Advances in Information-Set Decoding

Joint work with
Daniel J. Bernstein, Tanja Lange, and Henk van Tilborg

Christiane Peters

Technische Universiteit Eindhoven

EIDMA Cryptography Working Group, Utrecht
February 20, 2009

Motivation

$$\begin{tikzcd}
 & \begin{array}{c}\textbf{vanTilburg}\\ \boldsymbol{p=0}\end{array} \arrow[r] & \begin{array}{c}\textbf{vanTilburg}\\ \boldsymbol{p\geq 1}\end{array} & \textbf{CC} \\
\textbf{McEliece} \arrow[ur, "\leq \Theta(n)"] \arrow[r, "\Theta(n/\lg(n))"'] & \begin{array}{c}\textbf{LB}\\ \boldsymbol{p=1}\end{array} \arrow[ur, "\leq \Theta(n)"] \arrow[r, "\Theta(n/\lg(n))"'] & \begin{array}{c}\textbf{LB}\\ \boldsymbol{p\geq 2}\end{array} \arrow[u, "\leq \Theta(\lg(n))"'] \arrow[r] & \textbf{Stern} \arrow[u, "\leq 1+o(1)"']
\end{tikzcd}$$

1. Introduction
2. Information-set decoding
3. Asymptotic speedups since McEliece's original attack
4. Implications for code-based cryptography
5. A successfull attack on the original McEliece parameters

Linear codes

A binary $[n, k]$ code is a binary linear code of length n and dimension k, i.e., a k-dimensional subspace of $\mathbf{F}_2^n$.

A generator matrix of an $[n, k]$ code C is a $k \times n$ matrix G such that $C = \{\mathbf{x}\, G : \mathbf{x} \in \mathbf{F}_2^k\}$.

The matrix G corresponds to a map $\mathbf{F}_2^k \to \mathbf{F}_2^n$ sending a message $\mathbf{x}$ of length k to an n-bit string.

The Hamming distance between two words in $\mathbf{F}_2^n$ is the number of coordinates where they differ. The Hamming weight of a word is the number of non-zero coordinates.

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

Decoding problem

Consider binary linear codes with no obvious structure.

Classical decoding problem: find the closest codeword $\mathbf{x} \in C$ to a given $\mathbf{y} \in \mathbf{F}_2^n$, assuming that there is a unique closest codeword.

Berlekamp, McEliece, van Tilborg (1978) showed that the general decoding problem for linear codes is NP-complete.

Fixed-distance decoding

A fixed-distance-decoding algorithm searches for a codeword at a fixed distance from a received vector.

Inputs: the received vector $\mathbf{y}$ and a generator matrix G for the code.

Output: a sequence of weight-w elements $\mathbf{e} \in \mathbf{y} - \mathbf{F}_2^k G$.

Note that the output consists of error vectors $\mathbf{e}$, rather than codewords $\mathbf{y} - \mathbf{e}$.

In the important special case $\mathbf{y} = 0$, a fixed-distance-decoding algorithm searches for codewords of weight w.

Information sets

Given a generator matrix G of an $[n, k]$ code.

An information set is a size-k subset $I \subseteq \{1, 2, \ldots, n\}$ such that the I-indexed columns of G are invertible.

Denote the matrix formed by the I-indexed columns of G by G_I.
The I-indexed columns of $G_I^{-1}G$ are the $k \times k$ identity matrix.

Let $\mathbf{y} \in \mathbf{F}_2^n$ have distance w to a codeword in $\mathbf{F}_2^k G$, i.e.,
$\mathbf{y} = \mathbf{c} + \mathbf{e}$ for a codeword $\mathbf{c} \in \mathbf{F}_2^k G$ and a vector $\mathbf{e}$ of weight w.

Denote the I-indexed positions of $\mathbf{y}$ by $\mathbf{y}_I$.

If $\mathbf{y}_I$ is error-free, $\mathbf{y}_I G_I^{-1}$ is the original message and
$\mathbf{c} = (\mathbf{y}_I G_I^{-1})G$.

Example (1) – Setup

Assume we are a given a [8,4] code C by its generator matrix

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

Let $w = 2$. Take a codeword $\mathbf{c} = (0110)G = (11110111)$.

Let $\mathbf{y}$ be the received word $\mathbf{y} = \mathbf{c} + (00000011) = (11110100)$.

Find the error vector $\mathbf{e} = (00000011)$.

Example (2) – Lucky guess

Choose an information set $I = \{1, 2, 3, 5\}$.

We get $\mathbf{y}_I = (1110)$ from $\mathbf{y} = (11110100)$.

Compute the matrix $S = G_I^{-1}$ such that

$$SG = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}.$$

Since the error positions $\{7, 8\}$ and $I = \{1, 2, 3, 5\}$ are disjoint $\mathbf{y}_I S$ is the original message (0110) and thus $\mathbf{c} = (\mathbf{y}_I S)G$.

We find the error-vector $\mathbf{e} = \mathbf{y} - \mathbf{y}_I SG = (00000011)$.

Example (3) – one error among I-indexed columns of $\mathbf{y}$

What happens if an error occurred at a position indexed by I?

Let $\mathbf{c} = (0110)G$, $\mathbf{y} = \mathbf{c} + (10000001) = (01110110)$.
Again choose $I = \{1, 2, 3, 5\}$, and $\mathbf{y}_I = (0110)$.

The vector $\mathbf{y}_I S = (0111) \neq (0110)$ does not produce $\mathbf{c}$ and the output of the algorithm $\mathbf{y} - \mathbf{y}_I SG = (00010011)$ does not have weight 2 but 3.

"Repairing the damage":
Find the row G_a of SG corresponding to the error index $a \in I$.

For each $a \in I$ subtract the row G_a from $\mathbf{y} - \mathbf{y}_I SG$.

If $\mathbf{y} - \mathbf{y}_I SG - G_a$ has weight 2 it is the error pattern we are looking for.

The desired error vector is found by G_1:
$\mathbf{y} - \mathbf{y}_I SG - G_1 = (00010011) - (10010010) = (10000001)$.

The Lee–Brickell algorithm

Let $p \in \{0, 1, \ldots, w\}$.

The algorithm consists of a series of independent iterations. Each iteration has the following steps:

1. Select an information set $I \subseteq \{1, 2, \ldots, n\}$.
2. Find S and compute SG such that the I-indexed columns of SG are the $k \times k$ identity matrix.
3. Replace G by SG.
4. Use G to eliminate the I-indexed columns from $\mathbf{y}$, i.e., replace $\mathbf{y}$ by $\mathbf{y} - \mathbf{y}_I G$.
5. "Detecting p errors in I": For each size-p subset $A \subseteq \{1, \ldots, k\}$: Compute $\mathbf{e} = \mathbf{y} - \sum_{a \in A} G_a$, where G_a is the ath row of G; print $\mathbf{e}$ if it has weight w.

A weight-w error vector $\mathbf{e} \in \mathbf{y} - \mathbf{F}_2^k G$ is found by an information set I if and only if the I-indexed components of $\mathbf{e}$ have weight p, and the remaining components of $\mathbf{e}$ have weight $w - p$.

Information-set decoding algorithms

Error distribution among the columns of G.

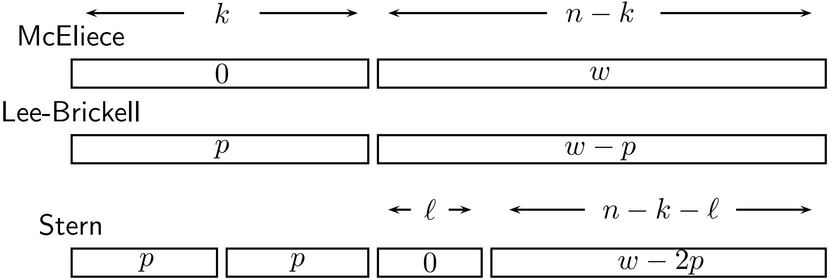


Figure from Overbeck and Sendrier: *Code-based Cryptography*, in *Post-Quantum Cryptography* (eds.: Bernstein, Buchmann, and Dahmen)

Stern's algorithm (1)

Let $p \in \{0, 1, \ldots, w\}$ and $\ell \in \{0, 1, \ldots, n-k\}$; $\ell \approx \lg \binom{k/2}{p}$.

Each iteration of this algorithm has the following steps:

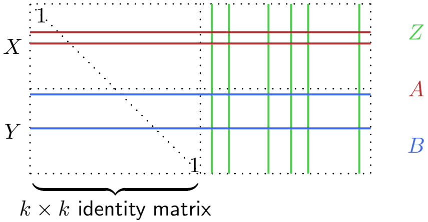
1. Select an information set $I \subset \{1, \ldots, n\}$.
2. Find S and compute SG such that the I-indexed columns of SG are the $k \times k$ identity matrix.
3. Replace G by SG.
4. Use G to eliminate the I-indexed columns from $\mathbf{y}$, i.e., replace $\mathbf{y}$ by $\mathbf{y} - \mathbf{y}_I G$.

Stern's algorithm (2)

"Detecting $2p$ errors in I":

5. Select a uniform random size-$\lfloor k/2 \rfloor$ subset $X \subseteq \{1, \ldots, k\}$;
6. Define $Y = \{1, \ldots, k\} \setminus X$.
7. Select a uniform random size-ℓ subset $Z \subseteq \{1, 2, \ldots, n\} \setminus I$.

Search for p rows G_a and p rows G_b such that $\mathbf{y} - \sum_{a \in A} G_a - \sum_{b \in B} G_b$ has weight w.



Consider only those sums of rows $\mathbf{y} - \sum G_a$, $\sum G_b$ which coincide on ℓ columns, i.e., those rows whose sum $\mathbf{y} - \sum G_a + \sum G_b$ has weight 0 on ℓ columns and obviously weight $2p$ on the information set.

Stern's algorithm (3)

8. For each size-p subset $A \subseteq X$: Compute $\varphi(A) \in \mathbf{F}_2^\ell$, the Z-indexed columns of $\mathbf{y} - \sum_{a \in A} G_a$.
9. For each size-p subset $B \subseteq Y$: Compute $\psi(B) \in \mathbf{F}_2^\ell$, the Z-indexed columns of $\sum_{b \in B} G_b$.
10. For each pair (A, B) such that $\varphi(A) = \psi(B)$: Compute $\mathbf{e} = \mathbf{y} - \sum_{a \in A} G_a - \sum_{b \in B} G_b$; print $\mathbf{e}$ if it has weight w.

A weight-w error vector $\mathbf{e} \in \mathbf{F}_2^k G + \mathbf{y}$ is found by an information set I along with X, Y, Z if and only if it has weight p in the part corresponding to X, weight p in the part corresponding to Y, and weight 0 in the part corresponding to Z.

Adaptive information sets

Choosing a uniform random matrix out of the set of $k \times k$ matrices over $\mathbf{F}_2$ provides a non-singular matrix with probability 0.2888.

The success probability of finding an information set among the n columns of the generator matrix of a binary linear $[n, k]$ code is highly biased by the code structure, and can be extremely small.

Workaround suggested by Stern:
instead of selecting k uniform random columns all at a time, choose k linearly independent columns of G adaptively, using each column for row reduction before choosing the next.

Motivation

Let R be the code rate and S the error fraction S.

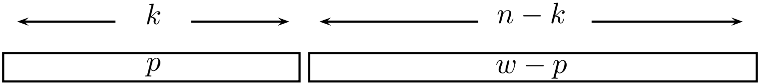
Goal: Measure the scalability of the information set algorithm.

The simplest form of information-set decoding takes time $2^{(\alpha(R,S)+o(1))n}$ to find Sn errors in a dimension-Rn length-n binary code if R and S are fixed while $n \to \infty$; here

$$\alpha(R,S) = (1-R-S)\lg(1-R-S) - (1-R)\lg(1-R) - (1-S)\lg(1-S)$$

and $\lg$ means the logarithm base 2.

Model of the number of iterations (Lee–Brickell)



If $\mathbf{e}$ is a uniform random weight-w element of $\mathbf{F}_2^n$, and I is a size-k subset of $\{1, \ldots, n\}$, then $\mathbf{e}$ has probability exactly

$$\mathrm{LBPr}(n, k, w, p) = \frac{\binom{n-k}{w-p}\binom{k}{p}}{\binom{n}{w}}$$

of having weight exactly p on I.

Consequently the Lee–Brickell algorithm, given $\mathbf{c} + \mathbf{e}$ as input for some codeword $\mathbf{c}$, has probability exactly $\mathrm{LBPr}(n, k, w, p)$ of printing $\mathbf{e}$ in the first iteration.

Note: These probabilities are averages over $\mathbf{e}$.

Model of the total cost (Lee–Brickell)

The function LBCost defined as

$$\mathrm{LBCost}(n, k, w, p) = \frac{\frac{1}{2}(n-k)^2(n+k) + \binom{k}{p}p(n-k)}{\mathrm{LBPr}(n, k, w, p)}.$$

is a model of the average time used by the Lee–Brickell algorithm.

- The term $\frac{1}{2}(n-k)^2(n+k)$ is a model of row-reduction time;
- $\binom{k}{p}$ is the number of size-p subsets A of $\{1, 2, \ldots, k\}$;
- and $p(n-k)$ is a model of the cost of computing $y - \sum_{a \in A} G_a$.

Note: Each vector G_a has n bits, but the k bits in columns corresponding to I can be skipped, since the sum in those columns is known to have weight p.

Stirling revisited

We assume that the code rate $R = k/n$ and error fraction $S = w/n$ satisfy $0 < S < 1 - R < 1$.

We put bounds on binomial coefficients as follows. Define $\epsilon(m)$ for each integer $m \geq 1$ by the formula

$$m! = \sqrt{2\pi}\ m^{m+1/2}\ e^{-m+\epsilon(m)}.$$

The classic Stirling approximation is $\epsilon(m) \approx 0$. Robbins showed that

$$\frac{1}{12m+1} < \epsilon(m) < \frac{1}{12m}.$$

Define $\mathrm{LBErr}(n, k, w, p)$ as

$$\frac{k!}{(k-p)!k^p} \frac{w!}{(w-p)!w^p} \frac{(n-k-w)!(n-k-w)^p}{(n-k-w+p)!} \frac{e^{\epsilon(n-k)+\epsilon(n-w)}}{e^{\epsilon(n-k-w)+\epsilon(n)}}.$$

Putting upper and lower bounds on $\mathrm{LBPr}(n, k, w, p)$

Define $\beta(R, S) = \sqrt{(1 - R - S)/((1 - R)(1 - S))}$.

Lemma

$\mathrm{LBPr}(n, k, w, p)$ *equals*

$$2^{-\alpha(R,S)n} \, \frac{1}{p!} \left(\frac{RSn}{1 - R - S} \right)^p \frac{1}{\beta(R, S)} \, \mathrm{LBErr}(n, k, w, p).$$

Furthermore

$$\frac{(1-\frac{p}{k})^p (1-\frac{p}{w})^p}{(1+\frac{p}{n-k-w})^p} e^{-\frac{1}{12n}(1 + \frac{1}{1-R-S})} < \mathrm{LBErr}(n, k, w, p) < e^{\frac{1}{12n}(\frac{1}{1-R} + \frac{1}{1-S})}.$$

Note that for fixed rate R, fixed error fraction S, and fixed p the error factor $\mathrm{LBErr}(n, nR, nS, p)$ is close to 1 as n tends to infinity.

Comparing Lee-Brickell for various p

Corollary

$\mathrm{LBCost}(n, Rn, Sn, 0) = (c_0 + O(1/n))2^{\alpha(R,S)n} n^3$ *as* $n \to \infty$
where $c_0 = (1/2)(1-R)(1-R^2)\beta(R,S)$.

Corollary

$\mathrm{LBCost}(n, Rn, Sn, 1) = (c_1 + O(1/n))2^{\alpha(R,S)n} n^2$ *as* $n \to \infty$
where $c_1 = (1/2)(1-R)(1-R^2)(1-R-S)(1/RS)\beta(R,S)$.

Corollary

$\mathrm{LBCost}(n, Rn, Sn, 2) = (c_2 + O(1/n))2^{\alpha(R,S)n} n$ *as* $n \to \infty$
where $c_2 = (1-R)(1+R^2)(1-R-S)^2(1/RS)^2\beta(R,S)$.

Corollary

$\mathrm{LBCost}(n, Rn, Sn, 3) = (c_3 + O(1/n))2^{\alpha(R,S)n} n$ *as* $n \to \infty$
where $c_3 = 3(1-R)(1-R-S)^3(1/S)^3\beta(R,S)$.

Decoding complexity comparison

$$
\begin{tikzcd}
 & \begin{matrix}\textbf{vanTilburg}\\ \boldsymbol{p=0}\end{matrix} \arrow[r] & \begin{matrix}\textbf{vanTilburg}\\ \boldsymbol{p\geq 1}\end{matrix} & \mathbf{CC} \\
\mathbf{McEliece} \arrow[ur, "\leq \Theta(n)"] \arrow[r, "\Theta(n/\lg(n))"'] & \begin{matrix}\textbf{LB}\\ \boldsymbol{p=1}\end{matrix} \arrow[ur, "\leq \Theta(n)"] \arrow[r, "\Theta(n/\lg(n))"'] & \begin{matrix}\textbf{LB}\\ \boldsymbol{p\geq 2}\end{matrix} \arrow[u, "\leq \Theta(\lg(n))"'] \arrow[r] & \textbf{Stern} \arrow[u, "\leq 1+o(1)"']
\end{tikzcd}
$$

- There are several variants of information-set decoding designed to reduce the cost of row reduction, sometimes at the expense of success probability.
- These variants save a non-constant factor for Lee–Brickell but save at most a factor $1 + o(1)$ for Stern. The critical point is that row reduction takes negligible time inside Stern's algorithm, since p is large.

McEliece PKC from an attacker's point of view

Given a $k \times n$ generator matrix G of a public code, and an error weight w.

To encrypt a message $\mathbf{m} \in \mathbf{F}_2^k$, the sender computes $\mathbf{m}G$, adds a random weight-w error vector $\mathbf{e}$, and sends $\mathbf{y} = \mathbf{m}G + \mathbf{e}$.

Not knowing the secret code and its decoding algorithm the attacker is faced with the problem of decoding $\mathbf{y}$ in a random-looking code.

McEliece proposed choosing random degree-t classical binary Goppa codes. The standard parameter choices are $k = n - t\lceil \lg n \rceil$ and $w = t$, typically with n a power of 2.

McEliece's original suggestion: $n = 1024$, $k = 524$, and $w = 50$.

Information-set decoding vs. McEliece

The standard choices $k = n - t\lceil \lg n \rceil$ and $w = t$ imply that the code rate $R = k/n$ and the error fraction $S = w/n$ are related by $S = (1 - R)/\lceil \lg n \rceil$.

For example, if $R = 1/2$, then $S = 1/(2\lceil \lg n \rceil)$. Consequently $S \to 0$ as $n \to \infty$.

Expand $\alpha(R, S)$ around $S = 0$. Consider the first two terms: For $R = 1/2$ and $S = 1/(2\lceil \lg n \rceil)$ we get $\alpha(1/2, 1/(2\lceil \lg n \rceil)) = ((1/2) + o(1))/\lg n$, so

$$\mathrm{LBCost}(n, (1/2)n, (1/2)n/\lceil \lg n \rceil, 0) = 2^{(1/2+o(1))n/\lg n}.$$

Taking more terms in the $\alpha(R, S)$ series gives a better approximation.

More careful analysis

For example, for McEliece's original $R = 524/1024$ and $S = 50/1024$, the leading factor $(1/(1-R))^w$ is $2^{51.71\ldots}$, and the next factor $e^{RSw/(2(1-R))}$ is $2^{1.84\ldots}$, with product $2^{53.55\ldots}$, while $\alpha(R,S) = 53.65\ldots$.

But: $\alpha(R,S)$ appears in all cost exponents.
Our lemma on LBCost allows much more precise comparisons between various decoding algorithms.

For example, increasing p from 0 to 2 saves a factor $(R^2(1-R^2)/(1+R^2)+o(1))n^2/(\lg n)^2$ in $\mathrm{LBCost}(n, Rn, (1-R)n/\lceil \lg n \rceil, p)$, and increasing p from 2 to 3 loses a factor $\Theta(\lg n)$.

1. Introduction

2. Information-set decoding

3. Asymptotic speedups since McEliece's original attack

4. Implications for code-based cryptography

5. A successfull attack on the original McEliece parameters

A successfull attack on the original McEliece parameters

Bernstein, Lange, P. (PQCrypto 2008):
Improved Stern's attack by

- Starting linear algebra part by using column selection from previous iteration.
- Forcing more existing pivots: *reuse exactly* $n-k-c$ *column selections (Canteaut et al.:* $c=1$*)*
- Faster pivoting
- Multiple choices of Z: *allow* m *disjoint sets* $Z_1,\dots,Z_m$ *s.t. the word we're looking for has weight* $p,p,0\dots,0$ *on the sets* $X,Y,Z_1,\dots,Z_m$
- Reusing additions of the ℓ-bit vectors for p-element subsets A of X
- Faster additions after collisions: *consider at most* w *instead of* $n-k$ *rows*

Running time in practice

Our attack software extracts a plaintext from a ciphertext by decoding 50 errors in a $[1024, 524]$ binary code.

Attack on a single computer with a 2.4GHz Intel Core 2 Quad Q6600 CPU would need, on average, approximately 1400 days (2^{58} CPU cycles) to complete the attack.

Running the software on 200 such computers would reduce the average time to one week.

Canteaut, Chabaud, and Sendrier: implementation on a 433MHz DEC Alpha CPU; one such computer would need approximately 7400000 days (2^{68} CPU cycles).

Note: Hardware improvements only reduce 7400000 days to 220000 days.

The remaining speedup factor of 150 comes from our improvements of the attack itself.

First successful attack

We were able to extract a plaintext from a ciphertext by decoding 50 errors in a $[1024, 524]$ binary code.

- there were about 200 computers involved, with about 300 cores
- computation finished in under 90 days
  (most of the cores put in far fewer than 90 days of work; some of which were considerably slower than a Core 2)
- used about 8000 core-days
- error vector found by Walton cluster at SFI/HEA Irish Centre of High-End Computing (ICHEC)
- the new parameters $m = 2$, $c = 12$ take only 5000 core-days on average

Thank you for your attention!