Public Key Cryptanalysis, Part I (Codes)

Lecture 1: Linear Codes

Alexander Ray, Ruhr-University Bochum
Preliminaries

Lecture: Tue 12:30 - 14:00 1A 02/480
Exercise: Tue 14:15 - 15:45 1A 02/480 (every 2nd week Pre-calculation exercise)
Script: This one, nothing else. But please start to read papers.
Videos: good question

Bonus for exam: 1% for every 10% of exercise points
We want you to learn cryptanalysis (of codes/lattices) + SageMath
Overview (Part I - Codes, not yet Part II - Lattices)

1) Def. linear codes, vector space $\mathbb{F}_2$, duality ← hope you like linear algebra
2) McEliece (primal/dual, abstract) ← our favourite (old and good)
3) Brute Force, Meet-in-the-Middle on message
4) Time-Memory Tradeoff: van Oorschot - Wiener
5) Information Set Decoding: Prouge, Stern, MMT
6) Locality Sensitive Hashing
7) McEliece: Goppa codes ← how we really do it
8) Partial Key Exposure attacks ← my personal favourite :>)
9) HQC/BIKE cryptosystem ← efficient schemes inspired by lattices
10) Quantum speedups via Grover search ← oh, also some quantum
Teaser for McEliece (Robert McEliece ’78, Standard Classic McEliece ’23?)

Offer 128-bit security

Public key \( pk \) (hopefully)

\[ k = 2720 \]

\[ n = 3488 \]

\[ \begin{array}{c}
011000 \\
\vdots
\end{array} \]

\[ G \]

Size of \( G \):

\[ 2720 \cdot 3488 \text{ bit} \approx 9713 \]

(compared with RSA-4096)

128-bit security: Best algorithm requires \( \geq 2^{128} \) steps classically, forget about quantum (for now).

Encryption:

\[ c = pk \cdot m \cdot G + e \]

\[ m \]

Notion:

\[ \begin{array}{c}
\vdots \\
\hline
14
\hline
12
\hline
10
\hline
8
\hline
6
\hline
4
\hline
2
\hline
0
\end{array} \]

Ciphertext, added error, small no. of ones (64)
Finite Fields

in German: Körper (only amateurs say field!)

field $\mathbb{F}_2 = \{0, 1\}$: arithmetic mod 2

$\{0, 1\}$ is additive group with neutral element 0

$\{1\}$ is multiplicative group with neutral element 1

<table>
<thead>
<tr>
<th>+</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$\cdot$</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Exercise: $\mathbb{F}_3, \mathbb{F}_5$

What about $\mathbb{F}_4$?

Fact: For every prime $p$: $\mathbb{F}_p = \{0, 1, \ldots, p-1\}$, $\cdot$) is a finite field (with arithmetic mod $p$).

For every prime power $p^n$: $\mathbb{F}_{p^n}$ with $p^n$ elements exists.

Theorem without proof (for laziness reasons) Construction later. Warning: not $\mathbb{F}_p^n$. 
Vector space $\mathbb{F}_2^n$

**Theorem:** $\mathbb{F}_2^n = (\mathbb{F}_2, \mathbb{F}_2^n, +, \cdot)$ is a vector space.

**Proof:**

1. **Associativity:** $x + (y + z) = (x + y) + z$
2. **Commutativity:** $x + y = y + x$
3. **Neutral element:** $0^n + x = x + 0^n = x \quad \forall x \in \mathbb{F}_2^n$
4. **Inverse:** $\forall x \in \mathbb{F}_2^n: x + x = 0^n$, i.e., $-x = x$.
5. **Scalar multiplication:** $\alpha(x + y) = \alpha x + \alpha y \quad \forall \alpha \in \mathbb{F}_2$

Great! We never need a minus, almost throughout the lecture!
Subspaces of $\mathbb{F}_2^n$

**Def:** $S \subseteq \mathbb{F}_2^n$ is a subspace iff

- $0^n \in S$ and
- $\forall x, y \in S: x + y \in S$ (neutral element)

In other words: $S \subseteq \mathbb{F}_2^n$ is a subspace iff $S$ is a vector space.

(maybe a good exercise)

**Examples:**

- $\{000, 100, 010, 110\} \subseteq \mathbb{F}_2^3$ is a subspace.
- $\{000, 100\} \subseteq \mathbb{F}_2^3$ is a subspace, but not $\{100\}$. 

if and only if (\(\Rightarrow\))

closedness
Generating system and Basis

**Def:** Let \( S \subseteq \mathbb{F}_2^n \) be a subspace.

We call \( G = \{ g_1, \ldots, g_k \} \subseteq S \) generating scheme for \( S \) if

\[ \forall x \in S \exists x_1, \ldots, x_k \in \mathbb{F}_2^n : x = x_1 g_1 + \cdots + x_k g_k. \]

We write \( x \) as a linear combination of vectors in \( G \).

**Notation:** \( S = \langle g_1, \ldots, g_k \rangle = \langle G \rangle \) if \( S \) is generated by \( g_1, \ldots, g_k \).

We call a generating scheme \( G \) basis, if it is minimal, i.e.

\( \forall G' \subset G : \langle G' \rangle + S \Rightarrow \text{no subset of } G \text{ generates } S \)

**Examples:**

\( S_n = \{ 000, 100, 010, 110 \} \subseteq \mathbb{F}_2^n \).

\( G_n = \{ 100, 010, 110 \} \) generates \( S_n \).

\( B_n = \{ 100, 010 \} \) and \( B_2 = \{ 100, 110 \} \) are bases of \( S \).
Some basic basis facts (you know, too lazy to prove, recall linear algebra)

Fact: Let $S \subseteq \mathbb{F}_2^n$ be a subspace. $S_4 = \{000, 100, 010, 110\}$

1. Every linearly independent subset of $S$ can be extended to a basis.
   - $\{000\} \cup \{010\}$ is a basis

2. Every basis of $S$ has the same cardinality, called $\dim(S)$. $\dim(S_4) = 2$

3. Every generating scheme $G$ of $S$ contains a subset that is a basis.
   - $\{000, 010, 110\} \setminus \{010\}$ is a basis
Linear Code. We omit binary, since we always work with the binary field \( \mathbb{F}_2 \).

**Def:** A (binary) linear code \( C \) is a subspace of \( \mathbb{F}_2^n \).

Let \( k = \dim(C) \). Any basis \( G \in \mathbb{F}_2^{k \times n} \) is called a generator matrix.

Notice that \( C = \{ x \in \mathbb{F}_2^n | x \in \mathbb{F}_2^k \} \) and therefore \( |C| = 2^k \).

Great for crypto: We compactly represent \( 2^k \) codewords from \( C \) with only \( k \cdot n \) bits.

**Example:** Repetition code \( R(3) \)

\[
G = \begin{pmatrix}
1 & 1 & 1 & \cdots & 1 & 1
\end{pmatrix} \in \mathbb{F}_2^{6 \times 3}
\]

\((x_1 x_2 \ldots x_6). G = x_1 x_2 x_3 x_4 x_5 x_6 \)
Public Key Cryptanalysis, Part I (Codes)

Lecture 2: Distance

Alexander Ray, Ruhr-University Bochum
Hamming Distance

**Def:** Let $x \in \mathbb{F}_2^n$. We define the support of $x$ as
$$\text{supp}(x) = \{i \in \mathbb{N} \mid x_i \neq 0\}.$$  

The Hamming weight of $x$ is defined as
$$w(x) = |\text{supp}(x)|.$$  

The distance of $x, y \in \mathbb{F}_2^n$ is defined as
$$d(x, y) = w(x + y).$$  

*Example:*
$$\text{supp}(0110) = \{2, 3\},$$  
$$w(0110) = 2,$$  
$$d(0110, 1000) = 3.$$
Hamming defines a metric

**Theorem:** For any \( x, y, z \in \mathbb{F}_2^n \) we have:

1. **Positivity:** \( d(x, y) \geq 0 \), with \( d(x, y) = 0 \) iff \( x = y \)
2. **Symmetry:** \( d(x, y) = d(y, x) \)
3. **Triangle inequality:** \( d(x, z) \leq d(x, y) + d(y, z) \)

**Proof of (3):** Assume \( d(x, z) > d(x, y) + d(y, z) \).

By flipping bits, we first change \( x \) to \( y \), then \( y + z \).

This requires flipping of \( d(x, y) + d(y, z) < d(x, z) \) bits. ✓ (too few flips)
Def: Let $C = \mathbb{F}_2^n$. Then the distance of $C$ is defined as $\min_{c \in C} d(c, 0) = \min_{c \in C} |c|_1$.

Theorem: Let $C = \mathbb{F}_2^n$ a linear code. Then $d(C) = \min_{c \in C} \min_{\bar{c} \in C} d(c + \bar{c}, 0)$.

Def: Let $C = \mathbb{F}_2^n$. Then the distance of $C$ is defined as $\min_{c \in C} d(c, 0) = \min_{c \in C} |c|_1$.

Proof: \[ d(C) = \min_{c \in C} \min_{\bar{c} \in C} d(c + \bar{c}, 0) \]
\[ = \min_{c \in C} \min_{\bar{c} \in C} d(c, 0) + d(\bar{c}, 0) \]
\[ = \min_{c \in C} |c|_1 + \min_{\bar{c} \in C} |\bar{c}|_1 \]
\[ = \min_{c \in C} |c|_1 \]

Proof: \[ d(C) = \min_{c \in C} |c|_1 \]
\[ = \min_{c \in C} \sum_{i=1}^{n} c_i \]
\[ = \min_{c \in C} \sum_{i=1}^{n} c_i \]
\[ = \min_{c \in C} \sum_{i=1}^{n} c_i \]
\[ = \min_{c \in C} |c|_1 \]
**Hamming ball**

**Notation:** A linear code $C \subseteq \mathbb{F}_2^n$ with $\dim(C) = k$ and $d(C)$ is denoted $[n,k,d]$-code. $R(3)$ is an $[3k,k,3]$-code.

**Def:** Let $x \in \mathbb{F}_2^n$ and $r \in \mathbb{N}$. The Hamming ball with center $x$ and radius $r$ is

$B^r(x,r) = \{ y \in \mathbb{F}_2^n \mid d(x,y) \leq r \}$.

We define the volume of $B^r(x,r)$ as $V^r(r) := |B^r(x,r)|$.

$B^4(0110,1) = \{ 0110, 1110, 0010, 0100, 0111 \}$, $V^4(1) = 1 + 4 = 5$

**Theorem:** $V^r(r) = \sum_{i=0}^{r} \binom{n}{i}$

**Proof:** There are $\binom{n}{i}$ vectors in distance $i$ of some center.
Entropy and Binomials

Notation: $3n^2 + 2n + 1 = \Theta(n^2)$, $2n^3 \cdot 2^n = \tilde{\Theta}(2^n)$

Fact (Stirling): $n! = \Theta((\frac{n}{e})^n)$

Note than $n^n = 2^n \log n$. Growth of $n!$ is faster than exponential in $n$.

Def: Let $p < 1$. Then

$$H(p) := -p \cdot \log p - (1-p) \cdot \log (1-p). \quad \text{Binary entropy}$$

Theorem: $(\frac{n!}{i!}) = \tilde{\Theta}(2^{\#(\frac{i}{n})} n)$

Proof:

$$\binom{n}{i} = \frac{n!}{(n-i)! \cdot i!} = \tilde{\Theta} \left( \frac{n^n}{(n-i)^{n-i} \cdot i!} \cdot e^{n-i} \cdot e^i \right) = \tilde{\Theta} \left( \frac{n^n}{(n(1-\frac{i}{n}))^{n-i} \cdot (n \cdot \frac{i}{n})^i} \right) = \tilde{\Theta} \left( \frac{n^n}{(1-\frac{i}{n})^{n-i} (\frac{i}{n})^i} \right) = \tilde{\Theta} \left( 2^{\#(\frac{i}{n})} \cdot n \right)$$
Packing radius & unique decoding

Def: Let $C \subseteq \mathbb{F}_2^n$ be a code. $C$'s packing radius is

$$\text{pr}(C) := \max_{r \in \mathbb{N}} \{ B^n(c, r) \text{ are disjoint for all } c \in C \}$$

Corollary: $\text{pr}(C) = \left\lfloor \frac{d-1}{2} \right\rfloor$.

Note: Every point in $B^n(c, \text{pr}(C))$ allows for unique decoding to $c$.

$d(R(3)) = 3 \Rightarrow \text{pr}(R(3)) = 1$

$R(3)$ allows for correcting one error (per block) via majority decision.

010 $\mapsto$ 000, 011 $\mapsto$ 111
Singleton Bound

Theorem: Any $[n,k,d]$-code $C$ satisfies $d \leq n-k+1$.

Proof: Let us remove the last $d-1$ entries of every codeword. Since $C$ has distance $d$, all shortened (punctured) codewords of length $n-(d-1)$ are different. But there exist at most $2^{n-(d-1)}$ different vectors of length $n-(d-1)$.

\[ |C| \leq 2^{n-(d-1)} \Rightarrow n-k+1 \leq d \leq n-k+1. \]

RC) is $[3k, k, 3]$. $3 \leq 3k-k+1$ // Pretty far away for large $k$
**GV Bound (Gilbert-Varshamov)**

**Theorem:** There exists an Ink$K$-code with distance $d$ satisfying
\[ H\left(\frac{d}{n}\right) \geq 1 - \frac{k}{n} \geq \frac{d}{n} - 1 \]

**Proof sketch:** Let $C \subseteq \mathbb{F}_2^n$ have maximal $k = \text{dim}(C)$ among all linear codes with distance $d$. Then $\bigcup_{\text{CCC}} B_{\text{CCC}}^{n}(d-1) = \mathbb{F}_2^n$.

(Otherwise we can add $v \in \mathbb{F}_2^n \setminus \bigcup_{\text{CCC}} B_{\text{CCC}}^{n}(d-1)$ to the basis)

\[ \Rightarrow Z^n = \left| \bigcup_{\text{CCC}} B_{\text{CCC}}^{n}(d-1) \right| \leq \sum_{\text{CCC}} |B_{\text{CCC}}^{n}(d-1)| = 1C \cdot V^{n}(d-1) \approx 2^k \cdot 2^{H\left(\frac{d}{n}\right) n} \]

\[ \Rightarrow H\left(\frac{d}{n}\right) \geq 1 - \frac{k}{n} \]

**Fact:** Codes with random $G \in \mathbb{F}_2^{k \times n}$ achieve max. distance $H\left(\frac{d}{n}\right) \approx 1 - \frac{k}{n}$. 
Public Key Cryptanalysis, Part I (Codes)

Lecture 3: Duality

Alexander Ray, Ruhr-University Bochum
**Inner Product**

**Def:** Let \( x, y \in \mathbb{F}_2^n \) with inner product
\[
\mathbb{F}_2^n \times \mathbb{F}_2^n \rightarrow \mathbb{F}_2, \quad (x, y) \rightarrow \langle x, y \rangle = \sum_{i=1}^{n} x_i y_i
\]

**Facts:**
1. **Symmetry:** \( \langle x, y \rangle = \langle y, x \rangle \)
2. **Bilinearity:** \( \langle x + y, z \rangle = \langle x, z \rangle + \langle y, z \rangle \)
3. **Scalar Associativity:** \( \langle \alpha x, y \rangle = \alpha \langle x, y \rangle = \langle x, \alpha y \rangle \) for \( \alpha \in \mathbb{F}_2 \)

**Def:** We call \( x, y \) orthogonal if \( \langle x, y \rangle = 0 \).

**Exercise:** Show that every \( x \in \mathbb{F}_2^n \setminus \emptyset \) is orthogonal to half of the vectors in \( \mathbb{F}_2^n \).

*Note of orthogonality in \( \mathbb{F}_2^n \).*
Orthogonal Complement

**Def**: Let $C \subseteq \mathbb{F}_2^n$ be a linear code. We denote the orthogonal complement of $C$ by $C^\perp = \{ x \in \mathbb{F}_2^n \mid \langle c, x \rangle = 0 \text{ for all } c \in C \}$. 

**Example**: Let $C$ be generated by $G = (1001)$. Elements $x \in C^\perp$ satisfy

\[
\begin{align*}
\langle 1011, x \rangle &= 0 \\
\langle 1001, x \rangle &= 0 \\
\langle 0001, x \rangle &= 0
\end{align*}
\]

\[
\begin{align*}
x_1 + x_3 + x_4 &= 0 \\
x_1 + x_4 &= 0 \\
x_3 &= 0
\end{align*}
\]

\[
\Rightarrow C^\perp = \{0000, 1001, 0100, 1101^T\}
\]

generated by $(1001)$.

Why does the basis suffice? Always a linear code?
Dual Code

**Theorem:** $C^\perp \subseteq \mathbb{F}_2^n$ is a linear code, called the dual code of $C$.

**Proof:**
1. $0^n \in \mathbb{F}_2^n$
2. Let $x, y \in C^\perp$. Then we have for all $c \in C^\perp$

$$\langle x+y, c \rangle = \langle x, c \rangle + \langle y, c \rangle = 0 \Rightarrow x+y \in C^\perp.$$
**Parity Check Matrix**

**Def:** Let \( C \subseteq \mathbb{F}_2^n \) be a linear code. Any matrix \( P \) is called a parity check matrix of \( C \) if \( C = \{ x \in \mathbb{F}_2^n \mid P \cdot x = 0 \} \). \(< \text{May define } C \text{ via } G \text{ or } P.\>

**Theorem:** Let \( C \) be generated by \( G \subseteq \mathbb{F}_2^{m \times n} \).

1. \( C^\perp = \{ x \in \mathbb{F}_2^n \mid Gx = 0 \} \), i.e., \( G \) is a parity check matrix of \( C^\perp \).
2. \( \dim(C^\perp) = n - \dim(C) = n - k \)
3. \( C^\perp = C \)

**Proof:**
1. Generalize example from two pages before.
2. By 1: \( G \cdot x = [G_1 | G_2] \cdot x = 0 \) (wlog \( \text{rank}(G_1) = k \))

\[
\Rightarrow [I_k \mid G_1^{-1} \cdot G_2] \cdot x = 0 \text{ with free choice of } n-k \text{ vars of } x \Rightarrow \dim(C^\perp) = n-k
\]
We show \( \{ a \} \ C \subseteq C^\perp \wedge \dim(C) = \dim(C^\perp) \implies C = C^\perp. \)

(a) Let \( c \in C. \) Since \( C^\perp = \{ x \in \mathbb{F}_2^n \mid \langle c, x \rangle = 0 \text{ for all } c \in C \}, \)
we conclude \( c \in C^\perp = \{ y \in \mathbb{F}_2^n \mid \langle x, y \rangle = 0 \text{ for all } x \in C^\perp \}. \)

(b) \( \dim(C^\perp) \odot n - \dim(C^\perp) \odot n - (n - \dim(C)) = \dim(C) \)

Corollary: Let \( C \) be a linear code, and let \( G \) generate \( C^\perp. \)

Then by \( \odot G \) is a parity check matrix of \( C^\perp \odot C. \)

Ensures existence of parity check matrix for \( C, \) phew.
Some McEliece implications

Recall: McEliece's pk is generator matrix \( G \).

Problem: pretty large.

Ideas:

1. Take parity check matrix \( P \). \( n-k=768 \)
   Saves already factor of \( \frac{2720}{768} \approx 4 \) in size.

2. Take systematic form of \( P \).
   Saves another \( 768^2 \) bits.

Exercise: Show that a random matrix from \( \mathbb{F}_2^{n \times n} \) is invertible with probability
\[
\prod_{i=0}^{n-1} \left( 1 - \frac{1}{2^{n-i}} \right) > 0.288.
\]
Equivalent codes

Def: Let $C$ have basis $G \in \mathbb{F}_2^{k \times n}$. $C'$ is equivalent to $C$ if there exists an invertible matrix $S \in \mathbb{F}_2^{k \times k}$ and a permutation matrix $P \in \mathbb{F}_2^{n \times n}$ such that $C'$ is generated by $G'$.

Another basis of $C$ permutes columns, i.e., codeword positions performs row operations, e.g., Gauss elimination

Notice: Equivalent codes have the same parameters $[n,k,d]$. 

Exercise: Show that any $C$ has an equivalent code with generator/parity check matrix in systematic form.
From generator to parity (and vice versa) \( F_2^{e(n-k)} \)

**Theorem:** Let \( C \) be generated by \( G = [I_k | A] \in F_2^{e \times n} \).

Then \( P = [H^T | I_{n-k}] \in F_2^{(e-k) \times n} \) is a parity check matrix for \( C \).

**Proof:** Let \( C' \) be the code with parity check matrix \( P \). We show \( C' = C \) via

1. **\( C' \subseteq C \):** For every row \( g_i \) of \( G \) we have \( Pg_i^T = 0 \), since its \( j \)-th entry is \((a_{ij} \ldots a_{ij} 0 \ldots 1 \ldots 0) \cdot (0 \ldots 1 \ldots 0 \ldots a_{ij} \ldots a_{ij}) = a_{ij} + a_{ij} = 0 \).

2. **\( \dim(C') = \dim(C) \):** \( (C')^T \) has generator \( P \in F_2^{(n-k) \times n} \)

\[ \Rightarrow \dim(C') = n - \dim(C')^T = k = \dim(C). \]
Public Key Cryptanalysis, Part I (Codes)

Lecture 4: Brute Force & Meet in the Middle

Alexander Ray, Ruhr-University Bochum
Syndrome

Def: Let $C$ be an $[n,k,d]$-code with parity check $P = k \times n$. Let $c \in \mathbb{F}_2^n$. We define the syndrome of $c$ as $s(c) = P \cdot c^t$.

Corollary: $C = \{ x \in \mathbb{F}_2^n | s(x) = 0^n \}$

Let $c = m \cdot G + e$. McEliece ciphertext

Then $s(c) = Pc^t = P \cdot (mG + e)^t = P \cdot ((mG)^t + e^t) = P \cdot (G^t \cdot m^t + e^t) = P \cdot G^t \cdot m^t + P \cdot e^t = s(e)$

Syndrome of $c$ only depend on $e$. 
Syndrome Decoding and Subset Sum

**Def:** We denote $\Phi_{2}^{n}(S) = \{ x \in \Phi_{2}^{n} \mid \omega(x) = eS \}$. Recall that $|\Phi_{2}^{n}(S)| \approx 2^{h(\frac{e}{n})}n$.

**Syndrome Decoding Problem** (NP-hard for random $P$)

**Given:** $P \in \Phi_{2}^{(n-2)xn}$, $s(e)$ for some $e \in \Phi_{2}^{n}(d-1)$

**Find:** $e$ \quad $P \cdot e^t = s$ \quad Find $\left\lfloor \frac{d-1}{2} \right\rfloor$ columns of $P$ that sum to $s$.

**Subset Sum Problem** (good for developing algorithms)

**Given:** $a_1, \ldots, a_n \in \mathbb{Z}_{2^n}$, target $t \in \mathbb{Z}_{2^n}$

**Find:** $e = (e_1, \ldots, e_n) \in \{0,1\}^n$ \quad $\sum_{i=1}^{n} e_i a_i = t \mod 2^n$ \quad Assume that solution $e$ exists and is unique.
McEliece reloaded

**primal**

\[ c := m \cdot G + e \in \mathbb{F}_2^n \]

Cipher text is erroneous codeword.

**dual**

\[ s(c) = P \cdot e^t \in \mathbb{F}_2^{n-k} \]

Cipher text is syndrome.

**Benefits of dual:**
1. (way) shorter public key
2. (way) shorter ciphertext

**Problem:** We have to encode \( m \) into \( e \in \mathbb{F}_2^n \).

**Exercise:** Provide an efficiently computable bijection

\[ \text{Encode} : \mathbb{F}_2^{\#(\frac{t}{n}) \cdot n} \rightarrow \mathbb{F}_2(t), \ m \mapsto \text{Encode}(m) \]

\[ n = 3488 \]

\[ k = 2720 \]

\[ n-k = 768 \]
Brute Force

Brute Force Strategy: (1) enumerate space of possible solutions. (2) Check for correctness of solution.

Brute Force Subset Sum

Input: \( a_1, \ldots, a_n, t \)

(1) For all \( \varepsilon \in \{0,1\}^n \) exponential in \( n \)

(2) If \( \sum_{i=1}^{n} \varepsilon_i a_i = t \mod 2^n \), output \( \varepsilon = (\varepsilon_1, \ldots, \varepsilon_n) \) poly\((n)\)

Correctness: obvious

Complexity: time \( \tilde{O}(2^n) \), space \( \tilde{O}(1) \)

\uparrow \text{size of search space for } \varepsilon
Meet in the Middle (MitM)

MitM Strategy: (1) Split solution space in two parts of equal size.
(2) Check via equation with one part on each side.

Idea for Subset Sum: (1) Split $S = (\{e_1, \ldots, e_{n/2}\} \cup \{e_{n/2+1}, \ldots, e_n\})$.
(2) MitM identity: $\sum_{i=1}^{n/2} e_i a_i = t - \sum_{i=n/2+1}^{n} e_i a_i \mod 2^n$

For all $e^{(1)} = (e_1, \ldots, e_{n/2}) \in \mathbb{Z}_{2^{n/2}}$: size $2^{n/2}$

(a) For all $e^{(1)} = (e_1, \ldots, e_{n/2}) \in \mathbb{Z}_{2^{n/2}}$:
(b) Construct list $L$ with entries $\left(\sum_{i=1}^{n/2} e_i a_i, e^{(1)}\right)$, sorted by $1^{st}$ entry.

(2a) For all $e^{(2)} = (e_{n/2+1}, \ldots, e_n) \in \mathbb{Z}_{2^{n/2}}$: size $2^{n/2}$

(2b) If $(t - \sum_{i=n/2+1}^{n} e_i a_i, e^{(2)}) \in L$, output $e = (e^{(1)} \cup e^{(2)})$. 
Correctness: still obvious, hopefully 😊

Complexity (time): Steps (1a)/(2a): $2^{\frac{n}{2}}$ each only $\tilde{O}(1)$

Step (1b): Sort $2^{\frac{n}{2}}$ elements in time $\Theta(2^{\frac{n}{2}} \cdot \log(2^{\frac{n}{2}})) = \tilde{O}(2^n)$. \textcolor{red}{\textless \textgreater quick sort}

Step (2b): Search in sorted list of size $2^{\frac{n}{2}}$ in time $O(\log 2^{\frac{n}{2}}) = O(1)$. ← binary search, how?

Total time: $\tilde{O}(2^{\frac{n}{2}})$ ← square root of brute force, great!

Complexity (space): Storing list $L$ with $2^{\frac{n}{2}}$ entries costs space: $\tilde{O}(2^{\frac{n}{2}})$ ← less great, very costly in practice

Question: Can we gain in time, without sacrificing space?
Back to Decoding

Brute Force Syndrome Decoding

Input: $P, e \in \mathbb{F}_2^{n-1}$, $s(e)$ for some $e$ with $w(e) - l \leq \left\lfloor \frac{d-1}{2} \right\rfloor$

1. For all $e \in \mathbb{F}_2^n(l)$:
   2. If $P \cdot e^t = s$, output $e$.

Correctness: $\checkmark$

Complexity:
- Time $\tilde{O}(1 + \mathbb{F}_2^n(l)) = \tilde{O}(2^{\frac{d}{n}} \cdot n)$
- Space $\tilde{O}(1)$

\[ l = \omega(e) = 64 \]
\[ n = 3348 \]
\[ \Rightarrow H(\frac{e}{n}) \cdot n \approx 457 \]
MitM Decoding

Idea: (1) Split $e^t = \left( \frac{e_1}{e_n} \right) e^{(1)} e^{\frac{a}{2}}$ and $P = (P_1 \mid P_2)$.

(2) MitM identity: $P_1 \cdot e^{(1)} = s + P_2 \cdot e^{(2)}$

MitM Syndrome Decoding

Input: $P \in \mathbb{F}_2^{(n-k) \times n}$, $s(e)$ for some $e$ with $\text{wt}(e) = l$

(1a) For all $e^{(1)} \in \mathbb{F}_2^{\frac{n}{2}}$:

(1b) Construct list $L$ with entries $(P_1 e^{(1)} e^{(1)})$, sorted by 1st entry.

(2a) For all $e^{(2)} \in \mathbb{F}_2^{\frac{n}{2}}$:

(2b) If $(s + P_2 e^{(2)}) e^{(1)} \in L$, output $e = (e^{(1)}, e^{(2)})$. Let us assume for a moment: $\text{wt}(e_1) = \text{wt}(e_2) = \frac{l}{2}$.
Correctness: \( \checkmark \) 

Complexity: Time/Space \( \tilde{O}(\text{I.F.}^\frac{n}{2}(\frac{e}{2})) = \tilde{O}(2^{\frac{2n}{3}(\frac{e}{2})}) \) 

\( \uparrow \) Square root of Brute Force

Idea for balancing weight to \( \frac{e}{2} \) in \( e^{(1)}, e^{(2)} \):

1. Randomly permute the columns of \( P \).
2. This randomly permutes coordinates of \( e \).

Let \( X \) be the event that a random permutation balances weight.

\[ \mathbb{E}s[X] = \left( \frac{n}{2} \right)^2 \cdot \left( \frac{1}{e} \right)^2 = \tilde{O}\left( \frac{2^{4n(n+1)} \cdot \frac{e}{2}}{2^{4n(n-1)} \cdot n} \right) = \tilde{O}(1) \]  
Meaning \( \frac{1}{\mathbb{E}s[X]} = \tilde{O}(1) \)

I.e. we expect to reach balancedness after \( \frac{1}{\mathbb{E}s[X]} = \tilde{O}(1) \) iterations.
Public Key Cryptanalysis, Part I (Codes)

Lecture 5: Pollard Rho

Alexander May, Ruhr-University Bochum
McEliece Primal Attacks

Recall: \( c = m \cdot G + e \)

Brute Force on \( m \)

Input: \( c \in \mathbb{F}_2^n \), \( e = \omega(e) = 1^{\frac{n-1}{2}} \)

1. For all \( m \in \mathbb{F}_2^k \): 
   \[ \text{if } \omega(c + mG) = e, \text{ output } m \]

Correctness: Exercise: Why is there a unique \( m \) satisfying the \( \text{If} \) condition?

Complexity: 

- \( T = \tilde{O}(2^k) \)
- \( N = \tilde{O}(1) \)

Recall: \( k = 2720 \). Way worse than dual attack with \( 2^{457} \cdot 2^{457} \approx 2^{914} \).
MitM identity: Split \( m = (m_1, m_2) \in \mathbb{F}_2^{k/2} \times \mathbb{F}_2^{k/2} \) and \( G = (G_1) \).

\[
C + m_1 G_1 = m_2 G_2 + e
\]

Problem: \( e \) is also unknown.

MitM on \( m \) (not yet complete)

Input: \( G = (G_1) \in \mathbb{F}_2^{k/2}, c \in \mathbb{F}_2^k, \omega(e) = e - E_{\frac{k}{2}!} \)

1. For all \( m_1 \in \mathbb{F}_2^{k/2} \): Store \((c + m_1 G_1, m_1)\) in list \( L \), sorted by 1st entry.
2. For all \( m_2 \in \mathbb{F}_2^{k/2} \): If \((y, m_1) \in L\) with \( d(y, m_2 G_2) = e\), output \((m_1, m_2)\).

Requires approximate, instead of exact matching.

Will come back to this.

(Technique: Locality Sensitive Hashing)
Motivation: Small memory Subset Sum

Brute Force: Enumerate $E = (e_1, \ldots, e_n) \in \{0, 1\}^n$ s.t. $\sum_{i=1}^{n} e_i a_i = t \mod 2^n$

$$\begin{align*}
\text{Time: } \tilde{O}(2^n) & \quad \text{Memory: } \tilde{O}(1) \\
\end{align*}$$

Hit+M: Identity $\sum_{i=1}^{\frac{n}{2}} e_i a_i = t - \sum_{i=\frac{n}{2}+1}^{n} e_i a_i \mod 2^n$

$$\begin{align*}
\text{Time: } \tilde{O}(2^{n/2}) & \quad \text{Memory: } \tilde{O}(2^{n/2}) \\
\end{align*}$$

Question: Can we improve $2^n$ with memory $\tilde{O}(1)^2$?

Agenda: We look for collisions between $f_1: \epsilon_1, \ldots, \epsilon_{\frac{n}{2}} \mapsto \sum_{i=1}^{\frac{n}{2}} \epsilon_i a_i$ and $f_2: \epsilon_{\frac{n}{2}+1}, \ldots, \epsilon_n \mapsto t - \sum_{i=\frac{n}{2}+1}^{n} \epsilon_i a_i$. 
Collision Finding Problem

Given: Function $f : \{0,1\}^n \rightarrow \{0,1\}$
Find: $x_1 + x_2$ with $f(x_1) = f(x_2)$

Assume random property:
$$P_{x_1,x_2}[f(x_1) = f(x_2)] = \frac{1}{2^n}$$

Brute Force: Sample (expected) $2^n$ pairs $(x_1, x_2)$, until collision found.

MitM Collision Finding

1. For $i = 0$ to $2^n$:
   Compute list $L$ with entries $(f(x_i), x_i)$ for $x_i \in \{0,1\}^n$
2. If $L$ contains $(y, x_i), (y, x_j)$ with $x_i \neq x_j$, output $(x_i, x_j)$.
   Else output FAIL.
Correctness:

Complexity: Time and Space $\tilde{O}(2^{\frac{n}{2}})$

Success probability:

$\text{Bad: Event that } L \text{ does not contain a collision.}$

$\Rightarrow \Pr[\text{Bad}] = \prod_{i=0}^{2^{\frac{n}{2}}} (1 - \frac{1}{2^n}) \leq \prod_{i=1}^{2^{\frac{n}{2}}} e^{-\frac{1}{2^n}}$

$= e^{-\frac{2^{\frac{n}{2}}}{2^n}} = e^{-\frac{2^{\frac{n}{2}}(2^{\frac{n}{2}}+1)}{2^n}}$

$\leq e^{-\frac{1}{2}} \approx 0.6$

Alg MitM Collision Finding succeeds with $\Pr[\text{Bad}] > 1 - e^{-\frac{1}{2}} \approx 0.4$. 
Pollard Rho

Notation: We denote $f^i(x) = \underbrace{f(f(\ldots(f(x))\ldots))}_{i \text{ applications of } f}$.

Observation: Let $y, \lambda$ be minimal with $f^y(x) = f^{y+\lambda}(x)$.

Then $f^{y+i}(x) = f^{y+\lambda+i}(x)$ for all $i \geq 0$.

Intuition: We expect that $y + \lambda \approx 2^{\frac{n}{2}}$.

Idea Collision Finding:

- Start two kangaroos in $x$, one jumping two steps, the other only one.
- Iterate until both kangaroos are in same position (collision).

Question: Why do the kangaroos meet at all?
Algorithm Cycle Finding

Input: $f: \{0,1\}^n \rightarrow \{0,1\}^n$

1. Repeat $x \in \{0,1\}^n$ until $x \neq f(x)$.
2. Set $i=1$, $k_i = f(x)$, $k_{2i} = f(f(x))$
3. While $k_i \neq k_{2i}$ do
   
   $k_{i+1} = f(k_i)$, $k_{2(i+1)} = f(f(k_{2i}))$, $i := i + 1$

4. Set $l=0$, $k_e = x$.

5. While $f(k_e) \neq f(k_{e+1})$ do

   $k_{e+1} = f(k_e)$, $k_{e+i+1} = f(k_{e+i})$, $e := e + 1$

Output: $x_1 = k_e$, $x_2 = k_{e+i}$ with $f(x_1) = f(x_2)$, $x_1 \neq x_2$
Correctness: While loop 3 terminates with $k_i = k_{2i}$. We know that $k_j = k_j + c \cdot \lambda$, $c \in \mathbb{N}$ for all $j \geq \gamma$.

$$\Rightarrow i = c \cdot \lambda \quad \text{(a multiple of the cycle length)}$$

Loop 5: We start over with kangaroos in positions 0 and $i$, both with stepsize 1. After $\gamma$ hops they are in positions $\gamma$ and $\gamma + i$. Since $i$ is a multiple of the cycle length, both meet for the first time. There $x_1 = k_{\gamma + i}$ and $x_2 = k_{\gamma + i} - 1$ satisfy $f(x_1) = f(x_2)$ with $x_1 \neq x_2$.

Complexity (sketch): After $\gamma$ steps, both kangaroos are in the cycle. It takes $\leq \lambda$ steps, until they meet. Why? In total we require at most $\gamma + \lambda + \gamma = \tilde{O}(\lambda)$. Memory is $\tilde{O}(1)$. When both meet, the fast kangaroo made additional cycles.
The Rho Method

**Theorem:** In a function $f : \{0,1\}^n \rightarrow \{0,1\}^n$ we find a collision in expected time $\tilde{O}(2^n)$ using memory $\tilde{O}(1)$.

**Two function collision**

**Given:** $f_1 : \{0,1\}^n \rightarrow \{0,1\}^n$, $f_2 : \{0,1\}^n \rightarrow \{0,1\}^n$ with random properties

**Find:** $x_1, x_2$ with $f_1(x_1) = f_2(x_2)$

**Theorem:** In two functions $f_1, f_2 : \{0,1\}^n \rightarrow \{0,1\}^n$ we find a collision in expected time $\tilde{O}(2^n)$ using memory $\tilde{O}(1)$.

**Exercise:** Adapt Algorithm Cycle Finding for two functions.
Idea: Define $f_1: \{0,1\}^{\frac{n}{2}} \to \mathbb{Z}_{2^{\frac{n}{2}}}$ and $f_2: \{0,1\}^{\frac{n}{2}} \to \mathbb{Z}_{2^{\frac{n}{2}}}$

$$f_1(x) = \sum_{i=0}^{\frac{n}{2}} x_i \mod 2^{\frac{n}{2}}, \quad f_2(x) = \sum_{i=0}^{\frac{n}{2}} x_i \mod 2^{\frac{n}{2}}$$

Remark: There exist $2^n$ pairs $(x_1, x_2) \in \{0,1\}^{\frac{n}{2}} \times \{0,1\}^{\frac{n}{2}}$. Any pair is a collision with $\Pr_{x_1, x_2}[f_1(x_1) = f_2(x_2)] = 2^{-\frac{n}{2}}$. Thus, we expect $2^n \cdot 2^{-\frac{n}{2}} = 2^{\frac{n}{2}}$ collisions.

One of them is $(\varepsilon^{(1)}, \varepsilon^{(2)})$, sometimes called the golden collision.
Subset Sum Collision

1. Repeat: Find a random collision \((x^{(1)}, x^{(2)})\) s.t. \(f_1(x^{(1)}) = f_2(x^{(2)})\).
2. Until \(\sum_{i=1}^{n} x_i a_i = t - \sum_{i=n+x+1}^{n+y} x_i a_i \mod 2^n\).

Output: \((x^{(1)}, x^{(2)}) = (x^{(1)}, x^{(2)})\)

\[\begin{align*}
&\text{Correctness: } \checkmark \\
&\text{Complexity: } \text{Each iteration of Repeat in 1 costs } \tilde{O}(2^{\frac{n}{2}}). \\
&\quad \text{We expect to find } 2^{\frac{n}{2}} \text{ collisions, before recovering the golden } (x^{(1)}, x^{(2)}). \\
&\quad \Rightarrow \text{Expected total runtime: } 2^{\frac{n}{2}} \cdot \tilde{O}(2^{\frac{n}{2}}). \quad \text{Memory } \tilde{O}(n)
\end{align*}\]
Public Key Cryptanalysis, Part I (Codes)

Lecture 6: PCS & Representations

Alexander Ray, Ruhr-University Bochum
\[ \frac{3}{4} \text{ a Rho algo for Syndrome Decoding} \]

**Def**: Let \( \mathbf{v} = (v_1, \ldots, v_n) \in \mathbb{F}_2^n \). Then we define \( \mathbf{v}_{\leq k} = (v_1, \ldots, v_k) \) for \( k \leq n \).

**Idea**: NiH7 \( e \) \( \text{eq} \quad P_{e_1}(\ell) = s + P_{e_2}(\ell), \quad e_1, e_2 \in \mathbb{F}_2^\frac{n}{2}(\ell) \)

\( f_1 : \mathbb{F}_2^\frac{n}{2}(\ell) \rightarrow \mathbb{F}_2^\frac{n}{2}(\ell) \quad f_2 : \mathbb{F}_2^\frac{n}{2}(\ell) \rightarrow \mathbb{F}_2^\frac{n}{2}(\ell) \)

\( e_1 \mapsto [P_{e_1} e_1]_{\mathbb{F}(\ell)^{\mathbb{F}_2(\ell)}} \quad e_2 \mapsto [s + P_{e_2}(\ell)]_{\mathbb{F}(\ell)^{\mathbb{F}_2(\ell)}} \)

**Rho Decoding**

**Input**: \( P = (P_1, P_2) \in \mathbb{F}_2^{(e_{1,2}) \times n}, s \in \mathbb{F}_2^n, \omega(e) = e \)

1. **Repeat**: Find random collision \( (x^{(1)}, x^{(2)}) \) for \( f_1, f_2 \)
   
   Until \( P_1 x^{(1)} = S + P_2 x^{(2)} \)

2. **Output** \( e = (e^{(1)}, e^{(2)}) = (x^{(1)}, x^{(2)}) \)

**Correctness**: √
Complexity: Need $\tilde{O}(2^{\frac{2(d/2)}{2}})$ per collision.

There are expected $2^{\frac{2(d/2)}{2}}$ collisions.

Total time: $\tilde{O}(2^{\frac{d}{4} \cdot (d/2)} \cdot n)$. Memory: $\tilde{O}(1)$

Representation technique (2010)

Problem: Only a single golden collision.

Idea: Construct more useful collision by splitting $e$ differently.

Def: Let $e, e^{(1)}, e^{(2)}$ be $n$-dimensional vectors. We call $(e^{(1)}, e^{(2)})$ a representation of $e$ if $e = e^{(1)} + e^{(2)}$
Lemma: Assume $\omega t(\varepsilon) = \frac{n}{2}$. Then $\varepsilon$ has at least \(\frac{n+1}{4}\) representations.

Proof: We choose \(\frac{n}{4}\) out of \(\frac{n}{2}\) ones for $\varepsilon^{(1)}$. This completely determines $\varepsilon^{(2)}$.

Idea:

Let $f_1 : \{0,1\}^n(\frac{n}{4}) \rightarrow \mathbb{Z}_2^{|\psi(\frac{n}{4})|}$

\[ f_1 : \varepsilon^{(1)} \mapsto \sum_{i=4}^{n} \varepsilon^{(1)}_i a_i \mod 2^{0.81n} \]

Let $f_2 : \{0,1\}^n(\frac{n}{4}) \rightarrow \mathbb{Z}_2^{|\psi(\frac{n}{4})|}$

\[ f_2 : \varepsilon^{(2)} \mapsto t - \sum_{i=4}^{n} \varepsilon^{(2)}_i a_i \mod 2^{0.81n} \]

High Rep-Rho Subset Sum

Input: $a_1, \ldots, a_n, t \in \mathbb{Z}_2^n$

1. Repeat: Find random collision $(x^{(1)}, x^{(2)}) \in \{0,1\}^n$ for $f_1, f_2$

   Until $\sum_{i=1}^{n} (x^{(1)}_i + x^{(2)}_i) \cdot a_i = t \mod 2^n$ and $x^{(1)} + x^{(2)} \in \{0,1\}^n$.

2. Output $\varepsilon = x^{(1)} + x^{(2)}$.

Notice: Addition may produce $2s$. 

\[ 0.72n \text{ Subset Sum} \]
Complexity: Cost per collision is $\tilde{O}(2^{0.405n})$.

We expect $2^{0.8n}$ collisions, but all $\tilde{O}(2^{n/2})$ representations are good.

$\Rightarrow Pr[\text{collision is good}] = 2^{-0.31n}$.

Total expected time: $\tilde{O}(2^{0.405n}) \cdot 2^{0.31n} = \tilde{O}(2^{0.72n})$.

\[ f_1 : \mathbb{F}_2^n(\frac{n}{2}) \rightarrow \mathbb{F}_2^n(\frac{n}{2}) \quad f_2 : \mathbb{F}_2^n(\frac{n}{2}) \rightarrow \mathbb{F}_2^n(\frac{n}{2}) \]

\[ e^{(1)} \rightarrow \left[ \mathbb{P}_1 e^{(1)} \right]_{H(\frac{n}{2n})} \quad e^{(2)} \rightarrow \left[ s + \mathbb{P}_2 e^{(2)} \right]_{H(\frac{n}{2n})} \]

Algorithm Rep-Rho Syndrome Decoding

Input: $P = (P_1, P_2) \in \mathbb{F}_2^{(n-h)x_n}$, $s \in \mathbb{F}_2^n$, $e = \omega(e)$

1. Repeat: Find random collision $(x^{(1)}, x^{(2)}) \in (\mathbb{F}_2^n(\frac{n}{2}))^2$ for $f_1, f_2$

Until $P_1 x^{(1)} = s + P_2 x^{(2)}$ and $\text{wt}(x^{(1)} + x^{(2)}) \leq l$.

2. Output $e = x^{(1)} + x^{(2)}$. In general we only have $l \leq l$. 
Complexity: Cost per collision is $\tilde{O}(2^{\frac{e}{2}}n^{\frac{e}{2}})$.  

Number of representations: $(\frac{e}{2}) = \tilde{O}(2^e)$.  

$\Rightarrow \Pr[\text{collision is good}] = \tilde{O}(\frac{2^e}{2^{\frac{e}{2}}n}) = \tilde{O}(2^{e-H(\frac{e}{2})-n})$  

Expected total time: $\tilde{O}(2^{\frac{e-H(\frac{e}{2})-n-e}{384-64}})$ 

McEliece parameter before: 545, gain: 45 bits

The $1+1=0$ trick

Idea: We choose $f_1 : \mathbb{F}_2^n (\frac{e}{2}+\varepsilon) \to \mathbb{F}_2^n (\frac{e}{2}+\varepsilon)$.  

i.e., we want that in $e^{(1)} + e^{(2)}$ exactly $\varepsilon$ ones cancel via $1+1=0$.  

Tradeoff: Larger search space, but also more representations.

Exercise: Show that the optimal $\varepsilon$ is 4, saving another 5 bit.
Theorem: Let $f_0, \ldots, f_e : \{0,1\}^n \rightarrow \{0,1\}^n$. Then one can find collisions between $f_0$ and all of $f_1, \ldots, f_e$ in expected time $\tilde{O}(\sqrt{k} \cdot 2^n)$ and memory $\tilde{O}(k)$.

Proof idea: Need to store "distinguished" points at start points $f_0 \rightarrow f_1 \rightarrow \ldots \rightarrow f_e$.

Corollary: Let $f_0, f_1 : \{0,1\}^n \rightarrow \{0,1\}^n$. Then one finds $k$ collisions between $f_0, f_1$ in expected time $\tilde{O}(\sqrt{k} \cdot 2^n)$ and memory $\tilde{O}(k)$.

Proof: Choose $f_1 = f_2 = \ldots = f_e$. 

Proof idea: Time-Memory Tradeoff with $\tilde{O}(\sqrt{k} \cdot 2^n)$ length collisions and $\tilde{O}(k)$ memory.
Time Memory Tradeoff for Rho Subset Sum

Idea: Recall that we need $2^{\frac{n}{2}}$ collisions. Construct them using PCS.

Theorem: Using memory $N \leq 2^{\frac{n}{2}}$, Subset Sum can be solved in expected time $\tilde{O}(\frac{N}{\sqrt{N}})$.

Proof: With memory $N$ we construct $N$ collisions in time $\tilde{O}(\sqrt{N} \cdot 2^{\frac{n}{4}})$ per iteration. I.e., we require on expectation $2^{\frac{n}{4}}/N$ collisions.

$\Rightarrow$ Total expected run time: $2^{\frac{n}{4}}/N \cdot \tilde{O}(\sqrt{N} \cdot 2^{\frac{n}{4}}) = \tilde{O}(\frac{2^{\frac{3n}{8}}}{N^{\frac{1}{8}}})$.

Time Memory Tradeoff graphically:

- 0\(\frac{n}{4}\)\(\frac{n}{2}\)\(\log N\)
- $\frac{3}{4} \log N$ in Rho algo
- $\frac{n}{4}$ in tradeoff
- Shamir-Shroeppel (soon to come)
Theorem: Using memory $M \leq 2^{\frac{H(E)}{\log n}}$, Syndrome Decoding can be solved in expected time $\tilde{O}(2^{\frac{1}{3} \cdot H(E)} \cdot \frac{n}{\sqrt{H(E)}})$.

Proof: Analogous to previous proof.

Time-Memory Tradeoff for Syndrome Decoding

Graphically:

- $\frac{1}{3} \cdot H(E) \cdot n$
- $H(E) \cdot \frac{n}{2}$
- $\frac{1}{3} \cdot n$ Rho algo
- Our tradeoff
- Shamir-Shroeppel (soon to come)

Graph with axes $H(E)$ and $\log n$.
Public Key Cryptanalysis, Part I (Codes)

Lecture 7: Schroeppel-Shamir

Alexander Ray, Ruhr-University Bochum
Multi-Instance Attacks

Given: $a_1, \ldots, a_n, t_1, \ldots, t_{2^n}

Find: $e^{(i)} \in \{0,1\}^n$ s.t. $\sum_{i=1}^n e^{(i)} a_i = t_j \mod 2^n$ for all $j = 1, \ldots, 2^n$

1st solution: Store all $t_i$ in a sorted list $L$. \[ T = \tilde{O}(2^n) \]

For all $e \in \{0,1\}^n$: Search for $\sum_{i=1}^n e a_i \mod 2^n$ in $L$. \[ T = O(2^n) \]

2nd solution: Use 0.72n algo for each $t_i$. \[ T = O(2^{0.97n}) \]

3rd solution: Define $f_0: \{0,1\}^n \rightarrow \mathbb{Z}_{2^n}$ and $f_j: \{0,1\}^n \rightarrow \mathbb{Z}_{2^n}$ for $j = 1, \ldots, 2^n$

\begin{align*}
    f_0 &\colon e^{(i)} \mapsto \sum_{i=1}^n e^{(i)} a_i \mod 2^n \\
    f_j &\colon e^{(i)} \mapsto t_j - \sum_{i=1}^n e^{(i)} a_i \mod 2^n
\end{align*}

PCS: $2^n$ collisions $(e^{(i)}, e^{(ii)})$ per iteration in $\tilde{O}(2^{3.7} \cdot 2^n)$.

Require $2^n$ iterations. In total: \[ T = \tilde{O}(2^{3.7} \cdot 2^n) \]

Exercise: Use 0.72n algo with PCS. Apply 3rd solution to Decoding.
Schroeppel-Shamir Technique (1977)

Idea: Generalize Meet-in-the-Middle to 4 lists.

\[
\sum_{i=0}^{n/4} e_i a_i + \sum_{i=n/4+1}^{n/2} e_i a_i = t - \sum_{i=n/2+1}^{n} e_i a_i - \sum_{i=3n/4+1}^{n} e_i a_i \quad \text{with } e_i, \ldots, e_n \in \{0,1\}
\]

\[
\rightarrow \quad \rightarrow \quad \rightarrow \quad \rightarrow
\]

\[L_1 \quad \rightarrow L_2 \quad \rightarrow L_3 \quad \rightarrow L_4\]

We have to match these values. But how?

Solution: We guess the value \( r = \sum_{i=0}^{n/4} e_i a_i = t - \sum_{i=n/2+1}^{n} e_i a_i \mod 2^{n/4} \).

Schroeppel-Shamir algo

Input: \( a_1, \ldots, a_n, t \)

1. For all \( r = 0 \) to \( 2^{n/4} - 1 \):
   Compute \( L_1 = \{ e^{(n)}, \sum_{i=0}^{n/4} e_i a_i \} \)
   Compute \( L_2 = \{ e^{(n/2)}, \sum_{i=n/4+1}^{n/2} e_i a_i \} \)
   Compute \( L_3 = \{ e^{(n/4)}, \sum_{i=n/2+1}^{3n/4} e_i a_i \} \)
   Compute \( L_4 = \{ e^{(1)}, \sum_{i=3n/4+1}^{n} e_i a_i \} \)

2. Compute \( L_{12} = \{ e = (e^{(n)}, e^{(n/2)}) \mid \sum_{i=0}^{n/4} e_i a_i \equiv r \mod 2^{n/4} \} \)
3. Compute \( L_{34} = \{ e = (e^{(n/4)}, e^{(1)}) \mid t - \sum_{i=n/2+1}^{n} e_i a_i \equiv r \mod 2^{n/4} \} \)

If \( e = (e^{(n)}, e^{(n/2)}, e^{(n/4)}, e^{(1)}) \in L_{12} \times L_{34} \) with \( \sum_{i=0}^{n/4} e_i a_i = t \), output \( e \).
Schroeppel-Shamir graphically as 4-list algo:

\[
\begin{align*}
\{L_1, L_2, L_3, L_4\} & \quad 2^{\frac{n}{4}} \\
\{L_{12}, L_{14}, L_{24}, L_{34}\} & \quad 2^{\frac{3n}{4}}
\end{align*}
\]

Complexity: \(2^{\frac{m}{4}}\) iterations for guessing \(r\).

\(L_{12}\) can be constructed in expected time \(\tilde{O}\left(\frac{|L_{12}| |L_{14}| |L_{24}| |L_{34}|}{2^{\frac{n}{4}}}\right) = \tilde{O}(2^{\frac{n}{2}})\).

Analogous for \(L_{34}\). Total expected time: \(T = 2^{\frac{n}{2}} \cdot \tilde{O}(2^{\frac{n}{2}}) = \tilde{O}(2^n)\).

Memory: \(\tilde{O}(2^{\frac{n}{2}})\).

Exercise: Design and analyze an 8-list algo.
Howgrave-Graham Joux algo (2010)

Idea: Instead of guessing $r$, use representations $(x^{(1)}, x^{(2)}) \in \{0, 1\}^n(n)$ of $e$.

Construct $L_{a_2}, L_{a_3}$ s.t. $L_{a_2} \times L_{a_3}$ contains one representation.

$\text{HGG algo}$

Input: $a_1, \ldots, a_n, t$

1. Compute
   $$L_1 = \{ x^{(1)} \mid \sum_{i=1}^{n} e^{(1)}_i a_i \in \mathbb{Z}_n \},$$
   $$L_2 = \{ x^{(2)} \mid \sum_{i=1}^{n} e^{(2)}_i a_i \in \mathbb{Z}_n \},$$
   $$L_3 = \{ x^{(3)} \mid \sum_{i=1}^{n} e^{(3)}_i a_i \in \mathbb{Z}_n \},$$
   $$L_4 = \{ x^{(4)} \mid e - \sum_{i=1}^{n} e^{(1)}_i a_i \in \mathbb{Z}_n \}.$$

2. Compute
   $$L_{a_2} = \{ x^{(2)} = (e^{(1)}, e^{(2)}) \mid \sum_{i=1}^{n} e^{(2)}_i a_i = 0 \mod 2^n \},$$
   $$L_{a_4} = \{ x^{(4)} = (e^{(3)}, e^{(4)}) \mid t - \sum_{i=1}^{n} e^{(3)}_i a_i = 0 \mod 2^n \}.$$

3. If $e = e^{(1)} + e^{(2)} \in L_{a_2} \times L_{a_4}$ with $\sum_{i=1}^{n} e_i a_i = t \mod 2^n$, output $e$. Might be randomized. On expectation satisfied by a single representation.
**Algorithm 69** graphically:

```
L_1 \rightarrow L_2 \rightarrow L_3 \rightarrow L_4 \rightarrow \leftarrow unbalanced
```

**Complexity:** List sizes $|L_1| = |L_2| = |L_3| = |L_4| = \tilde{O}
\left(\left(\frac{1}{8}\right)^{\frac{1}{2}}\right) = \tilde{O}
\left(2^{\frac{1}{4}H(y)}\right) = \tilde{O}(2^{0.405n})$.

$L_{12}, L_{34}$ can be constructed in time $\tilde{O}\left(\frac{|L_1| \cdot |L_2|}{2^{n/2}}\right) = \tilde{O}(2^{0.34n})$.

**Total time and memory:** $\tilde{O}(2^{0.405n})$

**Exercise:** Use different representations to balance complexities.

**Fact:** Current record time/memory $\tilde{O}(2^{0.88n})$: more reps, larger depth.
Information Set Decoding (Prange’62)

Idea:

\[
\begin{bmatrix}
P_1 & P_2 \\
\end{bmatrix}
\begin{bmatrix}
e_1 \\
e_2 \\
\end{bmatrix}
= \begin{bmatrix}
s_1 \\
\end{bmatrix}
\]

Gauss

\[
\begin{bmatrix}
P_1^{-1} P_2 \\
\end{bmatrix}
\begin{bmatrix}
e_1 \\
e_2 \\
\end{bmatrix}
= \begin{bmatrix}
p_1^{-1} s \\
\end{bmatrix}
\]

\[
e_1 + P_1^{-1} P_2 e_2 = p_1^{-1} s
\]

For \( e_2 = 0^k \) we obtain \( e_1 = p_1^{-1} s \), and therefore the solution \( e = (e_1, 0^k) \).

Def: We call the first \( n-k \) columns of \( P \) an information set.

Prange’s idea: Permute \( P \) s.t. the information set contains all ones of \( e \).
Public Key Cryptanalysis, Part I (Codes)

Lecture 8: Information Set Decoding (ISD)

Alexander Ray, Ruhr-University Bochum
Information Set Decoding (Prange'62)

Idea:

\[
\begin{bmatrix}
P_1 & P_2 \\
\vdots & \vdots \\
P_{n-k} & P_k
\end{bmatrix} - S
\]

\[
\begin{bmatrix}
e_1 \\
e_2
\end{bmatrix}
\]

Gauss reduction:

\[
\begin{bmatrix}
I_{n-k} & P_1^{-1} \cdot P_2
\end{bmatrix}
\]

\[
P_n^{-1} \cdot s
\]

\[
e_1 + P_1^{-1} P_2 \cdot e_2 = P_n^{-1} s
\]

For \( e_2 = 0^k \) we obtain \( e_1 = P_1^{-1} s \), and therefore the solution \( e = (e_1, 0^k) \).

Def: We call the first \( n-k \) columns of \( P \) an information set.

Prange's idea: Permute \( P \) s.t. the information set contains all ones of \( e \).
Praugas ISD
Input: $P \in \mathbb{F}_2^{(n-k)\times n}$, $s \in \mathbb{F}_2^{n-k}$, $\omega = w(e)$

1. Repeat
2. Repeat. Choose random permutation matrix $H \in \mathbb{F}_2^{n \times n}$. Let $PH = (P_1 \mid P_2)$.

   Until $P_1$ is invertible.

   Until $w(P_1^{-1}s) = \omega$.

Output: $e = H \cdot (P_1^{-1}s, 0)$

$H^{-1}e = (e_1, e_2)$

Complexity: ② has polynomial complexity.

④ succeeds with probability $\frac{(n-k)}{(n^2)} = 2$.

$\tilde{T} = \tilde{O}(Z^{143}(n \log n)^3)$, $\Pi = \tilde{O}(1)$. 

Neilsen params $Z^{143}$ (NiTi: $2^{230}$)

WoW
Idea: Relax the requirement that all error positions land in information set.

\[
\text{Lins - Brickell ISD ('88)}
\]

\[
\text{Let } P = \begin{bmatrix} e_1 & e_2 \\ I_n & I_n \end{bmatrix} = P_1^{-1} \cdot P_2 \cdot e_2
\]

\[
e_1 = P_1^{-1} \cdot s + P_1^{-1} \cdot P_2 \cdot e_2
\]

Lee - Brickell algo:

Input: \(P \in \mathbb{F}_2^{(n-k) \times n}, s \in \mathbb{F}_2^{m}, \omega = w(e), p < \omega\)

1. Repeat
   1.1 Repeat: Choose permutation \(H \in \mathbb{F}_2^{n \times n}\). Let \((P_1, P_2) = PH\). Until \(P_1\) invertible.
   1.2 For all \(e_2 \in \mathbb{F}_2^k(p)\) ← Brute Force of \(e_2\)
   Until \(w(P_1^{-1} \cdot s + P_1^{-1} \cdot P_2 \cdot e_2) = \omega - p\).
2. Output \(e = H(P_1^{-1} \cdot s + P_1^{-1} \cdot P_2 \cdot e_2, e_2)\)
Complexity:

Lee-Brickell

$\Pr(H_{\text{good}}) = \frac{(n-k)}{(w-p)} \cdot \frac{\binom{\frac{k}{\rho}}{\rho}}{\binom{\frac{n}{w}}{w}}$

Better than Prange for $p > 0$

no cost in Prange

Since $w < \frac{n}{2}$, we maximize $(\frac{n-k}{w-p})$ for the choice $p = 0$

Identical to Prange

Question: Lee-Brickell identical to Prange? What's the point?

Well, use Mitt instead of Brute-Force.
MitM ISD (1st try)

$w-p$ $p/2$ $p/2$

$e_1$ $e_2$ $e_3$

MitM identity:
$e_1 + H_1 \cdot e_2 + H_2 \cdot e_3 = s$

3 unknowns, but only 2 sides

Solution 1: Remove unknown $e_1$ (next slide)

Solution 2 (LSHT): Use approximate identity $H_1 \cdot e_2 \approx e_1 \cdot s + H_2 \cdot e_3$

Locality sensitive hashing

(Identity on all but $w-p$ positions)

(maybe later: better, but a bit more advanced 😊)
Leon's removal of $e_1$ ('88)

Leon's $e$-window: Use semi-systematic form. Q: How to compute?

Let $G \cdot P_2 = \left( \frac{A}{B} \right)_{n-k-l} \times (k+e)$ and $s = \left( \frac{s_1}{s_2} \right)_{n-k-l}$. Then

1. $e_1 = s_1 + \frac{A}{B} \cdot e_2$
2. $O = s_2 + B \cdot e_2 \quad$ (we removed the annoying $e_1$)
Dunne-Stem ISD ('89)

Idea:

\[
\begin{array}{c|c|c}
\omega-p & \rho/2 & \rho/2 \\
\hline
e_1 & e_2 & e_3 \\
\end{array}
\]

\[
\begin{array}{c|c|c}
I_{n-k-l} & I_1 & I_2 \\
\hline
0 & B_1 & B_2 \\
\end{array}
\]

\[
\begin{array}{c|c|c}
\text{n-k-l} & \text{e} & \text{g.s} \\
\hline
\text{n-k-l} & \text{e} & \text{g.s} \\
\end{array}
\]

Identities:
1. \( I_1 \cdot e_2 = e_4, \ S_1 + I_2 \cdot e_3 \) \hspace{1cm} \text{approximate}
2. \( B_1 \cdot e_2 = s_2 + B_2 \cdot e_3 \) \hspace{1cm} \text{exact}

Strategy: First check (2), then (1).
Dumer-Stern ISD

Output: $P \in \mathbb{F}_2^{(n-k) \times n}$, $s \in \mathbb{F}_2^{n-k}$, $\omega = w(e)$, $p < \omega$, $e \leq n-k$

1. Repeat
   1.1 Choose permutation $\Pi \in \mathbb{F}_2^{n \times n}$.
   1.2 Compute semi-systematic form $G \cdot P^\Pi = (I_{n-k}, \overline{\Pi}_1, \overline{\Pi}_2)$, $G \cdot s = (s_1, s_2)$.
   1.3 For all $e_2 \in \mathbb{F}_2^{n-k}$: Compute $L$ with entries $(\overline{\Pi}_1, e_2, e_2)$
   1.4 For all $e_3 \in \mathbb{F}_2^{n-k}$:
       1.4.1 For all $(s_2 + \overline{\Pi}_2 e_3, e_2) \in L$: All $(e_2, e_3)$ satisfying (1) satisfy also (2)?

2. Output: $e = H(\overline{\Pi}_1 e_2 + \overline{\Pi}_2 e_3 + s_2, e_2, e_3)$

(We assume this is doable, otherwise repeat.)
Complexity:

1. Pr [H is good] = \( \frac{(n-k-e) \cdot (k+e)/2}{n} \) candidates in candidates in \( \mathcal{A}_{4.4} \)

2. \( |\mathcal{F}_{\frac{p}{2}} (\frac{k+e}{2})| = (\frac{k+e}{2}) \cdot (\frac{k+e}{2}) \cdot 2^{-e} \)

\[ \Rightarrow T = \frac{(n-k-e) \cdot (k+e)/2}{n} \cdot (\frac{k+e}{2}) \cdot \max \{1, (\frac{k+e}{2}) \cdot 2^{-e}\} \]

\[ M = \frac{(k+e)/2}{p/2} \]

The Elieen parameters: \( n = 3488, k = 2720, w = 64 \)

Prange for \( l = k = 0 \): \( T = 2^{438} \), no memory

Optimized \( p = 10, e = 46 \): \( T = 2^{438}, M = 2^{45} \)

5 bit save for quite heavy memory
<table>
<thead>
<tr>
<th>Length</th>
<th>Weight</th>
<th>Authors</th>
<th>Algorithm</th>
<th>Date</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>1347</td>
<td>25</td>
<td>Daniel J. Bernstein, Tanja Lange, Christiane Peters</td>
<td>See <a href="https://isd.mceliece.org/1347.html">https://isd.mceliece.org/1347.html</a> for more information.</td>
<td>2023-02-24</td>
<td></td>
</tr>
<tr>
<td>1284</td>
<td>24</td>
<td>Andre Esser, Alex May and Floyd Zweydinger</td>
<td>MMT variant</td>
<td>2021-08-16</td>
<td></td>
</tr>
<tr>
<td>1223</td>
<td>23</td>
<td>Andre Esser, Alex May and Floyd Zweydinger</td>
<td>BJMM/MMT variant</td>
<td>2021-05-10</td>
<td></td>
</tr>
<tr>
<td>1161</td>
<td>22</td>
<td>Shintaro Narisada, Kazuhide Fukushima, and Shinsaku Kiyomoto</td>
<td>Dumer</td>
<td>2021-01-10</td>
<td></td>
</tr>
<tr>
<td>1101</td>
<td>21</td>
<td>Anders Nilson</td>
<td>Multi threads Dumer4, Gregory Landais impl.</td>
<td>2020-08-14</td>
<td></td>
</tr>
<tr>
<td>1041</td>
<td>19</td>
<td>Shintaro Narisada, Kazuhide Fukushima, and Shinsaku Kiyomoto</td>
<td>Dumer</td>
<td>2020-08-11</td>
<td></td>
</tr>
<tr>
<td>982</td>
<td>20</td>
<td>Noémie Bossard</td>
<td>Multiprocessed Dumer4, Gregory Landais original implementation</td>
<td>2020-07-02</td>
<td></td>
</tr>
<tr>
<td>923</td>
<td>19</td>
<td>Valentin Vasseur</td>
<td>Dumer</td>
<td>2020-03-17</td>
<td></td>
</tr>
<tr>
<td>865</td>
<td>18</td>
<td>Valentin Vasseur</td>
<td>Dumer</td>
<td>2019-11-06</td>
<td></td>
</tr>
<tr>
<td>808</td>
<td>17</td>
<td>Valentin Vasseur</td>
<td>Dumer</td>
<td>2019-11-06</td>
<td></td>
</tr>
<tr>
<td>751</td>
<td>16</td>
<td>Valentin Vasseur</td>
<td>Dumer</td>
<td>2019-11-06</td>
<td></td>
</tr>
<tr>
<td>695</td>
<td>14</td>
<td>Valentin Vasseur</td>
<td>Dumer</td>
<td>2019-09-22</td>
<td></td>
</tr>
<tr>
<td>640</td>
<td>13</td>
<td>P. Loidreau</td>
<td>dumer4 by G. Landais</td>
<td>2019-09-15</td>
<td></td>
</tr>
<tr>
<td>587</td>
<td>12</td>
<td>P. Loidreau</td>
<td>dumer4 by G. Landais</td>
<td>2019-11-06</td>
<td></td>
</tr>
<tr>
<td>No.</td>
<td>Page</td>
<td>Name</td>
<td>Email</td>
<td>Date</td>
<td>See Details</td>
</tr>
<tr>
<td>------</td>
<td>------</td>
<td>---------------------------</td>
<td>------------------------</td>
<td>------------</td>
<td>-------------</td>
</tr>
<tr>
<td>534</td>
<td>11</td>
<td>Francesco Tinarelli</td>
<td></td>
<td>2019-08-26</td>
<td>See Details</td>
</tr>
<tr>
<td>482</td>
<td>11</td>
<td>Francesco Tinarelli</td>
<td></td>
<td>2019-08-24</td>
<td>See Details</td>
</tr>
<tr>
<td>431</td>
<td>10</td>
<td>Francesco Tinarelli</td>
<td></td>
<td>2019-08-24</td>
<td>See Details</td>
</tr>
<tr>
<td>381</td>
<td>9</td>
<td>Francesco Tinarelli</td>
<td></td>
<td>2019-08-24</td>
<td>See Details</td>
</tr>
<tr>
<td>333</td>
<td>8</td>
<td>Francesco Tinarelli</td>
<td></td>
<td>2019-08-23</td>
<td>See Details</td>
</tr>
<tr>
<td>286</td>
<td>7</td>
<td>Julien Lavauzelle</td>
<td>Lee-Brickell</td>
<td>2019-08-13</td>
<td>See Details</td>
</tr>
<tr>
<td>240</td>
<td>6</td>
<td>Julien Lavauzelle</td>
<td>Lee-Brickell</td>
<td>2019-08-13</td>
<td>See Details</td>
</tr>
<tr>
<td>197</td>
<td>5</td>
<td>Julien Lavauzelle</td>
<td>Lee-Brickell</td>
<td>2019-08-13</td>
<td>See Details</td>
</tr>
<tr>
<td>156</td>
<td>4</td>
<td>Julien Lavauzelle</td>
<td>Lee-Brickell</td>
<td>2019-08-13</td>
<td>See Details</td>
</tr>
<tr>
<td>117</td>
<td>4</td>
<td>Julien Lavauzelle</td>
<td>Lee-Brickell</td>
<td>2019-08-13</td>
<td>See Details</td>
</tr>
<tr>
<td>80</td>
<td>3</td>
<td>Julien Lavauzelle</td>
<td>Lee-Brickell</td>
<td>2019-08-13</td>
<td>See Details</td>
</tr>
<tr>
<td>48</td>
<td>1</td>
<td>Aleksei Udovenko</td>
<td></td>
<td>2019-08-20</td>
<td>See Details</td>
</tr>
<tr>
<td>20</td>
<td>1</td>
<td>Aleksei Udovenko</td>
<td></td>
<td>2019-08-20</td>
<td>See Details</td>
</tr>
</tbody>
</table>
## Syndrome Decoding in the Goppa-McEliece Setting

### Details on record 14

<table>
<thead>
<tr>
<th><strong>Length</strong></th>
<th>640</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Challenge provider</strong></td>
<td>0 - Inria Paris</td>
</tr>
<tr>
<td><strong>Weight</strong></td>
<td>13</td>
</tr>
<tr>
<td><strong>Vector</strong></td>
<td>00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000011 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000010 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000</td>
</tr>
<tr>
<td><strong>Authors</strong></td>
<td>P. Loidreau</td>
</tr>
<tr>
<td><strong>Algorithm</strong></td>
<td>dumer4 by G. Landais</td>
</tr>
<tr>
<td><strong>Hardware</strong></td>
<td>I7-2.3Ghz</td>
</tr>
<tr>
<td><strong>Runtime</strong></td>
<td>40s</td>
</tr>
<tr>
<td><strong>Comments</strong></td>
<td>Theoretical work factor : $2^{32}$ with parameters for Dumer $p=4$ and $l = 8$</td>
</tr>
<tr>
<td><strong>Date</strong></td>
<td>2019-09-15</td>
</tr>
</tbody>
</table>
Public Key Cryptanalysis, Part I (Codes)

Lecture 9: Advanced ISD

Alexander Ray, Ruhr-University Bochum
Leon's removal of $e_n$ ('88)

Leon's $e$-window: Use semi-systematic form. Q: How to compute?

Let $G \cdot P_2 = (\frac{H}{B}) e \times (k+e)$ and $s = (\frac{s_1}{s_2}) \cdot \frac{1}{3} n \cdot (n-1)$. Then

1. $e_n = s_1 + H \cdot e_2$
2. $O = s_2 + B \cdot e_2$ (we removed the annoying $e_1$)
Dumes-Stem ISD ('89)

Idea:

<table>
<thead>
<tr>
<th>$\omega - p$</th>
<th>$p/2$</th>
<th>$p/2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$e_1$</td>
<td>$e_2$</td>
<td>$e_3$</td>
</tr>
</tbody>
</table>

\[
\begin{pmatrix}
I_{n-k-e} & T_{e_1} & T_{e_2} \\
0 & B_1 & B_2 \\
n-k-e & e + \frac{e}{2} & e + \frac{e}{2}
\end{pmatrix} = \begin{pmatrix}
s_1 \\
s_2 \\
\end{pmatrix}
\]

Identities:

1. $T_{e_1} \cdot e_2 = e_4$, $S_4 + T_{e_2} \cdot e_3$
2. $B_1 \cdot e_2 = s_2 + B_2 \cdot e_3$

Strategy: First check (2), then (1).
Dumer-Stern ISD

Output: $P \in \mathbb{F}_q^{(n-k) \times n}, s \in \mathbb{F}_q^{n-k}, \omega = \omega(e), p < \omega, \ell \leq n-k$

1) Repeat until success
   1.1 Choose permutation $\pi \in \mathbb{F}_q^{n \times n}$.
   1.2 Compute semi-systematic form $GPH = (I_{n-k} \mid B_1 \mid B_2)$, $G \cdot S = (s_1)$. We assume this is doable, otherwise repeat.
   1.3 For all $e_2 \in \mathbb{F}_q^{k \times (n-k)}$: Compute $L$ with entries $(B_1 e_2, e_2)$
   1.4 For all $e_3 \in \mathbb{F}_q^{k \times (n-k)}$:
      1.4.1 For all $(s_2 + B_2 e_3, e_2) \in L$:
      If $w(P_1 e_2 + P_2 e_3 + s_1) = \omega - p$, then success. Does $(e_2, e_3)$ satisfy (1) also satisfy (2)?

2) Output: $e = H(P_1 e_2 + P_2 e_3 + s_1, e_2, e_3)$
Complexity: 

1. \[ \Pr \left[ H \text{ is good} \right] = \frac{\binom{n-k+\ell}{n}}{\binom{w-\ell}{w}} \left( \frac{\ell}{\binom{\ell}{\ell/2}} \right)^2 \]

2. \[ \left| \mathcal{F_2}^{k+\ell} \right| = \binom{k+\ell}{\ell/2} \]

3. \[ \mathcal{T} = \frac{\binom{n-k+\ell}{n}}{\binom{w-\ell}{w}} \left( \frac{\ell}{\binom{\ell}{\ell/2}} \right)^2 \cdot \max \left\{ \frac{1}{2}, \frac{(k+\ell)/2}{\ell/2} \cdot 2^{-e^2} \right\} \]

Again omitting \( \ell\), we have:

\[ \mathcal{M} = \binom{k+\ell}{\ell/2} \]

Eliece parameters: \( n = 3488, k = 2720, \omega = 64 \)

- For \( e = p = 0 \): \( \mathcal{T} = 2^{143}, \mathcal{M} = 2^{45} \) (no memory)
- Optimized \( p = 10, e = 46 \): \( \mathcal{T} = 2^{138} \), \( \mathcal{M} = 2^{45} \)

5 bit save for quite heavy memory.
May-Meurer-Thoma (MNT '12)

Ideas:

\[ \begin{align*}
    w-p & \quad e_1 \\
    e & \quad e_2 + e_3 \\
    P/2 & \quad P/2 \\
\end{align*} \]

\[ \begin{align*}
    n-k-e & \quad \{ n-k-e \} \\
    i & \quad A \\
    B & \quad B \\
\end{align*} \]

Identities:

(1) \( P_{e_2} \approx s_1 + P_{e_3} \) \hspace{1cm} \text{approximate}

(2) \( B_{e_2} = s_2 + B_{e_3} \) \hspace{1cm} \text{exact}

Strategy: First check (2), then (1).
Output: $e = H(\overline{A}(e_2 + e_3) + s_n, e_2 + e_3)$

1. Repeat until success
   1.1 Choose permutation $h \in \mathbb{F}_2^{n-k}$. Set $R = (ph_e)$.  
   1.2 Compute semi-systematic form $G \cdot Ph = (\overline{A}^t \mid \overline{B}_h)$, $G \cdot S = (\overline{s}_1)$.  
   1.3 For all $e_2 \in \mathbb{F}_2^k$ (-section 1): Compute $L_1 = \{(Be_2, e_2) \mid \overline{[Be_2]}_p = 0\}$  
   1.4 For all $e_3 \in \mathbb{F}_2^k$ (-section 2): Compute $L_2 = \{(s_2 + Be_3, e_3) \mid \overline{[Be_3]}_p = 0\}$  
   1.4.1 For all $(Be_2, e_2, s_2 + Be_3, e_3) \in L_1 \times L_2$ with $Be_2 = s_2 + Be_3$,  
      If $w(\overline{A}(e_2 + e_3) + s_n) = \omega - p$, then success.

2. Output: $e = H(\overline{A}(e_2 + e_3) + s_n, e_2 + e_3)$
Complexity: \[ \Pr \left[ H \text{ is good} \right] = \frac{(n-k-e) \cdot (e+e)}{(n)} \]

\[ \text{Pr} = \frac{(n-k-e) \cdot (e+e)}{(n)} \]

Exercise: Construct a MiniM for \( L_1, L_2 \), good enough for \( \text{McEliece} \) parameters.

\( n = 3488, k = 2720, \omega = 64 \)

Prange for \( e = p = 0 \): \( T = 2^{143}, \) no memory

Optimized \( p = 18, \epsilon = 87 \): \( T = 2^{483}, T = 2^{54} \)

Dumer-Stern \( T = 2^{438}, \) \( n = 2^{45} \)

Here we assume that \( L_1, L_2 \) can be constructed in time \( L_1, L_2 \).
<table>
<thead>
<tr>
<th>Page</th>
<th>Number</th>
<th>Name</th>
<th>Title</th>
<th>Date</th>
<th>See Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>534</td>
<td>11</td>
<td>Francesco Tinarelli</td>
<td></td>
<td>2019-08-26</td>
<td>See Details</td>
</tr>
<tr>
<td>482</td>
<td>11</td>
<td>Francesco Tinarelli</td>
<td></td>
<td>2019-08-24</td>
<td>See Details</td>
</tr>
<tr>
<td>431</td>
<td>10</td>
<td>Francesco Tinarelli</td>
<td></td>
<td>2019-08-24</td>
<td>See Details</td>
</tr>
<tr>
<td>381</td>
<td>9</td>
<td>Francesco Tinarelli</td>
<td></td>
<td>2019-08-24</td>
<td>See Details</td>
</tr>
<tr>
<td>333</td>
<td>8</td>
<td>Francesco Tinarelli</td>
<td></td>
<td>2019-08-23</td>
<td>See Details</td>
</tr>
<tr>
<td>286</td>
<td>7</td>
<td>Julien Lavauzelle</td>
<td>Lee-Brickell</td>
<td>2019-08-13</td>
<td>See Details</td>
</tr>
<tr>
<td>240</td>
<td>6</td>
<td>Julien Lavauzelle</td>
<td>Lee-Brickell</td>
<td>2019-08-13</td>
<td>See Details</td>
</tr>
<tr>
<td>197</td>
<td>5</td>
<td>Julien Lavauzelle</td>
<td>Lee-Brickell</td>
<td>2019-08-13</td>
<td>See Details</td>
</tr>
<tr>
<td>156</td>
<td>4</td>
<td>Julien Lavauzelle</td>
<td>Lee-Brickell</td>
<td>2019-08-13</td>
<td>See Details</td>
</tr>
<tr>
<td>117</td>
<td>4</td>
<td>Julien Lavauzelle</td>
<td>Lee-Brickell</td>
<td>2019-08-13</td>
<td>See Details</td>
</tr>
<tr>
<td>80</td>
<td>3</td>
<td>Julien Lavauzelle</td>
<td>Lee-Brickell</td>
<td>2019-08-13</td>
<td>See Details</td>
</tr>
<tr>
<td>48</td>
<td>1</td>
<td>Aleksei Udovenko</td>
<td></td>
<td>2019-08-20</td>
<td>See Details</td>
</tr>
<tr>
<td>20</td>
<td>1</td>
<td>Aleksei Udovenko</td>
<td></td>
<td>2019-08-20</td>
<td>See Details</td>
</tr>
</tbody>
</table>
## Syndrome Decoding in the Goppa-McEliece Setting

### Hall of Fame

<table>
<thead>
<tr>
<th>Length</th>
<th>Weight</th>
<th>Authors</th>
<th>Algorithm</th>
<th>Date</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>1347</td>
<td>25</td>
<td>Daniel J. Bernstein, Tanja Lange, Christiane Peters</td>
<td>See <a href="https://isd.mceliece.org/1347.html">https://isd.mceliece.org/1347.html</a> for more information.</td>
<td>2023-02-24</td>
<td>See details</td>
</tr>
<tr>
<td>1284</td>
<td>24</td>
<td>Andre Esser, Alex May and Floyd Zweydinger</td>
<td>MMT variant</td>
<td>2021-08-16</td>
<td>See details</td>
</tr>
<tr>
<td>1223</td>
<td>23</td>
<td>Andre Esser, Alex May and Floyd Zweydinger</td>
<td>BJMM/MMT variant</td>
<td>2021-05-10</td>
<td>See details</td>
</tr>
<tr>
<td>1161</td>
<td>22</td>
<td>Shintaro Narisada, Kazuhide Fukushima, and Shinsaku Kiyomoto</td>
<td>Dumer</td>
<td>2021-01-10</td>
<td>See details</td>
</tr>
<tr>
<td>1101</td>
<td>21</td>
<td>Anders Nilson</td>
<td>Multi threads Dumer4, Gregory Landais impl.</td>
<td>2020-08-14</td>
<td>See details</td>
</tr>
<tr>
<td>1041</td>
<td>19</td>
<td>Shintaro Narisada, Kazuhide Fukushima, and Shinsaku Kiyomoto</td>
<td>Dumer</td>
<td>2020-08-11</td>
<td>See details</td>
</tr>
<tr>
<td>982</td>
<td>20</td>
<td>Noémie Bossard</td>
<td>Multi threaded Dumer4, Gregory Landais original implementation</td>
<td>2020-07-02</td>
<td>See details</td>
</tr>
<tr>
<td>923</td>
<td>19</td>
<td>Valentin Vasseur</td>
<td>Dumer</td>
<td>2020-03-17</td>
<td>See details</td>
</tr>
<tr>
<td>865</td>
<td>18</td>
<td>Valentin Vasseur</td>
<td>Dumer</td>
<td>2019-11-06</td>
<td>See details</td>
</tr>
<tr>
<td>808</td>
<td>17</td>
<td>Valentin Vasseur</td>
<td>Dumer</td>
<td>2019-11-06</td>
<td>See details</td>
</tr>
<tr>
<td>751</td>
<td>16</td>
<td>Valentin Vasseur</td>
<td>Dumer</td>
<td>2019-11-06</td>
<td>See details</td>
</tr>
<tr>
<td>695</td>
<td>14</td>
<td>Valentin Vasseur</td>
<td>Dumer</td>
<td>2019-09-22</td>
<td>See details</td>
</tr>
<tr>
<td>640</td>
<td>13</td>
<td>P. Loidreau</td>
<td>dumer4 by G. Landais</td>
<td>2019-09-15</td>
<td>See details</td>
</tr>
<tr>
<td>587</td>
<td>12</td>
<td>P. Loidreau</td>
<td>dumer4 by G. Landais</td>
<td>2019-</td>
<td>See details</td>
</tr>
</tbody>
</table>
## Syndrome Decoding in the Goppa-McEliece Setting

### Details on record 14

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Length</strong></td>
<td>640</td>
</tr>
<tr>
<td><strong>Challenge provider</strong></td>
<td>0 - Inria Paris</td>
</tr>
<tr>
<td><strong>Weight</strong></td>
<td>13</td>
</tr>
</tbody>
</table>
| **Vector**         | 00000000000000000000000000000000000000000011
|                    | 000000000000000000000000000000000000000000 |  
|                    | 000000000000000000000000000000000000000000 |
|                    | 000000000000000000000000000000000000000000 |
|                    | 000000000000000000000000000000000000000000 |
|                    | 000000000000000000000000000000000000000000 |
|                    | 000000000000000000000000000000000000000000 |
|                    | 000000000000000000000000000000000000000000 |
|                    | 000000000000000000000000000000000000000000 |
|                    | 000000000000000000000000000000000000000000 |
|                    | 000000000000000000000000000000000000000000 |
|                    | 000000000000000000000000000000000000000000 |
|                    | 000000000000000000000000000000000000000000 |
|                    | 000000000000000000000000000000000000000000 |
|                    | 000000000000000000000000000000000000000000 |
|                    | 000000000000000000000000000000000000000000 |
|                    | 000000000000000000000000000000000000000000 |
|                    | 000000000000000000000000000000000000000000 |
|                    | 000000000000000000000000000000000000000000 |
|                    | 000000000000000000000000000000000000000000 |
|                    | 000000000000000000000000000000000000000000 |
|                    | 000000000000000000000000000000000000000000 |
|                    | 000000000000000000000000000000000000000000 |
| **Authors**        | P. Loidreau                                 |
| **Algorithm**      | dumer4 by G. Landais                        |
| **Hardware**       | i7-2.3Ghz                                  |
| **Runtime**        | 40s                                        |
| **Comments**       | Theoretical work factor : $2^{32}$ with parameters for Dumer $p=4$ and $l = 8$ |
| **Date**           | 2019-09-15                                 |
Public Key Cryptanalysis, Part I (Codes)

Lecture 10: McEliece

(using lecture notes of Elena Kirshanova)

Alexander May, Ruhr-University Bochum
Goppa Code

McEliece parameters: \( n = 3488, t = 64, m = 12 \)
\( t_m = 12 \cdot 64 = 768 = n - 2 \)

1. \( m \) defines the large field \( \mathbb{F}_{2^m} = \mathbb{F}_{2^n} \) with 4096 elements.
2. \( t \) defines the degree of the irreducible Goppa polynomial \( g(x) \in \mathbb{F}_{2^m} [x] \), i.e.,
   \[ g(x) = \sum_{i=0}^{t} g_i x^i, \quad g_i \in \mathbb{F}_{2^m}. \]
   \( t \) implies \( n \leq 2^m \).
3. \( n \) defines the number of distinct Goppa points \( L = \{ \alpha_1, \ldots, \alpha_n \} \in \mathbb{F}_{2^m} \).

Definition: A Goppa code \( C \) of length \( n \) is
\[
C(L, g) = \{ c \in \mathbb{F}_2^n : \sum_{i=1}^n c_i \frac{x_i}{x^i - \alpha_i} = 0 \mod g(x) \}
\]
elegant, but not suited for defining a parity check matrix

Exercise: Check that \( C(L, g) \) is a code, i.e., a subspace of \( \mathbb{F}_2^n \).
Towards a Parity Check Matrix cancel Recall: \( g(x) = g_0 + g_1 x + \ldots + g_t x^t \)

Observe that \( \frac{1}{x-\alpha_i} = - \frac{g(x)}{x-\alpha_i} \cdot g^{-1}(\alpha_i) \mod g(x) \). Let \( g(x) = \sum_{i=0}^t g_i x^i \):

\[
\frac{g(x)-g(\alpha_i)}{x-\alpha_i} = g_1(x-\alpha_i) + \ldots + g_t(x^t-\alpha_i^t)
\]

\( g(\alpha_i) = g_1(\alpha_i) + g_2(\alpha_i^2) + \ldots + g_t(\alpha_i^t) \)

Codeword \( c = \sum_{i=0}^n c_i \frac{g(x)-g(\alpha_i)}{x-\alpha_i} g^{-1}(\alpha_i) \) has coefficients Identity: \( P^t c = 0^t \)

\[
x^t-1: \sum_{i=0}^n c_i g_t g^{-1}(\alpha_i)
\]

\[
x^{t-2}: \sum_{i=0}^n c_i (g_{t-1}^2 + \alpha_i g_t) g^{-1}(\alpha_i)
\]

\[
x^0: \sum_{i=0}^n c_i (g_1 + \alpha_i g_2 + \alpha_i^2 g_3 + \ldots + \alpha_i^{t-1} g_t) \cdot g^{-1}(\alpha_i)
\]
Notice: \( L, g \) define \( \overline{P} \).

Secret Parity Check Matrix

\[
\overline{P} = \left( \begin{array}{cccc}
1 & 1 & \cdots & 1 \\
\alpha_1 & \alpha_2 & \cdots & \alpha_n \\
\vdots & \vdots & \ddots & \vdots \\
\alpha_1^{t-n} & \alpha_2^{t-n} & \cdots & \alpha_n^{t-n}
\end{array} \right) \cdot \left( \begin{array}{c}
g^t(\alpha_1) \\
g^t(\alpha_2) \\
\vdots \\
g^t(\alpha_n)
\end{array} \right) \in F_{2m}^{t \times n}
\]
McEliece Public Key

Let $F_2^m = F_2[y]/f(y)$ for some irreducible (over $F_2$) $f$ with $\deg(f) = m$. Then elements $\beta \in F_2^m$ can be written as $\beta = \beta_0 + \beta_1y + \ldots + \beta_{m-1}y^{m-1}$ with $\beta_i \in F_2$.

Def: Let $F_2^m = F_2[y]/f(y)$. Then we call the map

$$F_2^m \rightarrow F_2^m, \quad \beta = \beta_0 + \beta_1y + \ldots + \beta_{m-1}y^{m-1} \mapsto (\beta_0, \beta_1, \ldots, \beta_{m-1})$$

the canonical embedding of $F_2^m$ into $F_2^m$ (with respect to $f$).

Apply the canonical embedding component-wise on $\bar{P}$:

$$\bar{P} \in F_2^{tmn} \xrightarrow{\text{canonical embedding}} \bar{P} \in F_2^{tm \times n} \xrightarrow{\text{systematic form}} P = (I_m \mid \bar{A})$$

Transformations should hide secret $L, g$.
Towards distance

**Def:** For $y \in \mathbb{F}_2^n$ we define the Goppa syndrome of $y$ as $s_y(x) := \sum_{i=1}^{n} \frac{y_i}{x - \alpha_i} \mod g(x)$.

The support of $y$ is defined as $\text{supp}(y) = \{ i \in \{1, \ldots, n\} | y_i = 1 \}$, i.e., $y$'s 1-positions.

The Goppa multiplier of $y \in \mathbb{F}_2^n$ is defined as $f_x(y) := \prod_{i \in \text{supp}(y)} (x - \alpha_i)$.

**Corollary:** $c \in C(L,g) \iff \sum_{i=1}^{n} \frac{c_i}{x - \alpha_i} = 0 \mod g(x) \iff s_c(x) = 0$

**Lemma:** For $y \in \mathbb{F}_2^n$ we have $s'_y(x) = \sum_{i \in \text{supp}(y)} \prod_{j \in \text{supp}(y)} (x - \alpha_j)$.

**Proof:** Apply product formula of differentiation: $(uvw)' = uv'w + uvw' + u'vw$.
Lemma: $c \in C(L, g) \Rightarrow f'_c(x) = 0 \mod g(x)$

Proof: From the previous corollary we have $c \in C(L, g) \Rightarrow s_c(x) = 0$. Since $s_c(x) = \prod_{i \in \text{supp}(c)} (x - \alpha_i)$ and $g(x)$ is irreducible of $\deg(g) = t > 1$, we have $\gcd(f'_c(x), g(x)) = 1$. Moreover, $s_c(x) \cdot f(x) = \sum_{i=1}^{n} \frac{c_i}{x - \alpha_i} \cdot \prod_{i \in \text{supp}(c)} (x - \alpha_i) = \prod_{j \in \text{supp}(c)} (x - \alpha_j) = f'_c(x) \mod g(x)$. It follows that $s_c(x) = 0 \mod g(x) \Rightarrow f'_c(x) = 0 \mod g(x)$. \[\square\]
Lemma: \( C(L, g) = C(L, g^2) \)

\( \ast \): Let \( c \in C(L, g^2) \). Then \( s_c(x) = 0 \mod g^2(x) \)
\[ \Rightarrow s_c(x) - 0 \mod g(x) \Rightarrow c \in C(L, g) \]

\( \ast \): Let \( c \in C(L, g) \). Then \( f_c'(x) = 0 \mod g(x) \).

Let \( f_c'(x) = \sum_{i=1}^{n} i f_i(x^{i/2}) \). For even \( i \) we have \( i f_i(x^{i/2}) = 0 \mod 2 \).
\[ \Rightarrow f_c'(x) = \sum_{i=0 \mod 2}^{n} f_i(x^{i/2})^{2} = (\sum_{i=0 \mod 2} f_i(x^{i/2}))^{2} \]

Recall that \( a^2 + b^2 = (a + b)^2 \) over \( \mathbb{F}_2 \).

Therefore \( f_c'(x) \) is a square, implying that every irreducible factor of \( f_c'(x) \) has to appear with even multiplicity. Thus \( g^2(x) \mid f_c'(x) \)
\[ \Rightarrow f_c'(x) = 0 \mod g(x) \Rightarrow c \in C(L, g^2) \]
\[ \square \]
Goppa code distance

Theorem: Let \( (C,L,g) \) be a Goppa code with \( \deg(g) = t \). Then \( d(C) \geq 2t+1 \).

Proof: Let \( c \in C \setminus \Theta^n \) be a codeword of minimal weight \( w(c) = d(C) \). We have
\[
\sum_{i \in \text{supp}(c)} \prod_{j \in \text{supp}(c_i)} (x-x_i) = 0 \mod g^2(x).
\]
Recall \( C(1,g) = C(L,g^2) \).

\[
g^2(x) \mid f_c'(x) \Rightarrow \deg(f_c'(x)) = d(C) - 1 > \deg(g^2(x)) = 2t.
\]
\( \square \)
Public Key Cryptanalysis, Part I (Codes)

Lecture 11: Goppa Decoding

Alexander Ray, Ruhr-University Bochum
Decoding Goppa Codes

**Exercise:** Implement it.

**Theorem:** Let \( y = c \cdot e \) for some \( e \in C(L, g) \) with \( \deg_\mathbb{F}_q (e) \leq t \).

Then \( c \) can be computed efficiently.

**Proof:** We have \( s_y(x) = \sum_{i=1}^{\deg_\mathbb{F}_q (e)} \frac{y_i}{x - \alpha_i} = \sum_{i=1}^{\deg_\mathbb{F}_q (e)} \frac{c_i}{x - \alpha_i} + \sum_{i=1}^{\deg_\mathbb{F}_q (e)} \frac{e_i}{x - \alpha_i} = \sum_{i=1}^{\deg_\mathbb{F}_q (e)} \frac{1}{x - \alpha_i} \mod g^2(x). \)

Multiplication by (the unknown) \( f_e(x) = \prod_{i=1}^{\deg_\mathbb{F}_q (e)} (x - \alpha_i) \mod g^2(x) \) yields

\[
\left( \prod_{i=1}^{\deg_\mathbb{F}_q (e)} (x - \alpha_i^j) \right) \cdot s_y(x) = \sum_{i=1}^{\deg_\mathbb{F}_q (e)} \prod_{j \neq i} (x - \alpha_j^j) = f_e(x) \mod g^2(x).
\]

We have \( \deg(f_e) = t \) and \( \deg(f_e') = t - 1 \).

Solve the \( 2t \) equations in the \( 2t - 1 \) unknown coefficients of \( f_e(x) \) and \( f_e'(x) \).

Factor \( f_e(x) \) over \( \mathbb{F}_2 \cdot \mathbb{F}_q \cdot [x] \) in linear factors. Determine \( \supp(e) \) from \( \alpha_i \)'s.

Why only \( 2t - 1 \) unknowns?
Partial Key Exposure

Recall: McEliece secret key: \( L = \{ x_1, \ldots, x_n \} \), \( g(x) \in \mathbb{F}_2[x] \), \( \deg(g) = t \)

\( n = 3488 \), \( m = 12 \), \( t = 64 \)

public key: \( P = (I_m | A) \in \mathbb{F}_2^{tm \times n} \) \( tm = 768 \)

ciphertext: \( c = P \cdot e_m \) with \( w(e_m) = t \)

encoding of \( m \) from \( \mathbb{F}_2^n(t) \)

Question: Can we reconstruct \( (L, g) \) from partial information?

Motivation Partial Key Recovery attack: Obtain partial information from side channels.
Theorem: Given $P \in \mathbb{F}_2^{\times n}$ and $L = (\alpha_1, \alpha_2, \ldots, \alpha_n) \in \mathbb{F}_2^m$. Then $g(x)$ can be computed efficiently.

Proof: Compute a codeword $c \in \mathbb{F}_2^n$ with $P \cdot c = 0$. Then $g(x)$ can be computed efficiently.

Factor $f_c(x) \in \mathbb{F}_2[x]$ into irreducible factors. If there is a unique degree-$t$ factor, output $g(x)$. Otherwise, restart with a different codeword $c$. 

Exercise: Give an algorithm.
Secret Key Recovery from \( t+1 \) Goppa points

**Theorem:** Given \( P \in \mathbb{F}_2^{tm \times n} \), \( I = \{1, \ldots, n\} \), \( |I| > tm + 1 \), \( (\alpha_i)_{i \in I} \). Then \( g(x) \) can be computed efficiently.

**Proof:** Let \([P]_I \in \mathbb{F}_2^{tm \times |I|}\) denote the projection of \( P \) to the coords in \( I \).

Compute \( c' \in \mathbb{F}_2^{|I|} \) with \([P]_I \cdot c = 0\). Again, how?

Expand \( c' \) with zeros to \( c \in \mathbb{C}(L, g) \) having \( \text{supp}(c) \subseteq I \).

\[
\Rightarrow f_c'(x) = \sum_{i \in \text{supp}(c)} \prod_{j \in \text{supp}(c)} (x - \alpha_j) \quad \text{and} \quad g(x) | f_c'(x).
\]

Find \( g(x) \) from factoring \( f_c'(x) \) over \( \mathbb{F}_2^m [x] \).
| \((n, t, m)\)   | \(\ell = tm + 1\) | \(|\mathcal{L}| = 1\) | \(\ell = tm + 2\) | \(|\mathcal{L}| = 1\) | Av. time |
|-----------------|-------------------|----------------|-----------------|----------------|---------|
| \((3488, 64, 12)\) | 769               | 97%            | 770             | 100%           | 18 sec  |
| \((4608, 96, 13)\) | 1249              | 99%            | 1250            | 100%           | 54 sec  |
| \((6960, 119, 13)\) | 1548              | 99%            | 1549            | 100%           | 91 sec  |
| \((8192, 128, 13)\) | 1665              | 99%            | 1666            | 100%           | 105 sec |

**Table:** Recovery of Goppa polynomial \(g(x)\).
Recovery of Remaining Points

Exercise: $Ax=b$ is solvable iff $\text{rank}(A) = \text{rank}(AB)$.

Theorem: Given $P \in \mathbb{F}_2^{m \times n}$, $I = \{1, \ldots, n\}$, $1 \leq b \leq m + 1$, $(x_i)_{i \in I}$, $g(x) \in \mathbb{F}_2^m[x]$. Then $L = (x_1, \ldots, x_m)$ can be recovered efficiently.

Proof: Let us recover $x_r$ for some $r \in \{1, \ldots, n\} \setminus I$.

Assume for simplicity that $\text{rank}(P) = \text{rank}(P|_I) = \text{rank}(P|_{I \cup \{r\}}) = t_m$. Solve linear equation

$[P]_I \cdot c' = [P]_r$

Expand $c'$ with zeros to $c \in C(L, g)$ with $\text{supp}(c) \in I \cup \{r\}$.

$\Rightarrow \sum_{i \in \text{supp}(c)} \frac{1}{x - a_i} = \frac{1}{x - a_r} \mod g(x)$

Compute $(\sum_{i \in \text{supp}(c)} \frac{1}{x - a_i})^{-1} = x - a_r \mod g(x)$, read of $a_r$. $\blacksquare$
<table>
<thead>
<tr>
<th>$(n, t, m)$</th>
<th>$\ell = tm + 1$</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(3488, 64, 12)$</td>
<td>769</td>
<td>42 sec</td>
</tr>
<tr>
<td>$(4608, 96, 13)$</td>
<td>1249</td>
<td>130 sec</td>
</tr>
<tr>
<td>$(6960, 119, 13)$</td>
<td>1548</td>
<td>167 sec</td>
</tr>
<tr>
<td>$(8192, 128, 13)$</td>
<td>1665</td>
<td>183 sec</td>
</tr>
</tbody>
</table>

Notice: $2^{13}$

Table: Experimental results for point recovery.
Support Splitting Algorithm

Setting: We know all Goppa points, but not their order.

Example McEliece: \( n = 8192 = 2^{13} = 2^m \Rightarrow L = \mathbb{F}_2^m \)

Question: Assume that we know \( g(x) \) and \( L = \{x_1, \ldots, x_t\} \), can we order \( L \)?

Theorem (Sendrier’s Support Splitting): Let \( P \in \mathbb{F}_2^{(n-2)\times n} \) the parity check matrix of \( C \).

Let \( P' = S \cdot P \cdot T \) for some invertible \( S \in \mathbb{F}_2^{(n-2)\times (n-2)} \) and permutation \( P' \in \mathbb{F}_2^{n\times n} \).

Then one can (efficiently) find the permutation \( P' \in \mathbb{F}_2^{n\times n} \).

Proof omitted. Nice algorithm, but tricky analysis.
Idea of attack (knowing \( g(x) \)):

- Let \( \mathbb{F}_{2^m} = \{ \alpha_1, \alpha_2, \ldots, \alpha_m \} \).
- Construct matrix
  \[
  H = \begin{pmatrix}
  1 & 1 & & \\
  \alpha_1 & \alpha_2 & & \\
  \alpha_1^2 & \alpha_2^2 & & \\
  \vdots & \vdots & \ddots & \\
  \alpha_1^{m-1} & \alpha_2^{m-1} & & \alpha_m^{m-1}
  \end{pmatrix}
  \begin{pmatrix}
  g(\alpha_1) & 0 \\
  g(\alpha_2) & 0 \\
  \vdots & \vdots \\
  0 & g(\alpha_m)
  \end{pmatrix}
  \in \mathbb{F}_{2^m}^{tn}.
  \]
- Apply canonical embedding \( H \rightarrow P' \in \mathbb{F}_2^{tm \times n} \).
- Run Support Splitting on McEliece public key \( P \) and \( P' \) to find \( H \).
- Apply \( \overline{H} \) to recover \( L \).

Notice: Brute-Force on \( g(x) \in \mathbb{F}_{2^m}[x] \) costs \( \tilde{O}(2^{tm}) \) trials. \( tm = 128.13 \)
Public Key Cryptanalysis, Part I (Codes)

Lecture 12: Quantum

Alexander Ray, Ruhr-University Bochum
Recovery of Remaining Points

Exercise: $Ah \cdot x = b$ is solvable iff $\text{rank}(H) = \text{rank}(Hb)$.

Theorem: Given $P \in \mathbb{F}_2^{tm \times n}$, $I = \{i_1, \ldots, i_{tm}\}$, $L = \{i_1, \ldots, i_{tm}\}$, $(a_i)_{i \in I}$, $g(x) \in \mathbb{F}_2m[x]$. Then $L = (x_1, \ldots, x_n)$ can be recovered efficiently.

Proof: Let us recover $x_r$ for some $r \in \{1, \ldots, n \} \setminus I$.
Assume for simplicity that $\text{rank}([P]_I) = \text{rank}(P) = tm$. Solve linear equation $[P]_I \cdot c' = [P]_{r}$,

\[- \text{1-th column of } P \rightarrow \text{1-th column of } P\]

Expand $c'$ with zeroes to $c \in C(L, g)$ with $\text{supp}(c) \in I \cup \{r\}$.

\[\Rightarrow \sum_{i \in \text{supp}(c)} \frac{1}{x-a_i} - \frac{1}{x-x_r} \text{ mod } g(x)\]

Compute $\left(\sum_{i \in \text{supp}(c)} \frac{1}{x-a_i}\right)^{-1} = x-x_r \text{ mod } g(x)$, read of $x_r$. \[\square\]
<table>
<thead>
<tr>
<th>((n, t, m))</th>
<th>(\ell = tm + 1)</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>((3488, 64, 12))</td>
<td>769</td>
<td>42 sec</td>
</tr>
<tr>
<td>((4608, 96, 13))</td>
<td>1249</td>
<td>130 sec</td>
</tr>
<tr>
<td>((6960, 119, 13))</td>
<td>1548</td>
<td>167 sec</td>
</tr>
<tr>
<td>((8192, 128, 13))</td>
<td>1665</td>
<td>183 sec</td>
</tr>
</tbody>
</table>

**Notice:** \(2^{43}\)

Table: Experimental results for point recovery.
Support Splitting Algorithm

Setting: We know all Goppa points, but not their order.

Example McEliece: \( m = 8192 = 2^{13} = 2^m \Rightarrow L = \mathbb{F}_2^m \)

Question: Assume that we know \( g(x) \) and \( L = \{x_1, \ldots, x_n\} \), can we order \( L \)?

Theorem (Sendrier's Support Splitting): Let \( \overline{P} \in \mathbb{F}_2^{(n-2) \times n} \) be the parity check matrix of \( C \).
Let \( P' = S \cdot P \cdot T \) for some invertible \( S \in \mathbb{F}_2^{(n-2) \times (n-2)} \) and permutation \( P \in \mathbb{F}_2^n \).

Then one can (efficiently) find the permutation \( P \in \mathbb{F}_2^n \).

Proof omitted. Nice algorithm, but tricky analysis.
Idea of attack, knowing $g(x)$:

- Let $F_{2m} = \{x_1, x_2, \ldots, x_m\}$.
- Construct matrix
  
  $$
  H = \begin{pmatrix}
  1 & 1 & 1 \\
  x_1 & x_2 & x_3 \\
  \vdots & \vdots & \vdots \\
  x_1^{t-1} & x_2^{t-1} & x_3^{t-1}
  \end{pmatrix}
  \begin{pmatrix}
  g(x_1) & g(x_2) & 0 \\
  0 & 0 & g(x_m)
  \end{pmatrix} \in F_{2m}^{t \times n}.
  $$

- Apply canonical embedding $H \rightarrow \mathcal{P}' \in F_2^{tm \times n}$.
- Run Support Splitting on McEliece public key $\mathcal{P}$ and $\mathcal{P}'$ to find $\mathcal{H}$.
- Apply $\mathcal{H}$ to recover $L$.

Notice: Brute-Force on $g(x) \in F_{2m}[x]$ costs $\tilde{O}(2^{tm})$ trials. $tm = 128.13$
Grover Search (’96) / Amplitude Amplification (’97)

Let it be an algorithm with success probability $p$.

**Theorem (Classical):** On expectation we run $\frac{1}{p}$ instantiations of $T$ until (first) success.

**Proof:** Expectation $E[X] = \frac{1}{p}$ of geometric distribution with $\Pr[X=n] = (1-p)^{n-1} \cdot p$.

**Theorem (Quantum):** On expectation we run $\sqrt[p]{2}$ quantum instantiations of $T$ until success.

(without proof)

Typical square root speedup, can be shown to be optimal.
Prange with Amplitude Amplification

Prange's ISD is dominated by

$$\rho = \Pr[\bar{u} \text{ is good}] = \frac{(n-k)}{(n)}$$

McEliece: \( n = 3488, k = 2720, \omega = 64 \) \( \Rightarrow \)

$$T_{\text{classical}} = \frac{1}{\rho} \approx 2^{143}$$

$$T_{\text{quantum}} = \sqrt[16]{\frac{1}{\rho}} \approx 2^{32}$$

Note: Quantumly, McEliece has less than 80 bit security.
(But: Amplitude Amplification requires large quantum circuit depth.)
Advertisement for WS 23/24

1. Practical lab course: bochum-challenges.
   Add McEliece challenges.

2. Seminar: Finite fields / Cryptanalysis / Real World Crypto

Don't forget the evaluation: tinyurl.com/2a2wr7xs