Advances in code-based public-key cryptography

D. J. Bernstein
University of Illinois at Chicago

Advertisements

1. pqcrypto.org:
Post-quantum cryptography—
hash-based, lattice-based,
code-based, multivariate quadratic
—introduction and bibliography.

2. pq.crypto.tw/pqc11/:
PQCrypto 2011, Taipei,
just before Asiacrypt.
Deadline **24 June 2011**.

3. 2011.indocrypt.org:
Indocrypt 2011, Chennai,
just after Asiacrypt.
Deadline **31 July 2011**.

The McEliece cryptosystem

(1978 McEliece)

McEliece public key:
linear map $G : \mathbf{F}_2^{524} \hookrightarrow \mathbf{F}_2^{1024}$
represented as 1024×524 matrix.

McEliece plaintext:
$m \in \mathbf{F}_2^{524}$;
and $e \in \mathbf{F}_2^{1024}$ of weight 50.

McEliece ciphertext:
$y = Gm + e \in \mathbf{F}_2^{1024}$.

Basic problem for attacker:
Given G, y, find codeword Gm
close to y in the code $G\mathbf{F}_2^{524}$.

Instead use parity-check matrix (1986 Niederreiter).

Niederreiter public key:
linear map $H : \mathbf{F}_2^{1024} \twoheadrightarrow \mathbf{F}_2^{500}$
represented as 500×1024 matrix.

Niederreiter plaintext:
$m \in \mathbf{F}_2^{1024}$ of weight 50.

Niederreiter ciphertext:
$s = Hm \in \mathbf{F}_2^{500}$.

Basic problem for attacker:
Given H, s, find low-weight $m \in \mathbf{F}_2^{1024}$ with $Hm = s$.
Equivalent to previous problem.

Information-set decoding

Choose random size-500 subset $S \subseteq \{1, 2, 3, \ldots, 1024\}$.

For almost all H:
Good chance
that $\mathbf{F}_2^S \hookrightarrow \mathbf{F}_2^{1024} \xrightarrow{H} \mathbf{F}_2^{500}$
is invertible.

Hope $m \in \mathbf{F}_2^S$; chance $\approx 2^{-53}$.
Apply inverse map to Hm,
revealing m if $m \in \mathbf{F}_2^S$.

If $m \notin \mathbf{F}_2^S$, try again.
Total cost $\approx 2^{80}$.

Long history, many improvements:
1962 Prange;
1981 Clark (crediting Omura);
1988 Lee–Brickell; 1988 Leon;
1989 Krouk; 1989 Stern;
1989 Dumer;
1990 Coffey–Goodman;
1990 van Tilburg; 1991 Dumer;
1991 Coffey–Goodman–Farrell;
1993 Chabanne–Courteau;
1993 Chabaud;
1994 van Tilburg;
1994 Canteaut–Chabanne;
1998 Canteaut–Chabaud;
1998 Canteaut–Sendrier.

1998 Canteaut–Chabaud–Sendrier: 2^{68} Alpha cycles to attack a McEliece ciphertext.

2008 Bernstein–Lange–Peters: further improvements; 2^{58} Core 2 Quad cycles to attack a McEliece ciphertext. Ran attack successfully!

Subsequent literature:
2009 Finiasz–Sendrier;
2010 Peters;
2011 Bernstein–Lange–Peters.

Higher security levels

Easily improve security
by scaling parameters up from
McEliece's $1024, 524, 50$ example.

Niederreiter public key:
linear map $H : \mathbf{F}_2^n \twoheadrightarrow \mathbf{F}_2^{n-k}$
represented as $(n-k) \times n$ matrix.

Niederreiter plaintext:
$m \in \mathbf{F}_2^n$ of weight w.

Niederreiter ciphertext:
$s = Hm \in \mathbf{F}_2^{n-k}$.

How large do n, k, w
have to be for 2^b security?

Basic information-set decoding:

Hope $m \in \mathbf{F}_2^S$.

Chance $\binom{n-k}{w} / \binom{n}{w}$.

Trying S costs $\approx n^3$.

Total cost $\approx n^3 \binom{n}{w} / \binom{n-k}{w}$.

Basic information-set decoding:

Hope $m \in \mathbf{F}_2^S$.

Chance $\binom{n-k}{w} / \binom{n}{w}$.

Trying S costs $\approx n^3$.

Total cost $\approx n^3 \binom{n}{w} / \binom{n-k}{w}$.

Standard entropy approximation:

If $w/n \to W$ as $n \to \infty$ then $\binom{n}{w}^{1/n} \to \frac{1}{W^W (1-W)^{1-W}}$.

Basic information-set decoding:
Hope $m \in \mathbf{F}_2^S$.
Chance $\binom{n-k}{w} / \binom{n}{w}$.

Trying S costs $\approx n^3$.
Total cost $\approx n^3 \binom{n}{w} / \binom{n-k}{w}$.

Standard entropy approximation:
If $w/n \to W$ as $n \to \infty$ then
$$\binom{n}{w}^{1/n} \to \frac{1}{W^W (1-W)^{1-W}}.$$

If furthermore $k/n \to R$ then
$$\binom{n-k}{w}^{1/n} \to \frac{(1-R)^{1-R}}{W^W (1-R-W)^{1-R-W}}.$$

So $\text{cost}^{1/n} \to \dfrac{(1-R-W)^{1-R-W}}{(1-R)^{1-R}(1-W)^{1-W}}$.

1988 Lee–Brickell idea:
Hope $m - e \in \mathbf{F}_2^S$ for
some weight-2 vector $e \in \mathbf{F}_2^{n-S}$.
Chance $\binom{n-k}{w-2}\binom{k}{2}/\binom{n}{w}$.

Trying S costs $\approx n^3$;
reuse one matrix inversion
for all choices of e.
Speedup $\approx k^2w^2/2(n-k-w)^2$.

1988 Lee–Brickell idea:
Hope $m - e \in \mathbf{F}_2^S$ for
some weight-2 vector $e \in \mathbf{F}_2^{n-S}$.
Chance $\binom{n-k}{w-2}\binom{k}{2}/\binom{n}{w}$.

Trying S costs $\approx n^3$;
reuse one matrix inversion
for all choices of e.
Speedup $\approx k^2w^2/2(n-k-w)^2$.

Not visible in cost$^{1/n}$ limit:
$$\text{cost}^{1/n} \to \frac{(1-R-W)^{1-R-W}}{(1-R)^{1-R}(1-W)^{1-W}}.$$
But still quite useful.

Many polynomial speedups
in subsequent papers.

e.g. 1988 Leon:
Choose random S as before;
invert $\mathbf{F}_2^S \hookrightarrow \mathbf{F}_2^n \xrightarrow{H} \mathbf{F}_2^{n-k}$;
choose size-ℓ subset $Z \subseteq S$.
Hope $m - e \in \mathbf{F}_2^{S-Z}$
for some weight-2 vector e.

Many polynomial speedups
in subsequent papers.

e.g. 1988 Leon:
Choose random S as before;
invert $\mathbf{F}_2^S \hookrightarrow \mathbf{F}_2^n \xrightarrow{H} \mathbf{F}_2^{n-k}$;
choose size-ℓ subset $Z \subseteq S$.
Hope $m - e \in \mathbf{F}_2^{S-Z}$
for some weight-2 vector e.

Advantage over Lee–Brickell:
quickly reject e if $\varphi(m - e) \neq 0$;
$\varphi : \mathbf{F}_2^n \to \mathbf{F}_2^Z$ is composition of
$\mathbf{F}_2^n \to \mathbf{F}_2^{n-k} \to \mathbf{F}_2^S \to \mathbf{F}_2^Z$.

Some loss of success chance
from disallowing $\mathbf{F}_2^Z$ in $m - e$.

Collision decoding (1989 Stern, independently 1989–1991 Dumer):

Again choose S, Z.
Partition $n - S$ into X, Y.
Hope $m - e - e' \in \mathbf{F}_2^{S-Z}$
for weight-p vectors e, e'
with $e \in \mathbf{F}_2^X$, $e' \in \mathbf{F}_2^Y$.

Collision decoding (1989 Stern, independently 1989–1991 Dumer):

Again choose S, Z.
Partition $n - S$ into X, Y.
Hope $m - e - e' \in \mathbf{F}_2^{S-Z}$
for weight-p vectors e, e'
with $e \in \mathbf{F}_2^X$, $e' \in \mathbf{F}_2^Y$.

Don't enumerate (e, e').
Make list of $\varphi(m - e)$;
make list of $\varphi(e')$;
find collisions between lists.

Collision decoding (1989 Stern, independently 1989–1991 Dumer):

Again choose S, Z.
Partition $n - S$ into X, Y.
Hope $m - e - e' \in \mathbf{F}_2^{S-Z}$
for weight-p vectors e, e'
with $e \in \mathbf{F}_2^X$, $e' \in \mathbf{F}_2^Y$.

Don't enumerate (e, e').
Make list of $\varphi(m - e)$;
make list of $\varphi(e')$;
find collisions between lists.

Optimal p is unbounded.
Exponential speedup for any (R, W), visible in $\text{cost}^{1/n}$ limit!

Ball-collision decoding (Bernstein–Lange–Peters, to appear at Crypto 2011):

Partition Z into A, B.
Hope $m - e - e' - f - f' \in \mathbf{F}_2^{S-Z}$
with $e \in \mathbf{F}_2^X$ of weight p,
$e' \in \mathbf{F}_2^Y$ of weight p,
$f \in \mathbf{F}_2^A$ of weight $\le q$,
$f' \in \mathbf{F}_2^B$ of weight $\le q$.

Expand $\varphi(m - e)$ into ball of radius q; similarly $\varphi(e')$; find collisions between balls.

Exponential speedup over Stern for any reasonable (R, W).

Decryption

How does legitimate receiver decrypt s (or y)?

Answer: Secretly generate a fast decoding algorithm D for a code $C(D)$.
Take random H (or G) with $C(D) = \operatorname{Ker} H$ (or $C(D) = G\mathbf{F}_2^k$).
Or systematic H: smaller, faster.

Fastest algorithms known to exploit McEliece's choice of D (by, e.g., computing D) are many orders of magnitude slower than collision decoding.

Fix a prime power q;
a positive integer m;
a positive integer $n \le q^m$;
distinct $a_1, \ldots, a_n \in \mathbf{F}_{q^m}$;
polynomial $g \in \mathbf{F}_{q^m}[x]$ with
$\deg g < n/m$ and
$g(a_1) \cdots g(a_n) \neq 0$.

The classical Goppa code
$\Gamma_q(a_1, \ldots, a_n, g)$
is the set of $c \in \mathbf{F}_q^n$ with
$\sum_i c_i/(x - a_i) = 0$ in $\mathbf{F}_{q^m}[x]/g$.

Code dimension $k \ge n - m \deg g$.
Almost always $k = n - m \deg g$.

McEliece's choice of $C(D)$:
$\Gamma_2(a_1, \ldots, a_n, g)$
with irreducible g of degree w.
Can you figure out $a_1, \ldots, a_n, g$
given $\Gamma_2(a_1, \ldots, a_n, g)$?

McEliece's choice of $C(D)$:
$\Gamma_2(a_1, \ldots, a_n, g)$
with irreducible g of degree w.
Can you figure out $a_1, \ldots, a_n, g$
given $\Gamma_2(a_1, \ldots, a_n, g)$?

McEliece's choice of D:
1975 Patterson algorithm
to decode $\deg g$ errors
given $a_1, \ldots, a_n, g$.

McEliece's choice of $C(D)$:
$\Gamma_2(a_1, \ldots, a_n, g)$
with irreducible g of degree w.
Can you figure out $a_1, \ldots, a_n, g$
given $\Gamma_2(a_1, \ldots, a_n, g)$?

McEliece's choice of D:
1975 Patterson algorithm
to decode $\deg g$ errors
given $a_1, \ldots, a_n, g$.

Original parameters: $m = 10$,
$w = 50$, $n = 1024$, $k = 524$.

McEliece's choice of $C(D)$:
$\Gamma_2(a_1, \ldots, a_n, g)$
with irreducible g of degree w.
Can you figure out $a_1, \ldots, a_n, g$
given $\Gamma_2(a_1, \ldots, a_n, g)$?

McEliece's choice of D:
1975 Patterson algorithm
to decode $\deg g$ errors
given $a_1, \ldots, a_n, g$.

Original parameters: $m = 10$,
$w = 50$, $n = 1024$, $k = 524$.

Much higher security: $m = 12$,
$w = 150$, $n = 3600$, $k = 1800$.

If $k/n \to R$ as $n \to \infty$
then $1 - m(\deg g)/n \to R$
but $m \geq (\lg n)/\lg q$
so $w/n = (\deg g)/n \to 0$.

Standard conjecture is that
decoding is still quite hard:
$(\text{constant}+o(1))^{n/\lg n}$ as $n \to \infty$.

McEliece reaches 2^b security
with $n \in b^{1+o(1)}$.
Encryption and decryption
cost only $b^{2+o(1)}$.

ECC also costs $b^{2+o(1)}$,
but ECC's $o(1)$ seems bigger
and ECC isn't post-quantum.

2008 Bernstein–Lange–Peters:

Why stop with $\deg g$ errors?
Can take w above $\deg g$.
Use fast list-decoding algorithms
for exactly the same codes.

List can have > 1 plaintext, but
standard "CCA2 conversions"
easily identify correct plaintext.

Each extra error makes
known attacks more difficult.
More security for same key size.
$\Rightarrow$ Smaller key for same security.

More codes

"I can increase w using
an asymptotically good code!
$k/n \to R > 0$ and
$w/n \to W > 0$."

More codes

"I can increase w using
an asymptotically good code!
$k/n \to R > 0$ and
$w/n \to W > 0$."

Maybe, but this isn't easy.
Do you also have a good D?
Does your D run quickly?
Are there many choices of D?
No exploitable structure in $C(D)$?
Is D actually better than Γ_2
for reasonable values of n?

Tempting to increase q.

$n/\sqrt{\lg q}, k/\sqrt{\lg q}, q$

have same key size as $n, k, 2$.

Maybe better security?

Tempting to increase q.
$n/\sqrt{\lg q}, k/\sqrt{\lg q}, q$
have same key size as $n, k, 2$.
Maybe better security?

Problem 1: Structural attacks
seem disastrous for large q.
e.g. 1992 Shestakov–Sidelnikov
broke 1986 Niederreiter proposal
using $\Gamma_q(\ldots)$ with $q \approx n$.

Tempting to increase q.
$n/\sqrt{\lg q}, k/\sqrt{\lg q}, q$
have same key size as $n, k, 2$.
Maybe better security?

Problem 1: Structural attacks seem disastrous for large q.
e.g. 1992 Shestakov–Sidelnikov broke 1986 Niederreiter proposal using $\Gamma_q(\ldots)$ with $q \approx n$.

Problem 2: Patterson's algorithm is specific to $q = 2$.
Conventional wisdom: correct only $(\deg g)/2$ errors for $q \geq 3$.

2010 Peters: switching from $q = 2$ to $q = 31$ gains factor 2 in key size with same security against information-set decoding, despite Problem 2.

2010 Peters: switching from $q = 2$ to $q = 31$ gains factor 2 in key size with same security against information-set decoding, despite Problem 2.

2010 Bernstein–Lange–Peters: "Wild Goppa codes" $\Gamma_q(\ldots, g^{q-1})$ with squarefree g correct $q(\deg g)/2$ errors, generalizing smoothly from $q = 2$. Even more with list decoding. Gain already for $q = 3$.

Ongoing work:
optimizing $\Gamma_q(\ldots, fg^{q-1})$.

Also many ongoing efforts
to reduce key size by creating
$C(D)$ with *visible* structure.
But safety is unclear.

e.g.
2010 Gauthier Umana–Leander
and 2010 Faugère–Otmani–
Perret–Tillich
broke most of the quasi-cyclic
and quasi-dyadic proposals
by 2009 Berger–Cayrel–Gaborit–
Otmani and 2009 Misocki–
Barreto.

List-decoding algorithms

Most often quoted results:

Take any alternant code over $\mathbf{F}_q$ of designed distance $t+1$. Assume $(n/t)q(\lg q^m) \in (\lg n)^{O(1)}$.

1999 Guruswami–Sudan: Polynomial-time algorithm for $w < n - \sqrt{n(n-t-1)}$. (Roughly: $w < t/2 + t^2/8n$.)

2000 Koetter–Vardy: Polynomial-time algorithm for $w < n' - \sqrt{n'(n'-t-1)}$ where $n' = n(q-1)/q$. (Roughly: $w < t/2 + t^2/8n + t^2/8n(q-1)$.)

What does this mean for Γ_q?

Easy application:
$\Gamma_q(\dots, g)$ is an alternant code
with designed distance $\deg g + 1$.
Slightly above $(\deg g)/2$ errors.

What does this mean for Γ_q?

Easy application:
$\Gamma_q(\ldots, g)$ is an alternant code with designed distance $\deg g + 1$.
Slightly above $(\deg g)/2$ errors.

2010 Bernstein–Lange–Peters:
Plug 1999 Guruswami–Sudan into 1975 Sugiyama–Kasahara–Hirasawa–Namekawa identity
$\Gamma_q(\ldots, g^{q-1}) = \Gamma_q(\ldots, g^q)$.

What does this mean for Γ_q?

Easy application:
$\Gamma_q(\ldots, g)$ is an alternant code with designed distance $\deg g + 1$.
Slightly above $(\deg g)/2$ errors.

2010 Bernstein–Lange–Peters:
Plug 1999 Guruswami–Sudan into 1975 Sugiyama–Kasahara–Hirasawa–Namekawa identity $\Gamma_q(\ldots, g^{q-1}) = \Gamma_q(\ldots, g^q)$.

2010 Augot–Barbier–Couvreur:
Plug 2000 Koetter–Vardy into 1975 Sugiyama–Kasahara–Hirasawa–Namekawa identity.

2011 Bernstein "Simplified high-speed high-distance list decoding for alternant codes":

Write $J' = n' - \sqrt{n'(n' - t - 1)}$.

$n^{O(1)}$ bit operations
if $w \leq J' + O((\lg n)/\lg\lg n)$.

2011 Bernstein "Simplified high-speed high-distance list decoding for alternant codes":

Write $J' = n' - \sqrt{n'(n' - t - 1)}$.

$n^{O(1)}$ bit operations
if $w \le J' + O((\lg n)/\lg\lg n)$.

$O(n^{4.5})$ bit operations
if $w \le J' + o((\lg n)/\lg\lg n)$.

2011 Bernstein "Simplified high-speed high-distance list decoding for alternant codes":

Write $J' = n' - \sqrt{n'(n' - t - 1)}$.

$n^{O(1)}$ bit operations
if $w \le J' + O((\lg n)/\lg\lg n)$.

$O(n^{4.5})$ bit operations
if $w \le J' + o((\lg n)/\lg\lg n)$.

$n(\lg n)^{O(1)}$ bit operations
if $w \le J' - n/(\lg n)^{O(1)}$.

Can of course combine with 1975 Sugiyama–Kasahara–Hirasawa–Namekawa identity.

Still not *really* fast.
Big problem for, e.g., $n = 3600$.

New wave of “rational” list-decoding algorithms promise much better speeds: 2007 Wu; 2008 Bernstein “List decoding for binary Goppa codes” (final version: IWCC 2011).

These algorithms are efficient only up to about J, not J'.
Can this limitation be removed?
I’m exploring one idea for this: “jet list decoding.”