

Batch NFS

D. J. Bernstein

University of Illinois at Chicago &
Technische Universiteit Eindhoven

Tanja Lange

Technische Universiteit Eindhoven

In this talk $\log L$ means

$$(1 + o(1))(\log N)^{1/3}(\log \log N)^{2/3}.$$

L is often written

$$“L_N(1/3)” \text{ or } “L_N(1/3)^{1+o(1)}”.$$

Exponents of L in this talk

are limited to $10^{-6}\mathbf{Z}$.

Rigorously proven? Ha ha ha.

2003 Shamir–Tromer, 2003
Lenstra–Tromer–Shamir–
Kortsmit–Dodson–Hughes–
Leyland, 2005 Geiselmann–
Shamir–Steinwandt–Tromer, 2005
Franke–Kleijnung–Paar–Pelzl–
Priplata–Stahlke, etc.: RSA-1024
is breakable in a year by an attack
machine costing $<10^9$ dollars.

2003 Shamir–Tromer, 2003
Lenstra–Tromer–Shamir–
Kortsmit–Dodson–Hughes–
Leyland, 2005 Geiselman–
Shamir–Steinwandt–Tromer, 2005
Franke–Kleinjung–Paar–Pelzl–
Priplata–Stahlke, etc.: RSA-1024
is breakable in a year by an attack
machine costing $<10^9$ dollars.

So the Internet switched to
RSA-2048, and we no longer care
about RSA-1024 security, right?

2003 Shamir–Tromer, 2003
Lenstra–Tromer–Shamir–
Kortsmitt–Dodson–Hughes–
Leyland, 2005 Geiselman–
Shamir–Steinwandt–Tromer, 2005
Franke–Kleijnung–Paar–Pelzl–
Priplata–Stahlke, etc.: RSA-1024
is breakable in a year by an attack
machine costing $<10^9$ dollars.

So the Internet switched to
RSA-2048, and we no longer care
about RSA-1024 security, right?

Wrong!

Example: The IP address of
dnssec-deployment.org
is signed by an RSA-1024 key

Example: The IP address of
dnssec-deployment.org
is signed by an RSA-1024 key
signed by an RSA-2048 key

Example: The IP address of
dnssec-deployment.org
is signed by an RSA-1024 key
signed by an RSA-2048 key
signed by org's RSA-1024 key

Example: The IP address of
dnssec-deployment.org
is signed by an RSA-1024 key
signed by an RSA-2048 key
signed by org's RSA-1024 key
signed by an RSA-2048 key

Example: The IP address of
dnssec-deployment.org
is signed by an RSA-1024 key
signed by an RSA-2048 key
signed by org's RSA-1024 key
signed by an RSA-2048 key
signed by a root RSA-1024 key

Example: The IP address of `dnssec-deployment.org` is signed by an RSA-1024 key signed by an RSA-2048 key signed by org's RSA-1024 key signed by an RSA-2048 key signed by a root RSA-1024 key signed by an RSA-2048 key.

Example: The IP address of `dnssec-deployment.org` is signed by an RSA-1024 key signed by an RSA-2048 key signed by org's RSA-1024 key signed by an RSA-2048 key signed by a root RSA-1024 key signed by an RSA-2048 key.

Most “DNSSEC” signatures follow a similar pattern.

Example: The IP address of `dnssec-deployment.org` is signed by an RSA-1024 key signed by an RSA-2048 key signed by org's RSA-1024 key signed by an RSA-2048 key signed by a root RSA-1024 key signed by an RSA-2048 key.

Most “DNSSEC” signatures follow a similar pattern.

Another example: SSL has used many millions of RSA-1024 keys. Imagine that an attacker has recorded tons of SSL traffic.

Users seem unconcerned:

1. “The attack machine costs more than this RSA key is worth.”
2. “The attack machine isn’t off-the-shelf; it’s only for attackers building ASICs.”
3. For signatures: “We switch keys every month, and the attack machine takes a year.”

Users seem unconcerned:

1. “The attack machine costs more than this RSA key is worth.”
2. “The attack machine isn’t off-the-shelf; it’s only for attackers building ASICs.”
3. For signatures: “We switch keys every month, and the attack machine takes a year.”

Real quote: “DNSSEC signing keys should be large enough to avoid all known cryptographic attacks during the effectivity period of the key.”

Continuation of quote: “To date, despite huge efforts, no one has broken a regular 1024-bit key; in fact, the best completed attack is estimated to be the equivalent of a 700-bit key. An attacker breaking a 1024-bit signing key would need to expend phenomenal amounts of networked computing power in a way that would not be detected in order to break a single key. Because of this, it is estimated that most zones can safely use 1024-bit keys for at least the next ten years.”

Goal of our “Batch NFS” paper:
analyze the *asymptotic* cost,
specifically *price-performance*
ratio, of breaking *many* RSA keys.

“Many”: e.g. millions.

“Price-performance ratio”:
area-time product for chips.

“RAM” metric (adding two 64-bit integers has same cost as accessing array of size 2^{64}) is not realistic; “*AT*” metric is realistic.

“Asymptotic”: We systematically suppress polynomial factors. Our speedups are superpolynomial.

Best result known for *one* key:

time $L^{1.185632}$

using chip area $L^{0.790420}$;

AT is $L^{1.976052}$.

Best result known for *one* key:

time $L^{1.185632}$

using chip area $L^{0.790420}$;

AT is $L^{1.976052}$.

Our main result for

a batch of $L^{0.5}$ keys:

time $L^{1.022400}$

using chip area $L^{1.181600}$;

AT per key is $L^{1.704000}$.

Best result known for *one* key:

time $L^{1.185632}$

using chip area $L^{0.790420}$;

AT is $L^{1.976052}$.

Our main result for

a batch of $L^{0.5}$ keys:

time $L^{1.022400}$

using chip area $L^{1.181600}$;

AT per key is $L^{1.704000}$.

This paper also looks more closely

at $L^{o(1)}$, analyzing asymptotic

speedup from early-abort ECM.

Results are not what one would

guess from 1982 Pomerance.

Asymptotic consequences:

1. Attack cost per key is reduced, so attacker can target lower-value keys.
2. Primary bottleneck is low-memory factorization—well suited for off-the-shelf graphics cards.
3. Attack time is reduced (and can be reduced more), breaking key rotation.

Asymptotic consequences:

1. Attack cost per key is reduced, so attacker can target lower-value keys.
2. Primary bottleneck is low-memory factorization—well suited for off-the-shelf graphics cards.
3. Attack time is reduced (and can be reduced more), breaking key rotation.

“Do the asymptotics really kick in before 1024 bits?” — Maybe not, but no basis for confidence.

Eratosthenes for smoothness

Sieving small integers $i > 0$
using primes 2, 3, 5, 7:

1				
2	2			
3		3		
4	2 2			
5			5	
6	2	3		
7				7
8	2 2 2			
9		3 3		
10	2		5	
11				
12	2 2	3		
13				
14	2			7
15		3	5	
16	2 2 2 2			
17				
18	2	3 3		
19				
20	2 2		5	

etc.

The Q sieve

Sieving i and $611 + i$ for small i
using primes 2, 3, 5, 7:

1				
2	2			
3		3		
4	2 2			
5			5	
6	2	3		
7				7
8	2 2 2			
9		3 3		
10	2		5	
11				
12	2 2	3		
13				
14	2			7
15		3	5	
16	2 2 2 2			
17				
18	2	3 3		
19				
20	2 2		5	

612	2 2	3 3		
613				
614	2			
615		3	5	
616	2 2 2			7
617				
618	2	3		
619				
620	2 2		5	
621		3 3 3		
622	2			
623				7
624	2 2 2 2	3		
625			5 5 5 5	
626	2			
627		3		
628	2 2			
629				
630	2	3 3	5	7
631				

etc.

Have complete factorization of the congruences $i \equiv 611 + i$ for some i 's.

$$14 \cdot 625 = 2^1 3^0 5^4 7^1.$$

$$64 \cdot 675 = 2^6 3^3 5^2 7^0.$$

$$75 \cdot 686 = 2^1 3^1 5^2 7^3.$$

$$\begin{aligned} &14 \cdot 64 \cdot 75 \cdot 625 \cdot 675 \cdot 686 \\ &= 2^8 3^4 5^8 7^4 = (2^4 3^2 5^4 7^2)^2. \end{aligned}$$

$$\begin{aligned} &\gcd\{611, 14 \cdot 64 \cdot 75 - 2^4 3^2 5^4 7^2\} \\ &= 47. \end{aligned}$$

$$611 = 47 \cdot 13.$$

The number-field sieve

Generalize $i \equiv i + N \pmod{N}$

$\rightarrow a \equiv a + bN \pmod{N}$

$\rightarrow a - bm \equiv a - b\alpha \pmod{m - \alpha}$

for root $\alpha \in \mathbf{C}$

of nonzero integer poly.

For any m can find α

so that factoring $m - \alpha$

produces factorization of N .

Optimal choice of $\log m$ is

$(\mu + o(1))(\log N)^{2/3}(\log \log N)^{1/3}$.

RAM cost analysis

1993 Buhler–Lenstra–Pomerance:

Smoothness bound $L^{0.961500}$.

Sieve $L^{1.923000}$ pairs (a, b) .

Find $L^{0.961500}$ pairs

with $a - bm$ and $a - b\alpha$ smooth.

Total RAM time $L^{1.923000}$.

1993 Coppersmith:

Total RAM time $L^{1.901884}$

using multiple number fields.

(Multiple number fields

don't seem to combine well

with AT , factory, et al.)

AT cost analysis

Sieving is a disaster
in realistic cost metric.

AT cost $L^{2.403750}$.

AT cost analysis

Sieving is a disaster
in realistic cost metric.

AT cost $L^{2.403750}$.

Fix: find smooth using ECM.

AT cost $L^{1.923000}$.

AT cost analysis

Sieving is a disaster
in realistic cost metric.

AT cost $L^{2.403750}$.

Fix: find smooth using ECM.

AT cost $L^{1.923000}$.

Linear algebra is also a disaster.

AT cost $L^{2.403750}$.

AT cost analysis

Sieving is a disaster
in realistic cost metric.

AT cost $L^{2.403750}$.

Fix: find smooth using ECM.

AT cost $L^{1.923000}$.

Linear algebra is also a disaster.

AT cost $L^{2.403750}$.

Semi-fix: Reduce smoothness
bounds to rebalance.

AT cost $L^{1.976052}$.

(2001 Bernstein)

The factorization factory

1993 Coppersmith:

There *exists* an algorithm that factors any integer with same #bits as N in RAM time $L^{1.638587}$.

Smoothness bound $L^{0.819290}$.

Smaller than before, so need more (a, b) .

Algorithm *knows* all (a, b) such that $a - bm$ is smooth.

Note: one m works for all N .

Algorithm uses ECM to check whether $a - b\alpha_N$ is smooth.

Finding this algorithm

is slower than running it.

Need to precompute all (a, b)

such that $a - bm$ is smooth.

RAM time $L^{2.006853}$.

Finding this algorithm

is slower than running it.

Need to precompute all (a, b)

such that $a - bm$ is smooth.

RAM time $L^{2.006853}$.

Standard conversion of

precomputation into batching:

if there are enough targets,

more than $L^{0.368266}$,

then precomputation cost

becomes negligible.

Finding this algorithm
is slower than running it.
Need to precompute all (a, b)
such that $a - bm$ is smooth.
RAM time $L^{2.006853}$.

Standard conversion of
precomputation into batching:
if there are enough targets,
more than $L^{0.368266}$,
then precomputation cost
becomes negligible.

The big problem: Coppersmith's
algorithm has size $L^{1.638587}$.
Huge AT cost; useless in reality.

Batch NFS

Goal: Optimize AT asymptotics.

1. Generate (a, b) in parallel.

Test $a - bm$ for smoothness.

2. Make many copies of each N , close to each (a, b) generator.

When smooth $a - bm$ is found, test each $a - b\alpha_N$ for smoothness.

3. After all smooths are found, reorganize: for each N , bring relevant (a, b) close together.

4. Linear algebra.

<p>Generate (a, b). Is $a - bm$ smooth? If so, store. Repeat.</p>	<p>Generate (a, b). Is $a - bm$ smooth? If so, store. Repeat.</p>	<p>Generate (a, b). Is $a - bm$ smooth? If so, store. Repeat.</p>	<p>Generate (a, b). Is $a - bm$ smooth? If so, store. Repeat.</p>
<p>Generate (a, b). Is $a - bm$ smooth? If so, store. Repeat.</p>	<p>Generate (a, b). Is $a - bm$ smooth? If so, store. Repeat.</p>	<p>Generate (a, b). Is $a - bm$ smooth? If so, store. Repeat.</p>	<p>Generate (a, b). Is $a - bm$ smooth? If so, store. Repeat.</p>
<p>Generate (a, b). Is $a - bm$ smooth? If so, store. Repeat.</p>	<p>Generate (a, b). Is $a - bm$ smooth? If so, store. Repeat.</p>	<p>Generate (a, b). Is $a - bm$ smooth? If so, store. Repeat.</p>	<p>Generate (a, b). Is $a - bm$ smooth? If so, store. Repeat.</p>
<p>Generate (a, b). Is $a - bm$ smooth? If so, store. Repeat.</p>	<p>Generate (a, b). Is $a - bm$ smooth? If so, store. Repeat.</p>	<p>Generate (a, b). Is $a - bm$ smooth? If so, store. Repeat.</p>	<p>Generate (a, b). Is $a - bm$ smooth? If so, store. Repeat.</p>

	<p>smooth?</p> <p>If so, store.</p> <p>Repeat.</p>	
<p>(a, b).</p> <p>$a - bm$</p> <p>smooth?</p> <p>If so, store.</p> <p>Repeat.</p>	<p>Generate (a, b).</p> <p>Is $a - bm$</p> <p>smooth?</p> <p>If so, store.</p> <p>Repeat.</p>	<p>Generate</p> <p>Is $a -$</p> <p>smoo</p> <p>If so, st</p> <p>Repea</p>
<p>te (a, b).</p> <p>$a - bm$</p> <p>oth?</p> <p>ore.</p>	<p>Generate (a, b).</p> <p>Is $a - bm$</p> <p>smooth?</p> <p>If so, store.</p> <p>Repeat.</p>	<p>Generate</p> <p>Is $a -$</p> <p>smo</p> <p>If so</p>
	<p>Generate (a, b).</p> <p>Is $a - bm$</p>	

<p>Is $a - b\alpha_1$ smooth? If so, store. Send (a, b) right. Repeat.</p>	<p>Is $a - b\alpha_2$ smooth? If so, store. Send (a, b) right. Repeat.</p>	<p>Is $a - b\alpha_3$ smooth? If so, store. Send (a, b) right. Repeat.</p>	<p>Is $a - b\alpha_4$ smooth? If so, store. Send (a, b) down. Repeat.</p>
<p>Is $a - b\alpha_5$ smooth? If so, store. Send (a, b) up. Repeat.</p>	<p>Is $a - b\alpha_6$ smooth? If so, store. Send (a, b) left. Repeat.</p>	<p>Is $a - b\alpha_7$ smooth? If so, store. Send (a, b) left. Repeat.</p>	<p>Is $a - b\alpha_8$ smooth? If so, store. Send (a, b) left. Repeat.</p>
<p>Is $a - b\alpha_9$ smooth? If so, store. Send (a, b) right. Repeat.</p>	<p>Is $a - b\alpha_{10}$ smooth? If so, store. Send (a, b) right. Repeat.</p>	<p>Is $a - b\alpha_{11}$ smooth? If so, store. Send (a, b) right. Repeat.</p>	<p>Is $a - b\alpha_{12}$ smooth? If so, store. Send (a, b) down. Repeat.</p>
<p>Is $a - b\alpha_{13}$ smooth? If so, store. Send (a, b) up. Repeat.</p>	<p>Is $a - b\alpha_{14}$ smooth? If so, store. Send (a, b) left. Repeat.</p>	<p>Is $a - b\alpha_{15}$ smooth? If so, store. Send (a, b) left. Repeat.</p>	<p>Is $a - b\alpha_{16}$ smooth? If so, store. Send (a, b) left. Repeat.</p>

If so, store. Send (a, b) left. Repeat.	If so, store. Send (a, b) left. Repeat.
---	---

Is $a - b\alpha_1$ smooth? If so, store. Send (a, b) right. Repeat.	Is $a - b\alpha_2$ smooth? If so, store. Send (a, b) right. Repeat.	Is $a - b\alpha_3$ smooth? If so, store. Send (a, b) right. Repeat.	Is $a - b\alpha_4$ smooth? If so, store. Send (a, b) down. Repeat.
Is $a - b\alpha_5$ smooth? If so, store. Send (a, b) up. Repeat.	Is $a - b\alpha_6$ smooth? If so, store. Send (a, b) left. Repeat.	Is $a - b\alpha_7$ smooth? If so, store. Send (a, b) left. Repeat.	Is $a - b\alpha_8$ smooth? If so, store. Send (a, b) left. Repeat.
Is $a - b\alpha_9$ smooth? If so, store. Send (a, b) right. Repeat.	Is $a - b\alpha_{10}$ smooth? If so, store. Send (a, b) right. Repeat.	Is $a - b\alpha_{11}$ smooth? If so, store. Send (a, b) right. Repeat.	Is $a - b\alpha_{12}$ smooth? If so, store. Send (a, b) down. Repeat.
Is $a - b\alpha_{13}$ smooth? If so, store. Send (a, b) up. Repeat.	Is $a - b\alpha_{14}$ smooth? If so, store. Send (a, b) left. Repeat.	Is $a - b\alpha_{15}$ smooth? If so, store. Send (a, b) left. Repeat.	Is $a - b\alpha_{16}$ smooth? If so, store. Send (a, b) down. Repeat.

Is $a - b\alpha_{17}$ smooth?	Is $a - b\alpha_{18}$ smooth?
-------------------------------	-------------------------------

