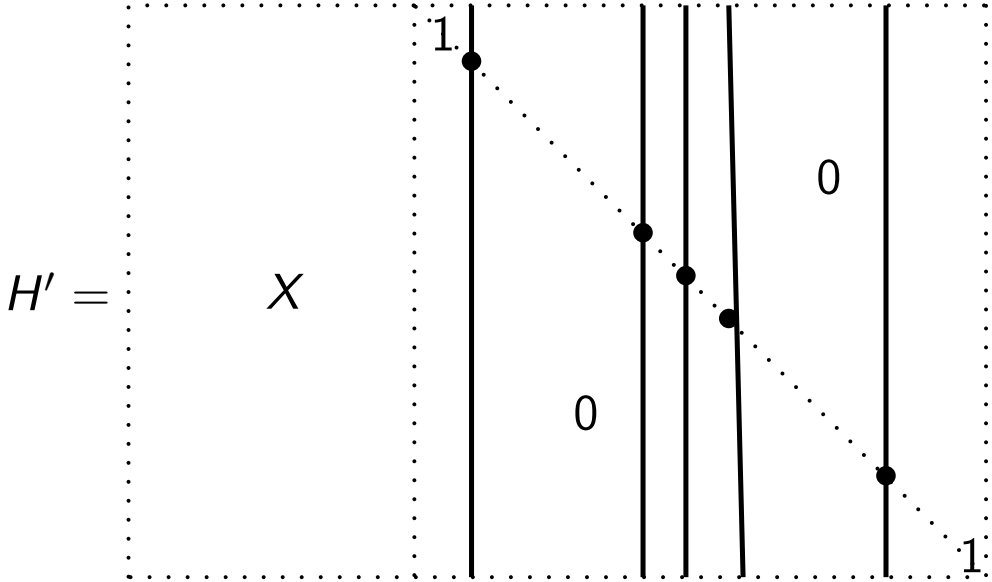
Generic attack: Information-set decoding, 1962 Prange



1. Permute H and bring to systematic form $H' = (X|I_{n-k})$. (If this fails, repeat with other permutation).
2. Then $H' = UHP$ for some permutation matrix P and U the matrix that produces systematic form.
3. This updates $\mathbf{s}$ to $U\mathbf{s}$.
4. If $\mathrm{wt}(U\mathbf{s}) = t$ then $UHP\mathbf{e}' = US$ where $\mathbf{e}' = (00\ldots0)||U\mathbf{s}$. Output $P\mathbf{e}'$.
5. Else return to 1 to rerandomize.

Cost:

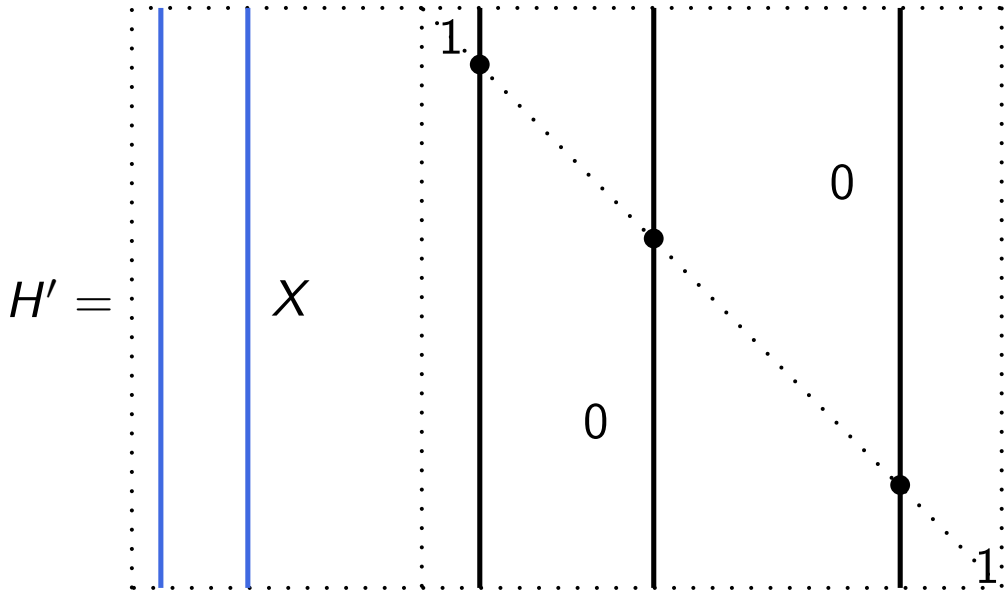
Generic attack: Information-set decoding, 1962 Prange



1. Permute H and bring to systematic form $H' = (X|I_{n-k})$. (If this fails, repeat with other permutation).
2. Then $H' = UHP$ for some permutation matrix P and U the matrix that produces systematic form.
3. This updates $\mathbf{s}$ to $U\mathbf{s}$.
4. If $\mathrm{wt}(U\mathbf{s}) = t$ then $UHP\mathbf{e}' = US$ where $\mathbf{e}' = (00\ldots0)||U\mathbf{s}$. Output $P\mathbf{e}'$.
5. Else return to 1 to rerandomize.

Cost: $O(\binom{n}{t}/\binom{n-k}{t})$ matrix operations.

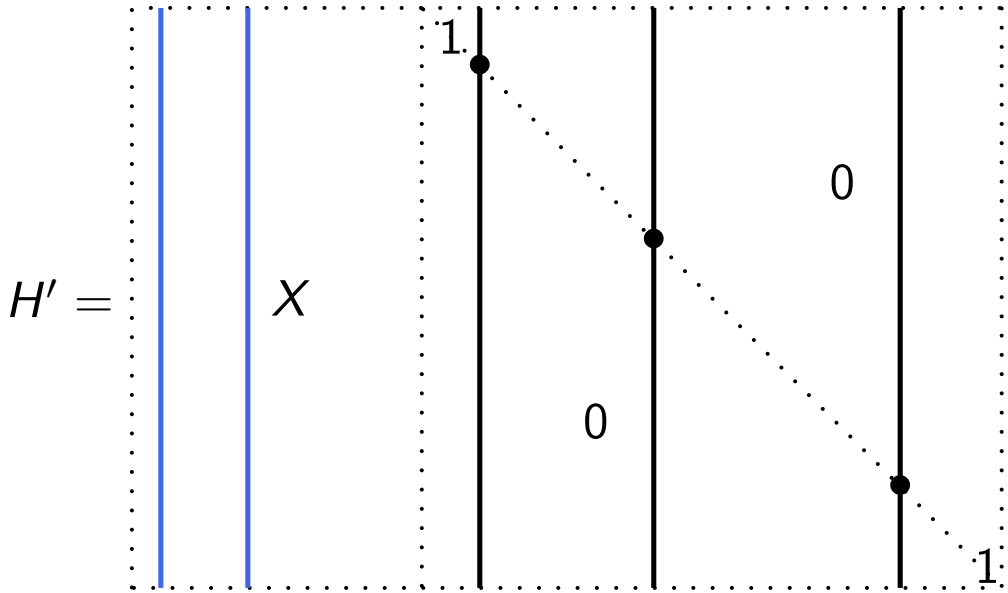
Lee–Brickell attack



1. Permute H and bring to systematic form $H' = (X|I_{n-k})$.
   (If this fails, repeat with other permutation). $\mathbf{s}$ is updated.
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.

Cost:

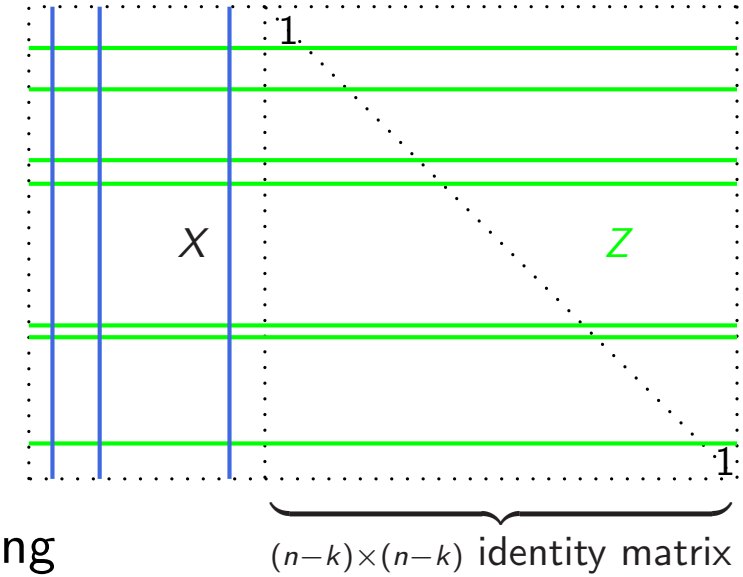
Lee–Brickell attack



1. Permute H and bring to systematic form $H' = (X|I_{n-k})$.
   (If this fails, repeat with other permutation). $\mathbf{s}$ is updated.
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.

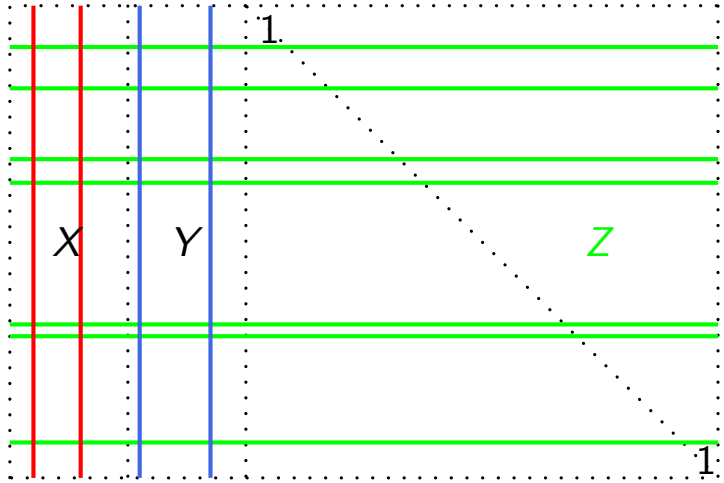
Cost: $O(\binom{n}{t}/(\binom{k}{p}\binom{n-k}{t-p}))$ [matrix operations$+\binom{k}{p}$ column additions].

Leon's attack



- Setup similar to Lee-Brickell's attack.
- Random combinations of p vectors will be dense, so have $\mathrm{wt}(\mathbf{s} + X\mathbf{p}) \sim (n-k)/2$.
- 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 $X\mathbf{p}$.
     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 $\mathbf{e}'$.
     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 $\mathbf{s} + X_Z\mathbf{p} = 0$:
  - 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; expand.
  - Iterate until word with $\mathrm{wt}(\mathbf{s} + X\mathbf{a} + Y\mathbf{b}) = t - 2p$ is found for some X, Y, Z.
- Select p, ℓ, and the subset of A to minimize overall work.

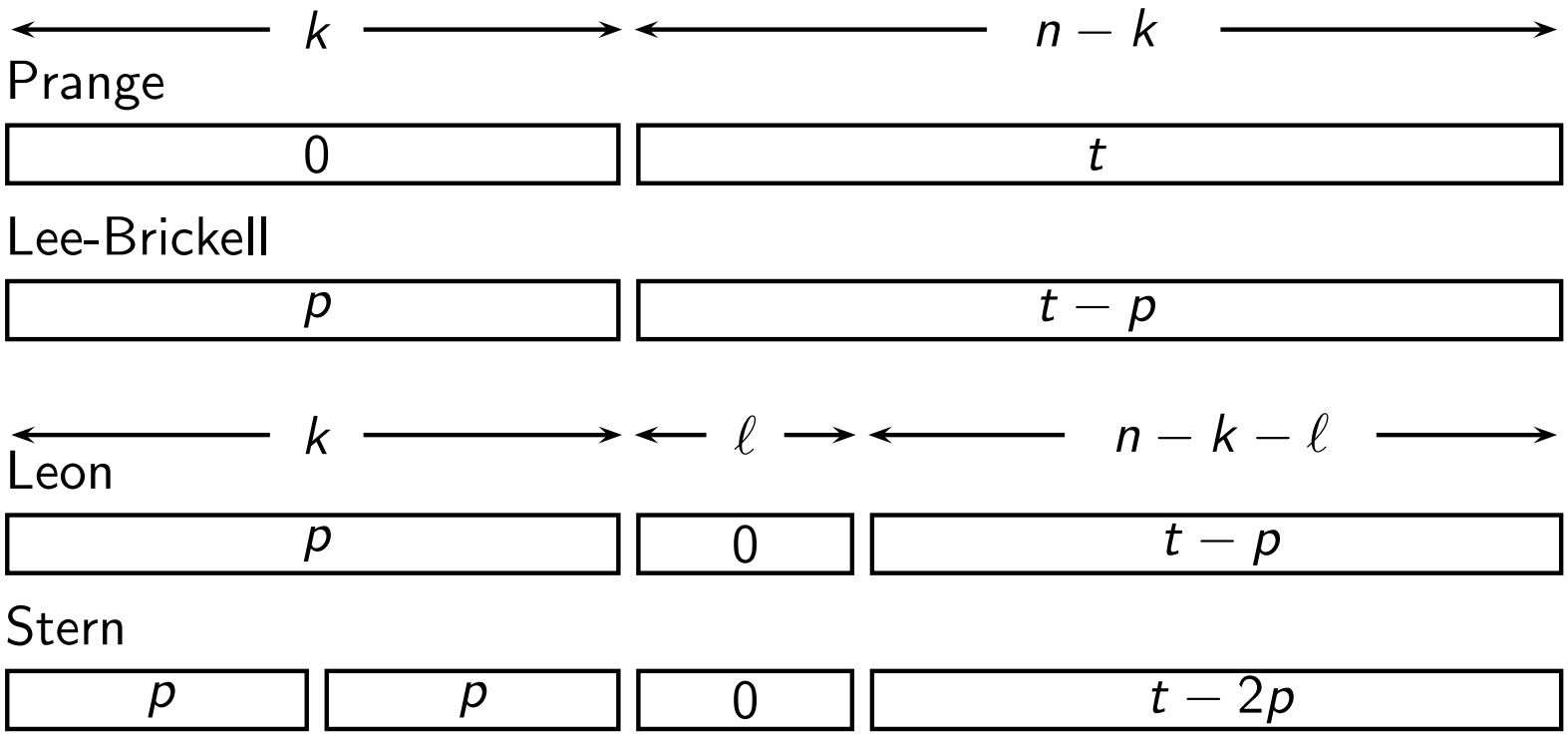
Running time in practice

2008 Bernstein, Lange, Peters.

- Wrote attack software against original McEliece parameters, decoding 50 errors in a $[1024, 524]$ code.
- Lots of optimizations for Stern, 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.

Information-set decoding

Methods differ in where the errors are allowed to be.



Classic McEliece
conservative code-based cryptography
https://classic.mceliece.org/

Daniel J. Bernstein, Tung Chou, Tanja Lange,
Ingo von Maurich, Rafael Misoczki, Ruben Niederhagen,
Edoardo Persichetti, Christiane Peters, Peter Schwabe,
Nicolas Sendrier, Jakub Szefer, Wen Wang

NIST PQC standardization process

- 1st round: Dec. 2017 – Mar. 2019; 2nd round: Apr. 2019 – now.

Public-key Encryption and Key-establishment Algorithms	Type
BIKE	Code-based
Classic McEliece	Code-based
CRYSTALS-KYBER	Lattice-based
FrodoKEM	Lattice-based
HQC	Code-based
LAC	Lattice-based
LEDAcrypt	Code-based
NewHope	Lattice-based
NTRU	Lattice-based
NTRU Prime	Lattice-based
NTS-KEM	Code-based
ROLLO	Code-based
Round5	Lattice-based
RQC	Code-based
SABER	Lattice-based
SIKE	Isogeny-based
THREE BEARS	Lattice-based

- Classic McEliece seems likely to enter the 3rd round
- https://csrc.nist.gov/projects/post-quantum-cryptography/

Classic McEliece highlights

Cons:

- Very big public keys
- Slow key genearation (but keys can be reused)

Pros:

- Security asymptotics unchanged by more than 40 years of cryptanalysis.
- Efficient and straightforward conversion of OW-CPA PKE into IND-CCA2 KEM.
- Very short ciphertexts.
- Constant-time software implementations.
- Fast encapsulation and decapsulation.
- Open-source (public domain) implementations.
- Patent-free.

Classic McEliece: proposed parameter sets

	mceliece348864	**mceliece460896**	**mceliece6688128**	**mceliece6960119**	**mceliece8192128**
(n, m, t)	$(3488, 12, 64)$	$(4608, 13, 96)$	$(6688, 13, 128)$	$(6960, 13, 119)$	$(8192, 13, 128)$
Public-key size	261120 bytes	524160 bytes	1044992 bytes	1047319 bytes	1357824 bytes
Secret-key size	6452 bytes	13568 bytes	13892 bytes	13908 bytes	14080 bytes
Ciphertext size	128 bytes	188 bytes	240 bytes	226 bytes	240 bytes
Key-gen time	52415436 cycles	181063400 cycles	467870488 cycles	417271280 cycles	424239104 cycles
Encapsulation time	43648 cycles	77380 cycles	140632 cycles	143908 cycles	187976 cycles
Decapsulation time	130944 cycles	267828 cycles	315920 cycles	295628 cycles	318484 cycles

See https://bench.cr.yp.to/results-kem.html#amd64-hiphop for the latest numbers.

Speed optimizations – McBits

- McBits (Bernstein, Chou, Schwabe, CHES 2013)
  - Non-conventional algorithms for fast constant-time decryption
  - Exploits external paralellism from multiple instances
  - High-throughput (but high-latency)

Speed optimizations – McBits

- McBits (Bernstein, Chou, Schwabe, CHES 2013)
  - Non-conventional algorithms for fast constant-time decryption
  - Exploits external paralellism from multiple instances
  - High-throughput (but high-latency)

- McBits revisited (Chou, CHES 2013)
  - Almost the same algorithms
  - Exploits internal paralellism in each algorithm
  - High-troughput, low-latency

Speed optimizations – faster key generation

Using semi-systematic form instead of systematic form (Chou, 2019).

- In systematic form, the i-th pivot must lie in the i-th column. In semi-systematic form, the last few μ pivots can lie in $\nu > \mu$ columns.
- Failure probability of public-key generation is reduced from $\approx 71\%$ to $\approx 2^{\mu-\nu}$ while preserving security.
- Leads to 'f' parameter sets of Classic McEliece with $\mu = 32, \nu = 64$.
- See `https://classic.mceliece.org/nist/mceliece-20190331.pdf` for more details.

Speed optimizations – faster key generation

Using semi-systematic form instead of systematic form (Chou, 2019).

- In systematic form, the i-th pivot must lie in the i-th column. In semi-systematic form, the last few μ pivots can lie in $\nu > \mu$ columns.
- Failure probability of public-key generation is reduced from $\approx 71\%$ to $\approx 2^{\mu-\nu}$ while preserving security.
- Leads to 'f' parameter sets of Classic McEliece with $\mu = 32, \nu = 64$.
- See `https://classic.mceliece.org/nist/mceliece-20190331.pdf` for more details.

Almost in-place LUP decomposition (Chou, 2020).

- Reduce the leftmost $(n-k) \times (n-k)$ matrix M to lower-triangular L, upper-triangular U, and permutation matrix P, such that

$$LPM = U \implies U^{-1}LP = M^{-1}.$$

- L and U are stored in the space of M.
- P is stored as an array of $n-k$ row indices.
- Reduces working set, helps caching.