ECC2K-130 on NVIDIA GPUs

D. J. Bernstein (1)

Joint work with:
Hsieh-Chung Chen (2)
Chen-Mou Cheng (3)
Tanja Lange (4)
Ruben Niederhagen (3, 4)
Peter Schwabe (4)
Bo-Yin Yang (2)

1: U. Illinois at Chicago
2: Academia Sinica
3: National Taiwan U.
4: Technische U. Eindhoven

1997: Elliptic-curve discrete-log challenges issued by Certicom.

1997–2004: Harley, Monico, et al. solve first ten challenges: ECCp-79, ECC2-79, ECCp-89, ECC2-89, ECCp-97, ECC2K-95, ECC2-97, ECC2K-108, ECCp-109, ECC2-109.

Certicom: Subsequent challenges "are expected to be infeasible against realistic software and hardware attacks, unless of course, a new algorithm for the ECDLP is discovered."

Smallest challenge: ECC2K-130. Certicom "estimated number of machine days": 2700000000.

Smallest challenge: ECC2K-130.
Certicom "estimated number of machine days": 2700000000.

What I said at INDOCRYPT 2009 (joint work with 23 people):

ECC2K-130 is breakable
in a year on average
by 3039 3GHz Core 2 CPUs,
or by 2716 GTX 295 cards,
or by 2466 Cell CPUs,
or by 2026 XC3S5000 FPGAs,
or any combination of these.

I also said: “Hope to
finish attack in first half of 2010.”

I also said: “Hope to
finish attack in first half of 2010.”

But collecting hardware turned
out to be harder than expected.

I also said: "Hope to
finish attack in first half of 2010."

But collecting hardware turned
out to be harder than expected.

FPGAs are low cost

I also said: “Hope to
finish attack in first half of 2010.”

But collecting hardware turned
out to be harder than expected.

FPGAs are low cost but building
an FPGA cluster isn’t easy.

I also said: “Hope to
finish attack in first half of 2010.”

But collecting hardware turned
out to be harder than expected.

FPGAs are low cost but building
an FPGA cluster isn’t easy.

Owner of 214 Cell CPUs

I also said: “Hope to
finish attack in first half of 2010.”

But collecting hardware turned
out to be harder than expected.

FPGAs are low cost but building
an FPGA cluster isn’t easy.

Owner of 214 Cell CPUs
switched them to another project.

I also said: "Hope to
finish attack in first half of 2010."

But collecting hardware turned
out to be harder than expected.

FPGAs are low cost but building
an FPGA cluster isn't easy.

Owner of 214 Cell CPUs
switched them to another project.

"Open" US TeraGrid cluster

I also said: “Hope to
finish attack in first half of 2010.”

But collecting hardware turned
out to be harder than expected.

FPGAs are low cost but building
an FPGA cluster isn’t easy.

Owner of 214 Cell CPUs
switched them to another project.

“Open” US TeraGrid cluster
actually provides only token
amounts of time to math/CS.

Two strategies to make progress:

1. Keep collecting hardware through politics,

Two strategies to make progress:

1. Keep collecting hardware through politics, bribery,

Two strategies to make progress:

1. Keep collecting hardware through politics, bribery, viruses.

Two strategies to make progress:

1. Keep collecting hardware
by asking people nicely.

Two strategies to make progress:

1. Keep collecting hardware
by asking people nicely.
Have had some success here.

Two strategies to make progress:

1. Keep collecting hardware
by asking people nicely.
Have had some success here.

2. Make our attack even faster
so we need less hardware.

Two strategies to make progress:

1. Keep collecting hardware
by asking people nicely.
Have had some success here.

2. Make our attack even faster
so we need less hardware.

Today's focus:
NVIDIA GTX 295 graphics cards.

INDOCRYPT 2009 talk:
2716 card-years.

Two strategies to make progress:

1. Keep collecting hardware
by asking people nicely.
Have had some success here.

2. Make our attack even faster
so we need less hardware.

Today's focus:
NVIDIA GTX 295 graphics cards.

INDOCRYPT 2009 talk:
2716 card-years.

INDOCRYPT 2010 paper:
1068 card-years. 2.5× faster!

The ECC2K-130 iteration

ECC2K-130 team already chose a good iteration function.

Iteration input:
$(x, y) \in \mathbf{F}_{2^{131}} \times \mathbf{F}_{2^{131}}$
where $y^2 + xy = x^3 + 1$;
$\mathrm{wt}(N(x)) \in 2\mathbf{Z}$; and $x \neq 0$.
$N(x)$ means x in normal basis.

The ECC2K-130 iteration

ECC2K-130 team already chose a good iteration function.

Iteration input:
$(x, y) \in \mathbf{F}_{2^{131}} \times \mathbf{F}_{2^{131}}$
where $y^2 + xy = x^3 + 1$;
$\mathrm{wt}(N(x)) \in 2\mathbf{Z}$; and $x \neq 0$.
$N(x)$ means x in normal basis.

Iteration output: (x', y') where
$j = 3 + \left(\frac{1}{2} \mathrm{wt}(N(x)) \bmod 8\right)$;
$\lambda = (y + y^{2^j})/(x + x^{2^j})$;
$x' = \lambda^2 + \lambda + x + x^{2^j}$;
$y' = \lambda(x + x') + x' + y$.

Computations in iteration:

1 computation of $\mathrm{wt}(N(x))$;

7 additions in $\mathbf{F}_{2^{131}}$;

2 multiplications; 1 squaring;

2 computations of 2^j power;

1 inversion.

Fermat-type inversion costs

8 mults, many squarings.

Compute a batch of B iterations

to reduce cost to ≈ 3 mults:

use Montgomery's trick

$(\frac{1}{d}, \frac{1}{e}) = (e\frac{1}{de}, d\frac{1}{de})$ to

replace B independent inversions

with 1 inversion, $3B - 3$ mults.

2010 Bernstein–Lange "Type-II optimal polynomial bases":

Can compute this iteration
in only 70110 bit operations,
plus constant$/B$ overhead.

Combines
recent type-II mult ideas
(2007 Shokrollahi),
refined Karatsuba etc.
(2009 Bernstein),
dynamic normal/poly switch,
new reduction strategy.

GTX 295 without fans, case:

Overclocked Radeon 5970:

Why GPUs are interesting

NVIDIA GTX 295 graphics card
has two GPUs.
Each GPU has 30 cores
running at 1.242GHz.
(NVIDIA: "30 multiprocessors.")

Each core can perform
8 32-bit operations/cycle.
Total GTX 295 power:
480 32-bit ops/cycle.
(NVIDIA: "480 cores.")

$> 2^{39}$ 32-bit ops/second.
$> 2^{69}$ 1-bit ops/year.

Why GPUs are difficult

GPU core issues each instruction to many threads. Using full GPU power is difficult with < 192 threads, impossible with < 128 threads.

All data used by these threads must fit into core's SRAM: 65536 bytes of registers, 16384 bytes of shared memory.

Copying data from DRAM has huge latency, low throughput.

Trivial GPU parallelization

In 32 cycles a core can issue
an XOR instruction $c \leftarrow a \oplus b$
to 256 threads.

i.e. 256 XOR operations:
$c_0 \leftarrow a_0 \oplus b_0$, $c_1 \leftarrow a_1 \oplus b_1$, ...,
$c_{255} \leftarrow a_{255} \oplus b_{255}$.

i.e. 8192 bit operations:
$c_{0,0} \leftarrow a_{0,0} \oplus b_{0,0}$,
$c_{0,1} \leftarrow a_{0,1} \oplus b_{0,1}$,
...,
$c_{0,31} \leftarrow a_{0,31} \oplus b_{0,31}$,
...,
$c_{255,31} \leftarrow a_{255,31} \oplus b_{255,31}$.

Have 70110 bit operations
in one ECC2K-130 iteration:
XOR, XOR, AND, ...

Have $8192 \cdot 70110$ bit operations
in 8192 independent iterations:
XOR^{8192}, XOR^{8192}, AND^{8192}, ...

Sounds perfect for GPUs!

Core issues 70110 instructions
to 256 threads:
XOR, XOR, AND, ...
$32 \cdot 70110$ cycles.

The memory constraint

This trivial parallelization doesn't work for ECC2K-130.

Recall that the core has only 65536 bytes of registers.

8192 independent iterations
$\Rightarrow$ each iteration has
only 64 bits of registers.

But we need much more:
131 bits for x,
131 bits for y, ...

We actually process 64×
fewer iterations in parallel:

- 2× from having 128 threads
  instead of 256.
- 32× from having 32 threads
  working on 1 iteration.

64× more space per iteration:
4096 bits of registers,
1024 bits of shared memory.
Enough space for mult etc.

Main challenge: Find 32 parallel
bit operations in each operation.
See paper for details.

Results

Best speed with NVIDIA compiler:
$\approx$ 3000 cycles/iteration.

Gave up on compiler, built
a new GPU assembly language,
rewrote the software:
1379 cycles/iteration.

Current software:
1164 cycles/iteration.

Lower bound for arithmetic:
273 cycles/iteration.
Main slowdown: loads + stores.

News: anti-collisions

P walks to $P + R_i$.
$Q \neq P$ walks to $Q + R_j$.
$Q + R_j = P + R_i$? Not if $i = j$.

Standard attack-time formula,
including ECC2K-130 formula,
accounts for this slowdown.

2010 Bernstein–Lange:
Standard formula is wrong!
There is a further slowdown
from higher-order anti-collisions:
e.g. $P + R_i + R_k \neq Q + R_j + R_l$
if $R_i + R_k = R_j + R_l$.
$\approx 1\%$ slowdown for ECC2K-130.