Public-key cryptography

Daniel J. Bernstein
Tanja Lange

Part II:
Factorization

15 August 2017

Sage scripts for some algorithms,
joint work with Heninger:
facthacks.cr.yp.to

Q sieve

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.

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" $i(611+i)$ for some i's.

$14 \cdot 625 = 2^1 3^0 5^4 7^1$.
$64 \cdot 675 = 2^6 3^3 5^2 7^0$.
$75 \cdot 686 = 2^1 3^1 5^2 7^3$.

$14 \cdot 64 \cdot 75 \cdot 625 \cdot 675 \cdot 686$
$= 2^8 3^4 5^8 7^4 = (2^4 3^2 5^4 7^2)^2$.

$\gcd\left\{611, 14 \cdot 64 \cdot 75 - 2^4 3^2 5^4 7^2\right\}$
$= 47$.

$611 = 47 \cdot 13$.

Why did this find a factor of 611?

Was it just blind luck:

$\gcd\{611, \text{random}\} = 47$?

Why did this find a factor of 611?
Was it just blind luck:
$\gcd\{611, \text{random}\} = 47$?

No.

By construction 611 divides $s^2 - t^2$
where $s = 14 \cdot 64 \cdot 75$
and $t = 2^4 3^2 5^4 7^2$.
So each prime > 7 dividing 611
divides either $s - t$ or $s + t$.

Not terribly surprising
(but not guaranteed in advance!)
that one prime divided $s - t$
and the other divided $s + t$.

Why did the first three
completely factored congruences
have square product?
Was it just blind luck?

Why did the first three completely factored congruences have square product?
Was it just blind luck?

Yes. The exponent vectors $(1,0,4,1),(6,3,2,0),(1,1,2,3)$ happened to have sum 0 mod 2.

Why did the first three completely factored congruences have square product? Was it just blind luck?

Yes. The exponent vectors $(1, 0, 4, 1), (6, 3, 2, 0), (1, 1, 2, 3)$ happened to have sum 0 mod 2.

But we didn't need this luck! Given long sequence of vectors, easily find nonempty subsequence with sum 0 mod 2.

This is linear algebra over $\mathbf{F}_2$.
Guaranteed to find subsequence if number of vectors exceeds length of each vector.

e.g. for $n = 671$:

$$
\begin{aligned}
1(n+1) &= 2^5 3^1 5^0 7^1;\\
4(n+4) &= 2^2 3^3 5^2 7^0;\\
15(n+15) &= 2^1 3^1 5^1 7^3;\\
49(n+49) &= 2^4 3^2 5^1 7^2;\\
64(n+64) &= 2^6 3^1 5^1 7^2.
\end{aligned}
$$

$\mathbf{F}_2$-kernel of exponent matrix is gen by (0 1 0 1 1) and (1 0 1 1 0); e.g., $1(n+1)15(n+15)49(n+49)$ is a square.

Plausible conjecture: **Q** sieve can separate the odd prime divisors of any n, not just 611.

Given n and parameter y:

Try to completely factor $i(n+i)$ for $i \in \{1, 2, 3, \ldots, y^2\}$ into products of primes $\le y$.

Look for nonempty set I of i's with $i(n+i)$ completely factored and with $\prod_{i \in I} i(n+i)$ square.

Compute $\gcd\{n, s-t\}$ where $s = \prod_{i \in I} i$ and $t = \sqrt{\prod_{i \in I} i(n+i)}$.

How large does y have to be for this to find a square?

Uniform random integer in $[1, n]$ has $n^{1/u}$-smoothness chance roughly u^{-u}.

Plausible conjecture:
$\mathbf{Q}$ sieve succeeds
with $y = \lfloor n^{1/u} \rfloor$
for all $n \geq u^{(1+o(1))u^2}$;
here $o(1)$ is as $u \to \infty$.

More generally, if $y \in \exp\sqrt{\left(\frac{1}{2c}+o(1)\right)\log n \log\log n}$, conjectured y-smoothness chance is $1/y^{c+o(1)}$.

Find enough smooth congruences by changing the range of i's: replace y^2 with $y^{c+1+o(1)} = \exp\sqrt{\left(\frac{(c+1)^2+o(1)}{2c}\right)\log n \log\log n}$.

Increasing c past 1 increases number of i's but reduces linear-algebra cost. So linear algebra never dominates when y is chosen properly.

Improving smoothness chances

Smoothness chance of $i(n+i)$ degrades as i grows. Smaller for $i \approx y^2$ than for $i \approx y$.

Crude analysis: $i(n+i)$ grows.
$\approx yn$ if $i \approx y$;
$\approx y^2 n$ if $i \approx y^2$.

More careful analysis:
$n+i$ doesn't degrade, but
i is always smooth for $i \leq y$,
only 30% chance for $i \approx y^2$.

Can we select congruences to avoid this degradation?

Choose q, square of large prime.
Choose a "q-sublattice" of i's:
arithmetic progression of i's
where q divides each $i(n+i)$.
e.g. progression $q-(n \bmod q)$,
$2q-(n \bmod q)$, $3q-(n \bmod q)$,
etc.

Check smoothness of
generalized congruence $i(n+i)/q$
for i's in this sublattice.
e.g. check whether $i,(n+i)/q$ are
smooth for $i = q-(n \bmod q)$ etc.

Try many large q's.
Rare for i's to overlap.

e.g. $n = 314159265358979323$:

Original $\mathbf{Q}$ sieve:

i	$n+i$
1	314159265358979324
2	314159265358979325
3	314159265358979326

Use 997^2-sublattice,
$i \in 802458 + 994009\mathbf{Z}$:

i	$(n+i)/997^2$
802458	316052737309
1796467	316052737310
2790476	316052737311

Crude analysis: Sublattices eliminate the growth problem. Have practically unlimited supply of generalized congruences

$$(q-(n \bmod q))\frac{n+q-(n \bmod q)}{q}$$

between 0 and n.

More careful analysis: Sublattices are even better than that! For $q \approx n^{1/2}$ have $i \approx (n+i)/q \approx n^{1/2} \approx y^{u/2}$ so smoothness chance is roughly $(u/2)^{-u/2}(u/2)^{-u/2} = 2^u/u^u$, 2^u times larger than before.

Even larger improvements
from changing polynomial $i(n+i)$.

"Quadratic sieve" (QS) uses
$i^2 - n$ with $i \approx \sqrt{n}$;
have $i^2 - n \approx n^{1/2+o(1)}$,
much smaller than n.

Even larger improvements
from changing polynomial $i(n+i)$.

"Quadratic sieve" (QS) uses
$i^2 - n$ with $i \approx \sqrt{n}$;
have $i^2 - n \approx n^{1/2+o(1)}$,
much smaller than n.

"MPQS" improves $o(1)$
using sublattices: $(i^2 - n)/q$.
But still $\approx n^{1/2}$.

Even larger improvements
from changing polynomial $i(n+i)$.

"Quadratic sieve" (QS) uses
$i^2 - n$ with $i \approx \sqrt{n}$;
have $i^2 - n \approx n^{1/2+o(1)}$,
much smaller than n.

"MPQS" improves $o(1)$
using sublattices: $(i^2 - n)/q$.
But still $\approx n^{1/2}$.

"Number-field sieve" (NFS)
achieves $n^{o(1)}$.

Generalizing beyond $\mathbf{Q}$

The $\mathbf{Q}$ sieve is a special case of the number-field sieve.

Recall how the $\mathbf{Q}$ sieve factors 611:

Form a square as product of $i(i + 611j)$ for several pairs (i, j):
$14(625) \cdot 64(675) \cdot 75(686)$
$= 4410000^2$.

$\gcd\{611, 14 \cdot 64 \cdot 75 - 4410000\}$
$= 47$.

The $\mathbf{Q}(\sqrt{14})$ sieve
factors 611 as follows:

Form a square
as product of $(i + 25j)(i + \sqrt{14}j)$
for several pairs (i, j):

$$(-11 + 3 \cdot 25)(-11 + 3\sqrt{14})$$
$$\cdot (3 + 25)(3 + \sqrt{14})$$
$$= (112 - 16\sqrt{14})^2.$$

Compute
$s = (-11 + 3 \cdot 25) \cdot (3 + 25)$,
$t = 112 - 16 \cdot 25$,
$\gcd\{611, s - t\} = 13$.

Why does this work?

Answer: Have ring morphism
$\mathbf{Z}[\sqrt{14}] \to \mathbf{Z}/611$, $\sqrt{14} \mapsto 25$,
since $25^2 = 14$ in $\mathbf{Z}/611$.

Apply ring morphism to square:

$$(-11 + 3 \cdot 25)(-11 + 3 \cdot 25)$$
$$\cdot (3 + 25)(3 + 25)$$
$$= (112 - 16 \cdot 25)^2 \text{ in } \mathbf{Z}/611.$$

i.e. $s^2 = t^2$ in $\mathbf{Z}/611$.

Unsurprising to find factor.

Generalize from $(x^2 - 14, 25)$
to (f, m) with irred $f \in \mathbf{Z}[x]$,
$m \in \mathbf{Z}$, $f(m) \in n\mathbf{Z}$.

Write $d = \deg f$,
$f = f_d x^d + \cdots + f_1 x^1 + f_0 x^0$.

Can take $f_d = 1$ for simplicity,
but larger f_d allows
better parameter selection.

Pick $\alpha \in \mathbf{C}$, root of f.
Then $f_d\alpha$ is a root of
monic $g = f_d^{d-1} f(x/f_d) \in \mathbf{Z}[x]$.

$$\mathbf{Q}(\alpha) \leftarrow \mathcal{O} \leftarrow \mathbf{Z}[f_d\alpha] \xrightarrow{f_d\alpha \mapsto f_d m} \mathbf{Z}/n$$

Build square in $\mathbf{Q}(\alpha)$ from congruences $(i - jm)(i - j\alpha)$ with $i\mathbf{Z} + j\mathbf{Z} = \mathbf{Z}$ and $j > 0$.

Could replace $i - jx$ by higher-deg irred in $\mathbf{Z}[x]$; quadratics seem fairly small for some number fields. But let's not bother.

Say we have a square $\prod_{(i,j)\in S}(i - jm)(i - j\alpha)$ in $\mathbf{Q}(\alpha)$; now what?

$\prod(i - jm)(i - j\alpha)f_d^2$
is a square in $\mathcal{O}$,
ring of integers of $\mathbf{Q}(\alpha)$.

Multiply by $g'(f_d\alpha)^2$,
putting square root into $\mathbf{Z}[f_d\alpha]$:
compute r with $r^2 = g'(f_d\alpha)^2 \cdot$
$\prod(i - jm)(i - j\alpha)f_d^2$.

Then apply the ring morphism
$\varphi : \mathbf{Z}[f_d\alpha] \to \mathbf{Z}/n$ taking
$f_d\alpha$ to $f_d m$. Compute gcd$\{n,$
$\varphi(r) - g'(f_d m) \prod(i - jm)f_d\}$.
In $\mathbf{Z}/n$ have $\varphi(r)^2 =$
$g'(f_d m)^2 \prod(i - jm)^2 f_d^2$.

How to find square product of congruences $(i - jm)(i - j\alpha)$?

Start with congruences for, e.g., y^2 pairs (i, j).

Look for y-smooth congruences: y-smooth $i - jm$ and y-smooth $f_d \operatorname{norm}(i - j\alpha) = f_d i^d + \cdots + f_0 j^d = j^d f(i/j)$. Here "$y$-smooth" means "has no prime divisor $> y$."

Find enough smooth congruences. Perform linear algebra on exponent vectors mod 2.

Asymptotic cost exponents

Number of bit operations
in number-field sieve,
with theorists' parameters,
is $L^{1.90\ldots+o(1)}$ where $L =$
$\exp((\log n)^{1/3}(\log\log n)^{2/3})$.

What are theorists' parameters?

Choose degree d with
$d/(\log n)^{1/3}(\log\log n)^{-1/3}$
$\in 1.40\ldots + o(1)$.

Choose integer $m \approx n^{1/d}$.
Write n as
$m^d + f_{d-1} m^{d-1} + \cdots + f_1 m + f_0$
with each f_k below $n^{(1+o(1))/d}$.
Choose f with some randomness
in case there are bad f's.

Test smoothness of $i - jm$
for all coprime pairs (i, j)
with $1 \le i, j \le L^{0.95\ldots+o(1)}$,
using primes $\le L^{0.95\ldots+o(1)}$.

$L^{1.90\ldots+o(1)}$ pairs.
Conjecturally $L^{1.65\ldots+o(1)}$
smooth values of $i - jm$.

Use $L^{0.12\ldots+o(1)}$ number fields.

For each (i, j)
with smooth $i - jm$,
test smoothness of $i - j\alpha$
and $i - j\beta$ and so on,
using primes $\le L^{0.82\ldots+o(1)}$.

$L^{1.77\ldots+o(1)}$ tests.
Each $|j^d f(i/j)| \le m^{2.86\ldots+o(1)}$.
Conjecturally $L^{0.95\ldots+o(1)}$
smooth congruences.

$L^{0.95\ldots+o(1)}$ components
in the exponent vectors.

Three sizes of numbers here:

$(\log n)^{1/3}(\log\log n)^{2/3}$ bits:
y, i, j.

$(\log n)^{2/3}(\log\log n)^{1/3}$ bits:
m, $i - jm$, $j^d f(i/j)$.

$\log n$ bits: n.

Unavoidably $1/3$ in exponent:
usual smoothness optimization
forces $(\log y)^2 \approx \log m$;
balancing norms with m
forces $d \log y \approx \log m$;
and $d \log m \approx \log n$.

Batch NFS

The number-field sieve used $L^{1.90\ldots+o(1)}$ bit operations finding smooth $i - jm$; only $L^{1.77\ldots+o(1)}$ bit operations finding smooth $j^d f(i/j)$.

Many n's can share one m; $L^{1.90\ldots+o(1)}$ bit operations to find squares for *all* n's.

Oops, linear algebra hurts; fix by reducing y.
But still end up factoring batch in much less time than factoring each n separately.

Asymptotic batch-NFS parameters:

$d/(\log n)^{1/3}(\log\log n)^{-1/3}$
$\in 1.10\ldots + o(1)$.

Primes $\le L^{0.82\ldots+o(1)}$.

$1 \le i, j \le L^{1.00\ldots+o(1)}$.

Computation independent of n finds $L^{1.64\ldots+o(1)}$ smooth values $i - jm$.

$L^{1.64\ldots+o(1)}$ operations for each target n.

Asymptotic batch-NFS parameters:

$d/(\log n)^{1/3}(\log\log n)^{-1/3}$
$\in 1.10\ldots + o(1)$.

Primes $\le L^{0.82\ldots+o(1)}$.

$1 \le i, j \le L^{1.00\ldots+o(1)}$.

Computation independent of n finds $L^{1.64\ldots+o(1)}$ smooth values $i - jm$.

$L^{1.64\ldots+o(1)}$ operations for each target n.

Wait: how do we recognize smooth integers so quickly?

The rho method

Define $\rho_0 = 0$, $\rho_{k+1} = \rho_k^2 + 11$.

Every prime $\le 2^{20}$ divides $S = (\rho_1 - \rho_2)(\rho_2 - \rho_4)(\rho_3 - \rho_6) \cdots (\rho_{3575} - \rho_{7150})$.
Also many larger primes.

Can compute $\gcd\{c, S\}$ using $\approx 2^{14}$ multiplications mod c, very little memory.

Compare to $\approx 2^{16}$ divisions for trial division up to 2^{20}.

More generally: Choose z.
Compute $\gcd\{c, S\}$ where $S = (\rho_1 - \rho_2)(\rho_2 - \rho_4)\cdots(\rho_z - \rho_{2z})$.

How big does z have to be for all primes $\le y$ to divide S?

Plausible conjecture: $y^{1/2+o(1)}$; so $y^{1/2+o(1)}$ mults mod c.

Reason: Consider first collision in $\rho_1 \bmod p, \rho_2 \bmod p, \ldots$.
If $\rho_i \bmod p = \rho_j \bmod p$ then $\rho_k \bmod p = \rho_{2k} \bmod p$ for $k \in (j - i)\mathbf{Z} \cap [i, \infty] \cap [j, \infty]$.

The $p - 1$ method

$S_1 = 2^{232792560} - 1$
has prime divisors
3, 5, 7, 11, 13, 17, 19, 23, 29, 31,
37, 41, 43, 53, 61, 67, 71, 73, 79,
89, 97, 103, 109, 113, 127, 131,
137, 151, 157, 181, 191, 199 etc.

These divisors include
70 of the 168 primes $\le 10^3$;
156 of the 1229 primes $\le 10^4$;
296 of the 9592 primes $\le 10^5$;
470 of the 78498 primes $\le 10^6$;
etc.

An odd prime p
divides $2^{232792560} - 1$
iff order of 2 in the
multiplicative group $\mathbf{F}_p^*$
divides $s = 232792560$.

Many ways for this to happen:
232792560 has 960 divisors.

Why so many?
Answer: $s = 232792560$
$= \operatorname{lcm}\{1, 2, 3, 4, \ldots, 20\}$
$= 2^4 \cdot 3^2 \cdot 5 \cdot 7 \cdot 11 \cdot 13 \cdot 17 \cdot 19.$

Can compute $2^{232792560} - 1$ using 41 ring operations. (Side note: 41 is not minimal.)

Ring operation: 0, 1, $+$, $-$, $\cdot$.

This computation: 1; $2 = 1 + 1$; $2^2 = 2 \cdot 2$; $2^3 = 2^2 \cdot 2$; $2^6 = 2^3 \cdot 2^3$; $2^{12} = 2^6 \cdot 2^6$; $2^{13} = 2^{12} \cdot 2$; 2^{26}; 2^{27}; 2^{54}; 2^{55}; 2^{110}; 2^{111}; 2^{222}; 2^{444}; 2^{888}; 2^{1776}; 2^{3552}; 2^{7104}; 2^{14208}; 2^{28416}; 2^{28417}; 2^{56834}; 2^{113668}; 2^{227336}; 2^{454672}; 2^{909344}; 2^{909345}; $2^{1818690}$; $2^{1818691}$; $2^{3637382}$; $2^{3637383}$; $2^{7274766}$; $2^{7274767}$; $2^{14549534}$; $2^{14549535}$; $2^{29099070}$; $2^{58198140}$; $2^{116396280}$; $2^{232792560}$; $2^{232792560} - 1$.

Given positive integer n,
can compute $2^{232792560} - 1 \bmod n$
using 41 operations in $\mathbf{Z}/n$.
Notation: $a \bmod b = a - b\lfloor a/b \rfloor$.

e.g. $n = 8597231219$: ...

$$\begin{aligned}
2^{27} \bmod n &= 134217728; \\
2^{54} \bmod n &= 134217728^2 \bmod n \\
&= 935663516; \\
2^{55} \bmod n &= 1871327032; \\
2^{110} \bmod n &= 1871327032^2 \bmod n \\
&= 1458876811; \ldots; \\
2^{232792560} - 1 \bmod n &= 5626089344.
\end{aligned}$$

Given positive integer n,
can compute $2^{232792560} - 1 \bmod n$
using 41 operations in $\mathbf{Z}/n$.
Notation: $a \bmod b = a - b\lfloor a/b \rfloor$.

e.g. $n = 8597231219$: ...

$$
\begin{aligned}
2^{27} \bmod n &= 134217728;\\
2^{54} \bmod n &= 134217728^2 \bmod n\\
&= 935663516;\\
2^{55} \bmod n &= 1871327032;\\
2^{110} \bmod n &= 1871327032^2 \bmod n\\
&= 1458876811; \ldots;
\end{aligned}
$$

$$2^{232792560} - 1 \bmod n = 5626089344.$$

Easy extra computation (Euclid):
$\gcd\{5626089344, n\} = 991$.

This $p-1$ method (1974 Pollard) quickly factored $n = 8597231219$. Main work: 27 squarings mod n.

Could instead have checked n's divisibility by $2, 3, 5, \ldots$. The 167th trial division would have found divisor 991.

Not clear which method is better. Dividing by small p is faster than squaring mod n. The $p-1$ method finds only 70 of the primes ≤ 1000; trial division finds all 168 primes.

Scale up to larger exponent
$s = \text{lcm}\{1, 2, 3, 4, \ldots, 100\}$:
using 136 squarings mod n
find 2317 of the primes $\le 10^5$.

Is a squaring mod n
faster than 17 trial divisions?

Or $s = \text{lcm}\{1, 2, 3, 4, \ldots, 1000\}$:
using 1438 squarings mod n
find 180121 of the primes $\le 10^7$.

Is a squaring mod n
faster than 125 trial divisions?

Extra benefit:
no need to store the primes.

Plausible conjecture: if K is $\exp\sqrt{\left(\frac{1}{2}+o(1)\right)\log H\log\log H}$ then $p-1$ divides $\operatorname{lcm}\{1,2,\ldots,K\}$ for $H/K^{1+o(1)}$ primes $p \le H$.
Same if $p-1$ is replaced by order of 2 in $\mathbf{F}_p^*$.

So uniform random prime $p \le H$ divides $2^{\operatorname{lcm}\{1,2,\ldots,K\}}-1$ with probability $1/K^{1+o(1)}$.

$(1.4\ldots+o(1))K$ squarings mod n produce $2^{\operatorname{lcm}\{1,2,\ldots,K\}}-1 \bmod n$.

Similar time spent on trial division finds far fewer primes for large H.

The $p+1$ factorization method

(1982 Williams)

Define $(X, Y) \in \mathbf{Q} \times \mathbf{Q}$ as the 232792560th multiple of $(3/5, 4/5)$ in the group Clock($\mathbf{Q}$).

The integer $S_2 = 5^{232792560} X$ is divisible by
82 of the primes $\le 10^3$;
223 of the primes $\le 10^4$;
455 of the primes $\le 10^5$;
720 of the primes $\le 10^6$;
etc.

Given an integer n,
compute $5^{232792560}X \bmod n$
and compute gcd with n,
hoping to factor n.

Many p's not found by $\mathbf{F}_p^*$
are found by Clock($\mathbf{F}_p$).

If -1 is not a square mod p
and $p+1$ divides 232792560
then $5^{232792560}X \bmod p = 0$.

Proof: $p \equiv 3 \pmod 4$,
so $(4/5 + 3i/5)^p = 4/5 - 3i/5$,
so $(p+1)(3/5, 4/5) = (0, 1)$
in the group Clock($\mathbf{F}_p$),
so $232792560(3/5, 4/5) = (0, 1)$.

The elliptic-curve method

Replace clock group with
a random elliptic curve.

Order of elliptic-curve group
$\in [p+1-2\sqrt{p}, p+1+2\sqrt{p}]$.
If a curve fails, try another.

Good news (for the attacker):
All primes $\le H$
seem to be found after a
reasonable number of curves.
Time subexponential in H.

More reading

eecm.cr.yp.to

cr.yp.to/papers.html#batchnfs

smartfacts.cr.yp.to
"Factoring RSA keys from certified smart cards: Coppersmith in the wild"

eprint.iacr.org/2016/961
"A kilobit hidden SNFS discrete logarithm computation"

eprint.iacr.org/2017/142
"Computing generator . . . and application to cryptanalysis of a [lattice-based] FHE scheme"