Picture credit: Rick Bowmer/AP

How to use the new 65-megawatt Bluffdale supercomputer: a gentle introduction to cryptanalysis

D. J. Bernstein
University of Illinois at Chicago &
Technische Universiteit Eindhoven

Disclaimers

Disclaimers

1. I don't work for NSA.

Disclaimers

1. I don't work for NSA.

2. NSA hasn't told me anything.

Disclaimers

1. I don't work for NSA.
2. NSA hasn't told me anything.
3. This is not a leak.

Disclaimers

1. I don't work for NSA.

2. NSA hasn't told me anything.

3. This is not a leak.

4. I'm *assuming* that
NSA is not stupid.

Disclaimers

1. I don't work for NSA.

2. NSA hasn't told me anything.

3. This is not a leak.

4. I'm *assuming* that
NSA is not stupid.

5. Also *assuming* use of
traditional transistors+wires,
probably with some optics;
plus long-term storage.
Quantum computing would
require different analysis.

Cryptographic challenges

My mission: Cryptographically protect every Internet packet against espionage+sabotage.

Cryptographic challenges

My mission: Cryptographically protect every Internet packet against espionage+sabotage.

User needs crypto to be fast on devices designed primarily for doing something else:

User also needs
crypto to be **secure**.

Some examples of crypto failing:

- 2009 exploit of RSA-512 signatures in TI calculators (small public computation);
- 2010 exploit of ECDSA signatures in PlayStation 3 (trivial—stupid Sony mistake);
- 2012 exploit of MD5-based signatures by Flame malware (somewhat larger computation).

User also needs
crypto to be **secure**.

Some examples of crypto failing:

- 2009 exploit of RSA-512 signatures in TI calculators (small public computation);
- 2010 exploit of ECDSA signatures in PlayStation 3 (trivial—stupid Sony mistake);
- 2012 exploit of MD5-based signatures by Flame malware (somewhat larger computation).

Presumably many more examples not known to the public.

Critical questions:

Which cryptographic systems
fit the user's cost constraints?
$\Rightarrow$ optimize choice of
cryptosystem + **algorithm**
for each user device.

Critical questions:

Which cryptographic systems
fit the user's cost constraints?
$\Rightarrow$ optimize choice of
cryptosystem + **algorithm**
for each user device.

Which cryptographic systems
can be broken by attackers?
$\Rightarrow$ optimize choice of
attack algorithm + **device**
for each cryptosystem.

Critical questions:

Which cryptographic systems
fit the user's cost constraints?
$\Rightarrow$ optimize choice of
cryptosystem + **algorithm**
for each user device.

Which cryptographic systems
can be broken by attackers?
$\Rightarrow$ optimize choice of
attack algorithm + **device**
for each cryptosystem.

Heavy interactions between
high-level algorithms and
low-level computer architecture.

Theory vs. experiment

Predictions made by theoretical physicists are often disputed, sometimes wrong.

Common sources of error: underlying models of physics; calculations from those models.

Theory vs. experiment

Predictions made by theoretical physicists are often disputed, sometimes wrong.

Common sources of error:
underlying models of physics;
calculations from those models.

Experiments aren't perfect
but catch many errors;
resolve many disputes;
provide raw data
leading to new theories;
build more confidence than
theory alone can ever produce.

Is physics uniquely error-prone?
Of course not.

Every field of science:
theoreticians make predictions
regarding observable phenomena;
experimental scientists
measure those phenomena;
we compare the results.

Is physics uniquely error-prone?
Of course not.

Every field of science:
theoreticians make predictions
regarding observable phenomena;
experimental scientists
measure those phenomena;
we compare the results.

What if measurements are
too expensive to carry out?
Measurements start with
scaled-down experiments,
work up towards
the scale of interest.

Algorithm analysis is another error-prone field of science.

Theoreticians make predictions regarding algorithm performance. These predictions are often disputed, sometimes wrong.

Algorithm analysis is another error-prone field of science.

Theoreticians make predictions regarding algorithm performance. These predictions are often disputed, sometimes wrong.

Particularly error-prone: cryptanalytic extrapolations from an academic computation to a serious real-world attack.

Algorithm analysis is another error-prone field of science.

Theoreticians make predictions regarding algorithm performance. These predictions are often disputed, sometimes wrong.

Particularly error-prone: cryptanalytic extrapolations from an academic computation to a serious real-world attack.

We catch errors, resolve disputes by carrying out experiments: actually running these algorithms on the largest scale we can.

1980s security evaluation:
"QS" factorization algorithm
costs 2^{100} to break RSA-1024.

1980s security evaluation:
"QS" factorization algorithm
costs 2^{100} to break RSA-1024.

1990 Pollard: new "NFS".

1980s security evaluation:
"QS" factorization algorithm
costs 2^{100} to break RSA-1024.

1990 Pollard: new "NFS".

1991 Adleman: NFS
won't beat QS for RSA-1024.

1980s security evaluation:
"QS" factorization algorithm
costs 2^{100} to break RSA-1024.

1990 Pollard: new "NFS".

1991 Adleman: NFS
won't beat QS for RSA-1024.

Subsequent experiments $\Rightarrow$
NFS is much faster; maybe 2^{80}?

1980s security evaluation:
"QS" factorization algorithm
costs 2^{100} to break RSA-1024.

1990 Pollard: new "NFS".

1991 Adleman: NFS
won't beat QS for RSA-1024.

Subsequent experiments $\Rightarrow$
NFS is much faster; maybe 2^{80}?

Actual security of RSA-1024 is
still a matter of dispute: e.g.,
2009 Bos–Kaihara–Kleinjung–
Lenstra–Montgomery oppose
NIST's transition to RSA-2048.

The attacker's supercomputer

Enough theory+experiment
should reach consensus
on amount of computation
required to break a system.

But can the attacker perform
this amount of computation?

The attacker's supercomputer

Enough theory+experiment
should reach consensus
on amount of computation
required to break a system.

But can the attacker perform
this amount of computation?

Hypothesize attacker resources.
This talk: $2 billion, 65MW.
Alternative: millions of
compromised Internet computers.

The interesting part: analyze
optimal use of those resources.

Communication vs. arithmetic

Bill Dally, 2013.06.17:
"Communication takes
more energy than arithmetic".

Communication vs. arithmetic

Bill Dally, 2013.06.17:
"Communication takes
more energy than arithmetic".

Stephen S. Pawlowski,
2013.06.18: "The majority of
energy that we spend today
is on transferring data."

Communication vs. arithmetic

Bill Dally, 2013.06.17:
"Communication takes
more energy than arithmetic".

Stephen S. Pawlowski,
2013.06.18: "The majority of
energy that we spend today
is on transferring data."

Depends what you're doing!

Computations fundamentally vary
in amount of communication
(distance and volume)
and amount of arithmetic.

Some algorithms using n^2 data:

Square matrix-vector product:
n^2 arithmetic.

Some algorithms using n^2 data:

Square matrix-vector product:
n^2 arithmetic.

FFT for input size n^2:
$n^2 \lg n$ arithmetic.

Some algorithms using n^2 data:

Square matrix-vector product:
n^2 arithmetic.

FFT for input size n^2:
$n^2 \lg n$ arithmetic.

Matrix-matrix product:
typically n^3 arithmetic
without Strassen etc.

Some algorithms using n^2 data:

Square matrix-vector product:
n^2 arithmetic.

FFT for input size n^2:
$n^2 \lg n$ arithmetic.

Matrix-matrix product:
typically n^3 arithmetic
without Strassen etc.

Integrals in quantum chemistry,
many common iterations,
graph algorithms, etc.:
n^4 arithmetic, sometimes more.

Chip area $n^{2+\epsilon}$
is enough to store
all data for size-n^2 FFT.

Chip area $n^{2+\epsilon}$
is enough to store
all data for size-n^2 FFT.

Chip area $n^{2+\epsilon}$
is also enough for
n^2 parallel ALUs.

Chip area $n^{2+\epsilon}$
is enough to store
all data for size-n^2 FFT.

Chip area $n^{2+\epsilon}$
is also enough for
n^2 parallel ALUs.

FFT takes time n^{ϵ},
thanks to parallelism? No!
Routing the FFT data
occupies area $n^{2+\epsilon}$
for time $n^{1+\epsilon}$.

Chip area $n^{2+\epsilon}$
is enough to store
all data for size-n^2 FFT.

Chip area $n^{2+\epsilon}$
is also enough for
n^2 parallel ALUs.

FFT takes time n^ϵ,
thanks to parallelism? No!
Routing the FFT data
occupies area $n^{2+\epsilon}$
for time $n^{1+\epsilon}$.

1981 Brent–Kung: need $n^{1+\epsilon}$
even without wire delays.

Chip area $n^{2+\epsilon}$
is enough to store
several $n \times n$ matrices.

Routing matrix product
occupies area $n^{2+\epsilon}$
for time $n^{1+\epsilon}$.

Typical n^3 arithmetic
also occupies n^2 ALUs
for time $n^{1+\epsilon}$.

Closer look at ϵ:
the ALU cost dominates,
although not by much.

>90% of the cost
of typical supercomputers
is spent on communication;
<10% on ALUs.

Is Bluffdale built this way?

$>$90% of the cost
of typical supercomputers
is spent on communication;
$<$10% on ALUs.

Is Bluffdale built this way?
No; NSA is not stupid.

Doubling number of ALUs
would cost $<$10% extra.
Would $\approx$double performance
of matrix-matrix product
and heavier-arith computations.

NSA's computations have a mix
of heavy arith and heavy comm.

GPUs have many ALUs
but relatively little
communication capacity:
a few long wires to RAM.

Is Bluffdale built this way?

GPUs have many ALUs
but relatively little
communication capacity:
a few long wires to RAM.

Is Bluffdale built this way?
No; NSA is not stupid.

Adding communication
between adjacent ALUs
would cost very little.
Would drastically speed up
matrix-matrix product
and heavier-comm computations:
FFT, sorting, etc.

Documentation tells me that Intel Xeon Phi has many ALUs and a few long wires to RAM **plus** adjacent one-dimensional communication (ring bus).

Is Bluffdale built this way?

Documentation tells me that
Intel Xeon Phi has many ALUs
and a few long wires to RAM
plus adjacent one-dimensional
communication (ring bus).

Is Bluffdale built this way?
No; NSA is not stupid.

Adding **two-dimensional grid**
would drastically speed up
heavy-comm computations.
e.g. 1977 Thompson–Kung.

Grid examples: MasPar; FPGAs.
But FPGAs have other problems.

Save even more time
with 3D arrangement of ALUs?
e.g. 1983 Rosenberg.

Huge engineering challenge.

2D allows easy scaling of
energy input, heat output
up to very large chip area.
3D is hard to scale.

Some limited progress
(most interesting: optics),
presumably used by NSA.
Progress often exaggerated:
e.g., $4 \times 16384 \times 16384$
is often called "3D".

Special vs. general purpose

Typical cryptanalytic arith:
between 100× and 1000×
better performance per transistor
from ASICs than from
mass-market CPUs, GPUs.

Some exceptions, but overall
ASICs bring massive speedup.

Special vs. general purpose

Typical cryptanalytic arith:
between 100× and 1000×
better performance per transistor
from ASICs than from
mass-market CPUs, GPUs.

Some exceptions, but overall
ASICs bring massive speedup.

Only in cryptanalysis? No.
Estimated ASIC improvement
from preliminary scan of other
supercomputing arith problems:
usually $>10\times$, often $>100\times$.

Frequent observation:
chips spend area, time, energy
on decoding+scheduling insns.

$\Rightarrow$ CPU/GPU design trend:
reduce insn-handling cost
by adding vectorization—
apply same instruction
to multiple data/threads.

Frequent observation:
chips spend area, time, energy
on decoding+scheduling insns.

$\Rightarrow$ CPU/GPU design trend:
reduce insn-handling cost
by adding vectorization—
apply same instruction
to multiple data/threads.

But this does nothing
to reduce costs of
reading data from reg file,
writing data to reg file.

Obvious strategy to
reduce these reg costs:
combine arith operations,
doing more arith
between read and write.

Example: Build circuit
to compute $xy + z$.
CPU reads regs x, y, z;
computes $xy + z$; writes.

With separate mul, add:
CPU reads x, y; computes xy;
writes; reads back; reads z;
computes $xy + z$; writes.

Common fp operations evolved in this way.

Chip designer saw many single-precision fp muls, eventually spent area on circuit for those muls.

Common fp operations
evolved in this way.

Chip designer saw many
single-precision fp muls,
eventually spent area on
circuit for those muls.

Then spent much more area
to expand the multiplier
to double-precision fp.

Common fp operations
evolved in this way.

Chip designer saw many
single-precision fp muls,
eventually spent area on
circuit for those muls.

Then spent much more area
to expand the multiplier
to double-precision fp.

But people still run many
single-precision computations.
The multiplier transistors are
mostly sitting idle.

Another example:
Your application does
mul-sub-sub-sub-sub
in its inner loop.

Should CPU designer
include mul circuit,
4 separate sub circuits?

Another example:
Your application does
mul-sub-sub-sub-sub
in its inner loop.

Should CPU designer
include mul circuit,
4 separate sub circuits?

Same CPU then runs
another application.
Subtraction circuits are
mostly sitting idle.

Another example:
Your application does
mul-sub-sub-sub-sub
in its inner loop.

Should CPU designer
include mul circuit,
4 separate sub circuits?

Same CPU then runs
another application.
Subtraction circuits are
mostly sitting idle.

CPU designer says no,
reduces area per core.
$\Rightarrow$ Your application runs slowly.

Many ASIC fp speedups
beyond today's CPUs/GPUs:

- Squaring is cheaper
  than multiplication.
- Skip most normalizations.
- Reduce precision to
  what is actually needed.
- Add very fast sqrt
  if application needs it.
- etc.

Many ASIC fp speedups
beyond today's CPUs/GPUs:

- Squaring is cheaper
  than multiplication.
- Skip most normalizations.
- Reduce precision to
  what is actually needed.
- Add very fast sqrt
  if application needs it.
- etc.

Cryptanalysis involves
many multiplications
but also a much wider
variety of operations.
Even larger ASIC speedups.

So NSA builds ASICs
for each application?

The small problem:
ASIC design effort.
Not a serious issue
for $2 billion.

So NSA builds ASICs
for each application?

The small problem:
ASIC design effort.
Not a serious issue
for $2 billion.

The big problem:
Unpredictable application mix.
NSA will want some agility
to adapt to new computations
and stop old computations.

Quantify using historical data:
how long is an ASIC useful?

Obvious solution for NSA:
some ASICs, plus heterogeneous
mix of **application-tuned**
integrated circuits (ATICs).

Take a general-purpose CPU.
Add exactly the big insn
XYZZY needed by application,
plus some vectorization.
Think ahead, add agility:
XYZZ? XZZY? XYQZZY?
Still similar cost to ASIC.

New CPU for each application.
Merge similar applications
if not much cost in area.

1-slide Bluffdale user guide

Critical for algorithm designer and implementor:

1-slide Bluffdale user guide

Critical for algorithm designer and implementor:

Massive parallelism.

1-slide Bluffdale user guide

Critical for algorithm designer and implementor:

Massive parallelism.

Grid communication.

1-slide Bluffdale user guide

Critical for algorithm designer and implementor:

Massive parallelism.

Grid communication.

Multiple instruction sets with very useful instructions.

1-slide Bluffdale user guide

Critical for algorithm designer and implementor:

Massive parallelism.

Grid communication.

Multiple instruction sets with very useful instructions.

Some vectorization.

1-slide Bluffdale user guide

Critical for algorithm designer and implementor:

Massive parallelism.

Grid communication.

Multiple instruction sets with very useful instructions.

Some vectorization.

Occasional faults.

1-slide Bluffdale user guide

Critical for algorithm designer and implementor:

Massive parallelism.

Grid communication.

Multiple instruction sets with very useful instructions.

Some vectorization.

Occasional faults.

Need to understand cryptanalysis: ECM, sparse linear algebra, differentials, FFTs, much more.