Type-II optimal polynomial bases

D. J. Bernstein
University of Illinois at Chicago

Joint work with:
Tanja Lange
Technische Universiteit Eindhoven

Bigger project: “Breaking ECC2K-130”. Daniel V. Bailey, Lejla Batina, Daniel J. Bernstein, Peter Birkner, Joppe W. Bos, Hsieh-Chung Chen, Chen-Mou Cheng, Gauthier van Damme, Giacomo de Meulenaer, Luis Julian Dominguez Perez, Junfeng Fan, Tim Güneysu, Frank Gürkaynak, Thorsten Kleinjung, Tanja Lange, Nele Mentens, Ruben Niederhagen, Christof Paar, Francesco Regazzoni, Peter Schwabe, Leif Uhsadel, Anthony Van Herrewege, Bo-Yin Yang.

The target: ECC2K-130

1997: Certicom announces several elliptic-curve challenges.

"The Challenge is to compute the ECC private keys from the given list of ECC public keys and associated system parameters. This is the type of problem facing an adversary who wishes to completely defeat an elliptic curve cryptosystem."

Goals: help users select key sizes; compare random and Koblitz; compare $\mathbf{F}_{2^m}$ and $\mathbf{F}_p$; etc.

1997: ECCp-79 broken by Baisley and Harley.

1997: ECC2-79 broken by Harley et al.

1998: ECCp-89, ECC2-89 broken by Harley et al.

1998: ECCp-97 broken by Harley et al. (1288 computers).

1998: ECC2K-95 broken by Harley et al. (200 computers).

1999: ECC2-97 broken by Harley et al. (740 computers).

2000: ECC2K-108 broken by Harley et al. (9500 computers).

Certicom: "The 109-bit Level I challenges are feasible using a very large network of computers. The 131-bit Level I challenges are expected to be infeasible against realistic software and hardware attacks, unless of course, a new algorithm for the ECDLP is discovered."

2002: ECCp-109 broken by Monico et al. (10000 computers).

2004: ECC2-109 broken by Monico et al. (2600 computers).

Next challenge: ECC2K-130.

The attacker: ECRYPT

European Union has funded ECRYPT I network (2004–2008), ECRYPT II network (2008–2012).

ECRYPT II: KU Leuven; ENS; EPFL; RU Bochum; RHUL; TU Eindhoven; TU Graz; U Bristol; U Salerno; France Télécom; IBM Research; 22 adjoint members.

Work is handled by "virtual labs":

- SymLab: secret-key crypto;
- MAYA: public-key crypto;
- VAMPIRE: implementations.

Working groups in VAMPIRE:

- VAM1: "Efficient Implementation of Security Systems".
- VAM2: "Physical Security".

2009.02: VAMPIRE (VAM1)
sets its sights on ECC2K-130.
Exactly how difficult
is breaking ECC2K-130?
Also ECC2-131 etc.

Sensible topic for implementors.
Optimizing ECC attacks
isn't far from optimizing ECC.

With our latest implementations,
ECC2K-130 is breakable
in two years on average by

- 1595 Phenom II x4 955 CPUs,

With our latest implementations,
ECC2K-130 is breakable
in two years on average by

- 1595 Phenom II x4 955 CPUs,
- or 1231 Playstation 3s,

With our latest implementations,
ECC2K-130 is breakable
in two years on average by

- 1595 Phenom II x4 955 CPUs,
- or 1231 Playstation 3s,
- or 631 GTX 295 cards,

With our latest implementations,
ECC2K-130 is breakable
in two years on average by

- 1595 Phenom II x4 955 CPUs,
- or 1231 Playstation 3s,
- or 631 GTX 295 cards,
- or 308 XC3S5000 FPGAs,

With our latest implementations,
ECC2K-130 is breakable
in two years on average by

- 1595 Phenom II x4 955 CPUs,
- or 1231 Playstation 3s,
- or 631 GTX 295 cards,
- or 308 XC3S5000 FPGAs,
- or any combination thereof.

With our latest implementations,
ECC2K-130 is breakable
in two years on average by

- 1595 Phenom II x4 955 CPUs,
- or 1231 Playstation 3s,
- or 631 GTX 295 cards,
- or 308 XC3S5000 FPGAs,
- or any combination thereof.

This is a computation that
Certicom called “infeasible”?

With our latest implementations,
ECC2K-130 is breakable
in two years on average by

- 1595 Phenom II x4 955 CPUs,
- or 1231 Playstation 3s,
- or 631 GTX 295 cards,
- or 308 XC3S5000 FPGAs,
- or any combination thereof.

This is a computation that
Certicom called “infeasible”?

Certicom has now backpedaled,
saying that ECC2K-130
“may be within reach”.

ECC2K-130 optimization

"Breaking ECC2K-130" combines many levels of optimization.

This talk focuses on arithmetic in the finite field $\mathbf{F}_{2^{131}}$ inside ECC2K-130.

ECC2K-130 optimization

"Breaking ECC2K-130" combines many levels of optimization.

This talk focuses on arithmetic in the finite field $\mathbf{F}_{2^{131}}$ inside ECC2K-130.

Many implementation decisions, platform-specific optimizations.

This talk focuses on a simple but useful cost metric, namely # bit operations: i.e., #ANDs + #XORs.

Define $M(n)$ as minimum
bit operations for
multiplying n-bit polys.

e.g. $M(131) \leq 34061$ from
schoolbook multiplication:
131^2 ANDs + 130^2 XORs.

Define $M(n)$ as minimum
bit operations for
multiplying n-bit polys.

e.g. $M(131) \leq 34061$ from
schoolbook multiplication:
131^2 ANDs + 130^2 XORs.

Much lower costs are known.
Optimizations: Karatsuba
(not "Karatsuba–Ofman":
K–O paper credits Karatsuba);
refined Karatsuba; Toom; etc.

Current record (CRYPTO 2009):
$M(131) \leq 11961$.

“Your metric is too simple!
Hardware has area-time tradeoffs!
Software does not work on bits!”

"Your metric is too simple!
Hardware has area-time tradeoffs!
Software does not work on bits!"

Response: Optimizing
bit operations is very close to
optimizing the *throughput*
of unrolled, pipelined hardware.
See, e.g., ECC2K-130 FPGA
paper to appear at FPL 2010.

"Your metric is too simple!
Hardware has area-time tradeoffs!
Software does not work on bits!"

Response: Optimizing
bit operations is very close to
optimizing the *throughput*
of unrolled, pipelined hardware.
See, e.g., ECC2K-130 FPGA
paper to appear at FPL 2010.

Also very close to optimizing
the speed of software using
vectorized bit operations.
See, e.g., ECC2K-130 Cell
paper at AFRICACRYPT 2010.

All of the implementations of the ECC2K-130 attack started with the standard pentanomial basis $1, z, z^2, \ldots, z^{130}$ of $\mathbf{F}_{2^{131}} = \mathbf{F}_2[z]/(z^{131} + z^{13} + z^2 + z + 1)$.

Cost $M(131) + 455$
for multiplication.

Cost 203 for squaring.

Our final attack iteration has
5 mults, 21 squarings,
1 normal-basis Hamming weight.

Question at start of project:
Work entirely in normal basis?
Critical issue: mult speed.

Type-I normal basis of $\mathbf{F}_{2^n}$
is a permutation of
$\zeta, \zeta^2, \ldots, \zeta^n$
in $\mathbf{F}_2[\zeta]/(\zeta^n + \cdots + \zeta + 1)$.

Cost $M(n)$ to multiply,
obtaining coefficients of
$\zeta^2, \zeta^3, \ldots, \zeta^{2n}$.

Cost $2n - 2$ to reduce
$\zeta^2, \zeta^3, \ldots, \zeta^{2n}$
to $\zeta, \zeta^2, \ldots, \zeta^n$.

Alternative (1989 Itoh–Tsujii),
slightly faster when n is large:
redundant $1, \zeta, \ldots, \zeta^n$;
cost $M(n+1) + n$.

But $\mathbf{F}_{2^{131}}$ doesn't have
a type-I normal basis.

$\mathbf{F}_{2^{131}}$ has a type-II
normal basis $\zeta + \zeta^{-1}$,
$\zeta^2 + \zeta^{-2}$, $\zeta^4 + \zeta^{-4}$,
$\ldots$, $\zeta^{2^{130}} + \zeta^{-2^{130}}$ where
ζ is a primitive 263rd root of 1.

1995 Gao–von zur Gathen–
Panario: Can multiply on
type-II normal basis of $\mathbf{F}_{2^n}$
by multiplying in
$\mathbf{F}_2[\zeta]/(\zeta^{2n} + \cdots + 1)$.
Cost $> 2M(n)$.

2001 Bolotov–Gashkov:

Can quickly convert
from type-II normal basis
$c, c^2, c^4, \ldots, c^{2^{n-1}}$
to "standard basis"
$1, c, c^2, \ldots, c^{n-1}$
where $c = \zeta + \zeta^{-1}$.

Cost $\leq (n/2) \lg n + 3n$.
e.g. ≤ 853 for $n = 131$.

Same cost for inverse.

(Analysis is too pessimistic;
actual cost is lower.)

Bolotov–Gashkov multiply
in this "standard basis"
with a poly mult, cost $M(n)$,
and a reduction modulo
the minimal polynomial of c.

e.g. $c^{131} + c^{130} + c^{128} + c^{124} + c^{123} + c^{122} + c^{120} + c^{115} + c^{114} + c^{112} + c^{99} + c^{98} + c^{96} + c^{67} + c^{66} + c^{64} + c^3 + c^2 + 1 = 0$ for $n = 131$.

Bolotov–Gashkov reduction
uses sparsity; cost ≤ 2340.
Overall cost $\leq M(131) + 4899$
for type-II normal-basis mult.
Still too slow to be useful.

2007 Shokrollahi
(first published in Ph.D. thesis,
then in WAIFI 2007 paper
by von zur Gathen, Shokrollahi,
and Shokrollahi):

Convert from type-II normal basis
to redundant $1, c, c^2, \ldots, c^n$.

Multiply polynomials, producing
redundant $1, c, c^2, \ldots, c^{2n}$.

Convert to redundant
$1, \zeta + \zeta^{-1}, \zeta^2 + \zeta^{-2},$
$\zeta^3 + \zeta^{-3}, \ldots\ \zeta^{2n} + \zeta^{-2n}$.

Use $\zeta^{2n+1} = 1$
to eliminate redundancy.

For $n = 131$:

Shokrollahi's analysis says
$\leq M(132) + 3462$.

Our analysis of Shokrollahi's
algorithm says $M(132) + 1559$.

Easy speedup: $M(131) + 1559$.

5 mults, other iteration overhead:
$5M(131) + 12249$.

Compare to pentanomial basis:
$5M(131) + 14372$.

Do even better by mixing
permuted type-II optimal normal
basis $\zeta + \zeta^{-1}$, $\zeta^2 + \zeta^{-2}$,
$\zeta^3 + \zeta^{-3}$, ..., $\zeta^n + \zeta^{-n}$
with "type-II optimal polynomial
basis" $\zeta + \zeta^{-1}$, $(\zeta + \zeta^{-1})^2$,
$(\zeta + \zeta^{-1})^3$, ..., $(\zeta + \zeta^{-1})^n$.

Use normal basis for outputs
that will be provided to squaring,
poly basis for outputs
that will be provided to mult.

Use a new reduction algorithm
for poly-basis output.
See paper for details.

Current iteration cost:
$5M(131) + 10305$.

Practical impact:
All of the ECC2K-130
implementations have upgraded
from pentanomial basis
to type-II bases, saving time.

Ongoing project:
We are working on
fast high-security ECC software
using big Koblitz curves;
much bigger than ECC2K-130!