# Generating random primes faster

D. J. Bernstein

---

pqRSA project team:

Daniel J. Bernstein
Josh Fried
Nadia Heninger
Paul Lou
Luke Valenta

cr.yp.to/papers.html#pqrsa



The standard algorithm
to generate random primes:

```
proof.arithmetic(False)
while True:
  p = randrange(2^(n-1),2^n)
  p = ZZ(p)
  if p.is_prime(): print p
```

$n^{1+o(1)}$ iterations per prime.

# Generating random primes faster

D. J. Bernstein

---

pqRSA project team:

Daniel J. Bernstein
Josh Fried
Nadia Heninger
Paul Lou
Luke Valenta

cr.yp.to/papers.html#pqrsa



The standard algorithm
to generate random primes:

```
proof.arithmetic(False)
while True:
  p = randrange(2^(n-1),2^n)
  p = ZZ(p)
  if p.is_prime(): print p
```

$n^{1+o(1)}$ iterations per prime.

Standard speedup using wheels:
e.g., force $p \bmod 6 \in \{1, 5\}$.
Wheel using all primes $q \le n^{O(1)}$:
$n^{1+o(1)}$ iterations per prime.
Recall $\prod_{q \le y} \left(1 - \frac{1}{q}\right) \in \Theta\left(\frac{1}{\log y}\right)$.

ng random primes faster

ernstein

project team:

. Bernstein
ed
eninger
u
lenta

to/papers.html#pqrsa



The standard algorithm
to generate random primes:

```
proof.arithmetic(False)
while True:
  p = randrange(2^(n-1),2^n)
  p = ZZ(p)
  if p.is_prime(): print p
```

$n^{1+o(1)}$ iterations per prime.

Standard speedup using wheels:
e.g., force $p \bmod 6 \in \{1,5\}$.
Wheel using all primes $q \le n^{O(1)}$:
$n^{1+o(1)}$ iterations per prime.
Recall $\prod_{q \le y} \left(1 - \frac{1}{q}\right) \in \Theta\left(\frac{1}{\log y}\right)$.

2007 Mi
$n^{3+o(1)}$

n primes faster

m:

n

s.html#pqrsa



The standard algorithm
to generate random primes:

```
proof.arithmetic(False)
while True:
  p = randrange(2^(n-1),2^n)
  p = ZZ(p)
  if p.is_prime(): print p
```

$n^{1+o(1)}$ iterations per prime.

Standard speedup using wheels:
e.g., force $p \bmod 6 \in \{1, 5\}$.
Wheel using all primes $q \le n^{O(1)}$:
$n^{1+o(1)}$ iterations per prime.
Recall $\prod_{q \le y} \left(1 - \frac{1}{q}\right) \in \Theta\left(\frac{1}{\log y}\right)$.

2007 Mihailescu: 
$n^{3+o(1)}$ bit ops to

aster

qrsa



The standard algorithm
to generate random primes:

```
proof.arithmetic(False)
while True:
  p = randrange(2^(n-1),2^n)
  p = ZZ(p)
  if p.is_prime(): print p
```

$n^{1+o(1)}$ iterations per prime.

Standard speedup using wheels:
e.g., force $p \bmod 6 \in \{1, 5\}$.
Wheel using all primes $q \le n^{O(1)}$:
$n^{1+o(1)}$ iterations per prime.
Recall $\prod_{q \le y} \left(1 - \frac{1}{q}\right) \in \Theta\left(\frac{1}{\log y}\right)$.



2007 Mihailescu: conjectura
$n^{3+o(1)}$ bit ops to prove $p$ p

The standard algorithm
to generate random primes:

```
proof.arithmetic(False)
while True:
  p = randrange(2^(n-1),2^n)
  p = ZZ(p)
  if p.is_prime(): print p
```

$n^{1+o(1)}$ iterations per prime.

Standard speedup using wheels:
e.g., force $p \bmod 6 \in \{1, 5\}$.
Wheel using all primes $q \le n^{O(1)}$:
$n^{1+o(1)}$ iterations per prime.
Recall $\prod_{q \le y} \left(1 - \frac{1}{q}\right) \in \Theta\left(\frac{1}{\log y}\right)$.



2007 Mihailescu: conjecturally
$n^{3+o(1)}$ bit ops to prove $p$ prime.

The standard algorithm
to generate random primes:

```
proof.arithmetic(False)
while True:
  p = randrange(2^(n-1),2^n)
  p = ZZ(p)
  if p.is_prime(): print p
```

$n^{1+o(1)}$ iterations per prime.

Standard speedup using wheels:
e.g., force $p \bmod 6 \in \{1,5\}$.
Wheel using all primes $q \le n^{O(1)}$:
$n^{1+o(1)}$ iterations per prime.
Recall $\prod_{q \le y}\left(1 - \frac{1}{q}\right) \in \Theta\left(\frac{1}{\log y}\right)$.



2007 Mihailescu: conjecturally
$n^{3+o(1)}$ bit ops to prove $p$ prime.

2010 Bernstein conjecture:
correctly recognize primality using
$n^{o(1)}$ tests, total $n^{2+o(1)}$ bit ops.
Fermat test, then Lucas test
(as in 1980 Baillie–Wagstaff, 1980
Pomerance–Selfridge–Wagstaff),
then cubic test (1995 Atkin), etc.;
or some elliptic-curve tests.

The standard algorithm
to generate random primes:

```
proof.arithmetic(False)
while True:
  p = randrange(2^(n-1),2^n)
  p = ZZ(p)
  if p.is_prime(): print p
```

$n^{1+o(1)}$ iterations per prime.

Standard speedup using wheels:
e.g., force $p \bmod 6 \in \{1, 5\}$.
Wheel using all primes $q \le n^{O(1)}$:
$n^{1+o(1)}$ iterations per prime.
Recall $\prod_{q \le y} (1 - \frac{1}{q}) \in \Theta(\frac{1}{\log y})$.



2007 Mihailescu: conjecturally
$n^{3+o(1)}$ bit ops to prove $p$ prime.

2010 Bernstein conjecture:
correctly recognize primality using
$n^{o(1)}$ tests, total $n^{2+o(1)}$ bit ops.
Fermat test, then Lucas test
(as in 1980 Baillie–Wagstaff, 1980
Pomerance–Selfridge–Wagstaff),
then cubic test (1995 Atkin), etc.;
or some elliptic-curve tests.

Most iterations are much simpler:
Fermat test rejects $p$.
Fast reject by trial division/ECM?
Still $n^{3+o(1)}$ bit ops per prime.

ndard algorithm
ate random primes:

```
rithmetic(False)
rue:
andrange(2^(n-1),2^n)
Z(p)
is_prime(): print p
```

iterations per prime.

d speedup using wheels:
ce $p$ mod $6 \in \{1, 5\}$.
sing all primes $q \leq n^{O(1)}$:
iterations per prime.
$\prod_{q \leq y} \left(1 - \frac{1}{q}\right) \in \Theta\left(\frac{1}{\log y}\right)$.



2007 Mihailescu: conjecturally $n^{3+o(1)}$ bit ops to prove $p$ prime.

2010 Bernstein conjecture: correctly recognize primality using $n^{o(1)}$ tests, total $n^{2+o(1)}$ bit ops. Fermat test, then Lucas test (as in 1980 Baillie–Wagstaff, 1980 Pomerance–Selfridge–Wagstaff), then cubic test (1995 Atkin), etc.; or some elliptic-curve tests.

Most iterations are much simpler: Fermat test rejects $p$.
Fast reject by trial division/ECM?
Still $n^{3+o(1)}$ bit ops per prime.

New: $n^2$
to gener

rithm

m primes:

```
(False)

2^(n-1),2^n)

): print p
```

per prime.

using wheels:

$6 \in \{1, 5\}$.

imes $q \leq n^{O(1)}$:

per prime.

$\frac{1}{q}) \in \Theta\left(\frac{1}{\log y}\right)$.



2007 Mihailescu: conjecturally $n^{3+o(1)}$ bit ops to prove $p$ prime.

2010 Bernstein conjecture: correctly recognize primality using $n^{o(1)}$ tests, total $n^{2+o(1)}$ bit ops. Fermat test, then Lucas test (as in 1980 Baillie–Wagstaff, 1980 Pomerance–Selfridge–Wagstaff), then cubic test (1995 Atkin), etc.; or some elliptic-curve tests.

Most iterations are much simpler: Fermat test rejects $p$. Fast reject by trial division/ECM? Still $n^{3+o(1)}$ bit ops per prime.



New: $n^{2.5+o(1)}$ bit

to generate $2^{n^{0.5+}}$

2^n)

p

eels:

$n^{O(1)}$:

$\frac{1}{\log y})$.



2007 Mihailescu: conjecturally $n^{3+o(1)}$ bit ops to prove $p$ prime.

2010 Bernstein conjecture: correctly recognize primality using $n^{o(1)}$ tests, total $n^{2+o(1)}$ bit ops. Fermat test, then Lucas test (as in 1980 Baillie–Wagstaff, 1980 Pomerance–Selfridge–Wagstaff), then cubic test (1995 Atkin), etc.; or some elliptic-curve tests.

Most iterations are much simpler: Fermat test rejects $p$. Fast reject by trial division/ECM? Still $n^{3+o(1)}$ bit ops per prime.



New: $n^{2.5+o(1)}$ bit ops per p
to generate $2^{n^{0.5+o(1)}}$ primes

2007 Mihailescu: conjecturally $n^{3+o(1)}$ bit ops to prove $p$ prime.

2010 Bernstein conjecture: correctly recognize primality using $n^{o(1)}$ tests, total $n^{2+o(1)}$ bit ops. Fermat test, then Lucas test (as in 1980 Baillie–Wagstaff, 1980 Pomerance–Selfridge–Wagstaff), then cubic test (1995 Atkin), etc.; or some elliptic-curve tests.

Most iterations are much simpler: Fermat test rejects $p$. Fast reject by trial division/ECM? Still $n^{3+o(1)}$ bit ops per prime.



New: $n^{2.5+o(1)}$ bit ops per prime to generate $2^{n^{0.5+o(1)}}$ primes.

2007 Mihailescu: conjecturally $n^{3+o(1)}$ bit ops to prove $p$ prime.

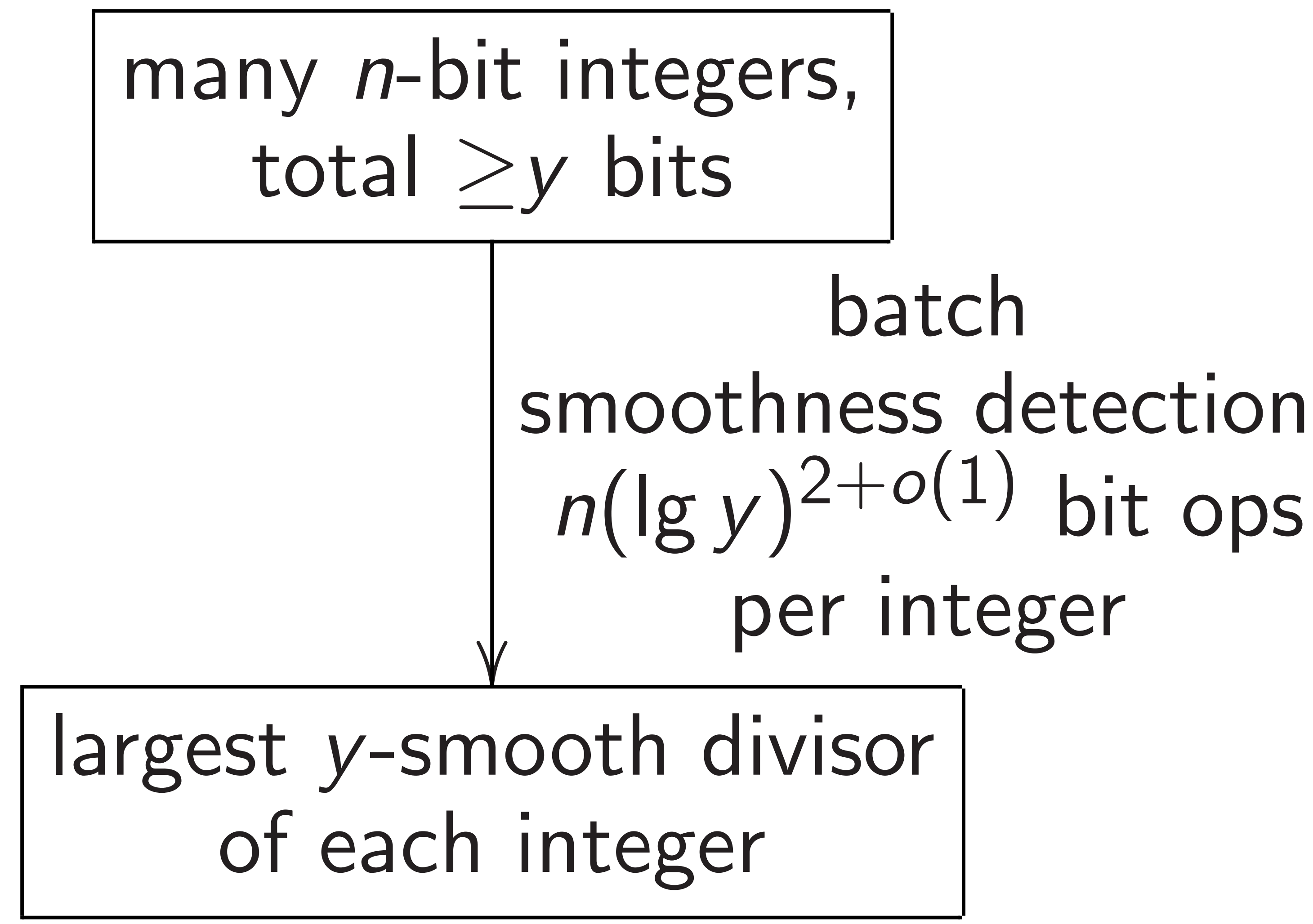
2010 Bernstein conjecture: correctly recognize primality using $n^{o(1)}$ tests, total $n^{2+o(1)}$ bit ops.
Fermat test, then Lucas test (as in 1980 Baillie–Wagstaff, 1980 Pomerance–Selfridge–Wagstaff), then cubic test (1995 Atkin), etc.; or some elliptic-curve tests.

Most iterations are much simpler:
Fermat test rejects $p$.
Fast reject by trial division/ECM?
Still $n^{3+o(1)}$ bit ops per prime.



New: $n^{2.5+o(1)}$ bit ops per prime to generate $2^{n^{0.5+o(1)}}$ primes.

Recall:



2007 Mihailescu: conjecturally $n^{3+o(1)}$ bit ops to prove $p$ prime.

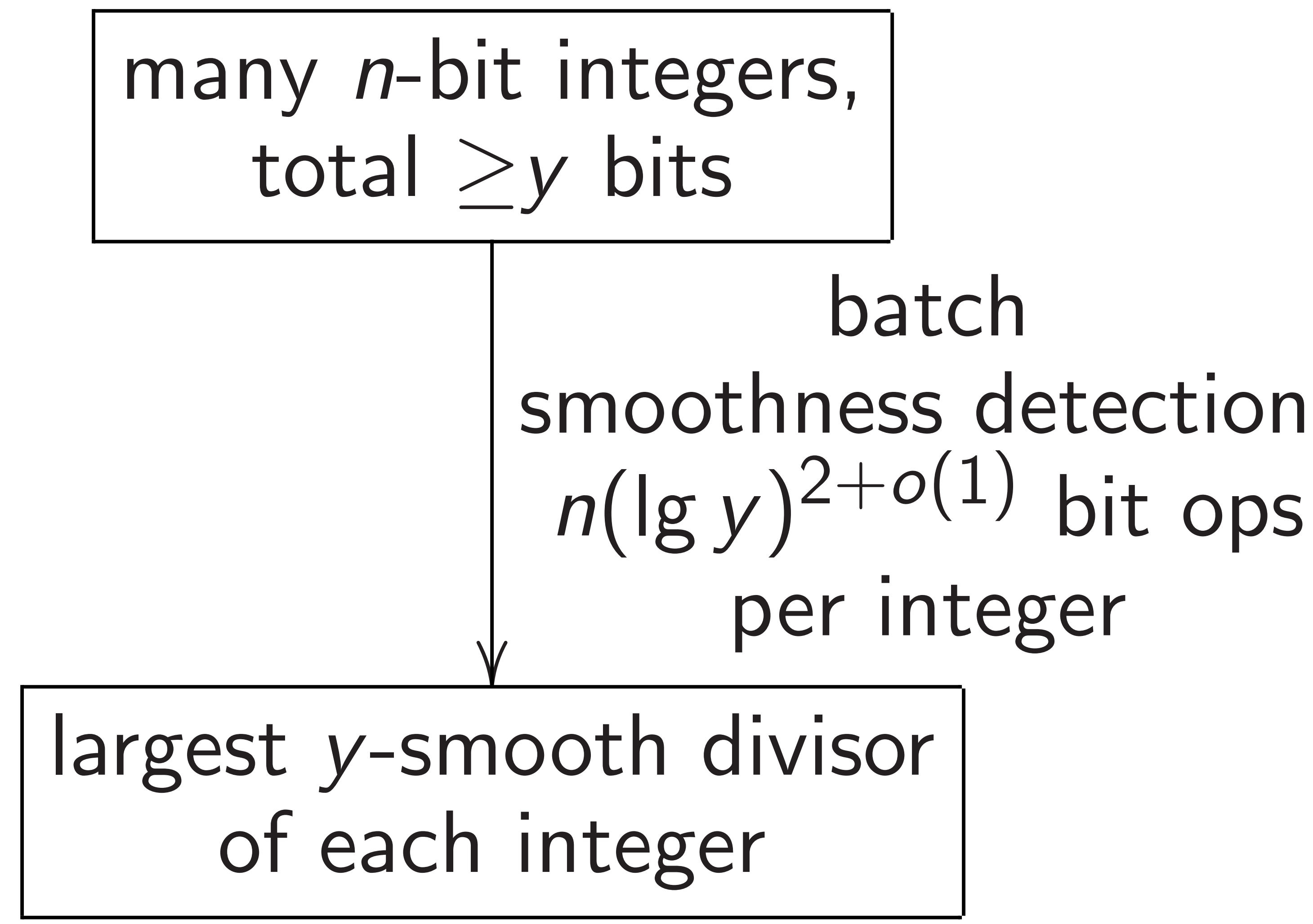
2010 Bernstein conjecture: correctly recognize primality using $n^{o(1)}$ tests, total $n^{2+o(1)}$ bit ops. Fermat test, then Lucas test (as in 1980 Baillie–Wagstaff, 1980 Pomerance–Selfridge–Wagstaff), then cubic test (1995 Atkin), etc.; or some elliptic-curve tests.

Most iterations are much simpler: Fermat test rejects $p$.
Fast reject by trial division/ECM?
Still $n^{3+o(1)}$ bit ops per prime.



New: $n^{2.5+o(1)}$ bit ops per prime to generate $2^{n^{0.5+o(1)}}$ primes.

Recall:




Apply batch smoothness detection for $y = 2^{2^0}$, then $y = 2^{2^1}$, then $y = 2^{2^2}$, . . . , then $y \approx 2^{n^{0.5+o(1)}}$.