The NSA sieving circuit

D. J. Bernstein
University of Illinois at Chicago

NSF DMS–9970409

Sieving c and $611 + c$ for small c:

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 $c(611 + c)$ for some c's.

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

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

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

$611 = 47 \cdot 13$.

Given n and parameter y:

1. Use powers of primes $\leq y$ to sieve c and $n+c$ for $1 \leq c \leq y^2$.

2. Look for nonempty set of c's with $c(n+c)$ completely factored and with $\prod_c c(n+c)$ square.

3. Compute $\gcd\{x, n\}$ where $x = \prod_c c - \sqrt{\prod_c c(n+c)}$.

This is the **Q sieve**.

Same principles:

Continued fraction method
(Lehmer, Powers,
Brillhart, Morrison).

Linear sieve (Schroeppel).

Quadratic sieve (Pomerance).

Number field sieve
(Pollard, Buhler, Lenstra,
Pomerance, Adleman).

The basic sieve problem

Handle sieving in pieces:
sieve $\{n+1, \ldots, n+y\}$;
sieve $\{n+y+1, \ldots, n+2y\}$;
sieve $\{n+2y+1, \ldots, n+3y\}$;
etc.

The **basic sieve problem**:
Sieve $\{n+1, n+2, \ldots, n+y\}$
using primes $\leq y$.

Don't worry about prime powers.

Trial division

For each $s \in \{n+1, \ldots, n+y\}$:

For each prime $p \leq y$:

Check if p divides s.

$y^{2+\epsilon}$ time, y^{ϵ} hardware.

Can handle p's in parallel:
$y^{1+\epsilon}$ time, $y^{1+\epsilon}$ hardware.

Georgia Cracker (Pomerance),
TWINKLE (Shamir), etc.

Sieving in memory

Use array of size y,
locations $n+1, n+2, \ldots, n+y$.

For each prime p:

    For each multiple s of p:

        Mark p in location s.

Total number of marks
$\approx \sum_p y/p \approx y \log\log y$.

$y^{1+\epsilon}$ time, $y^{1+\epsilon}$ hardware.

In other words:

Consider all pairs (p, s)
where s is a multiple of p.

Use a distribution sort
to sort these pairs
in order of s.

Then can see p's for each s.

$y = 9,\ n = 611$:

$(2, 612)$ $(2, 614)$ $(2, 616)$ $(2, 618)$
$(2, 620)$ $(3, 612)$ $(3, 615)$ $(3, 618)$
$(5, 615)$ $(5, 620)$ $(7, 616)$

Sorted: $(2, 612)$ $(3, 612)$ $(2, 614)$
$(3, 615)$ $(5, 615)$ $(2, 616)$ $(7, 616)$
$(2, 618)$ $(3, 618)$ $(2, 620)$ $(5, 620)$

The NSA circuit

Build $y^{1/2} \times y^{1/2}$ mesh
of simple processors.

Consider all pairs (p, i)
with $1 \leq i \leq \lceil y/p \rceil$.
$y^{1+\epsilon}$ such pairs.

Spread pairs among processors.
Build y^{ϵ} pairs (p, i)
into each processor.

Spread c's among processors.

Each processor is $\#c$ for one c.

$\#1$ $(2,1)$ $(5,2)$	$\#2$ $(2,2)$ $(7,1)$	$\#3$ $(2,3)$ $(7,2)$
$\#4$ $(2,4)$	$\#5$ $(2,5)$	$\#6$ $(3,1)$
$\#7$ $(3,2)$	$\#8$ $(3,3)$	$\#9$ $(5,1)$

Given n:

For each (p, i), processor
generates ith multiple s of p
in $\{n+1, n+2, \ldots, n+y\}$,
if there is one,
and sends (p, s) to $\#(s-n)$
through the mesh.

With random routing:
$y^{1/2+\epsilon}$ time, $y^{1+\epsilon}$ hardware.

Three-dimensional version

$y^{1/3+\epsilon}$ time, $y^{1+\epsilon}$ hardware.

Can use Batcher odd-even sort
to eliminate randomness and
achieve y^{ϵ} circuit depth.
But needs long wires.
$y^{1/3+\epsilon}$ time, $y^{1+\epsilon}$ hardware.

Conclusions

For sufficiently large y:

Don't trial divide (TWINKLE).
Don't sieve in memory.

NSA circuit is much faster.