Mersenne factorization factory

Thorsten Kleinjung
Arjen K. Lenstra & Joppe W. Bos*

École Polytechnique Fédérale de Lausanne
laboratory for cryptologic algorithms

Microsoft Research

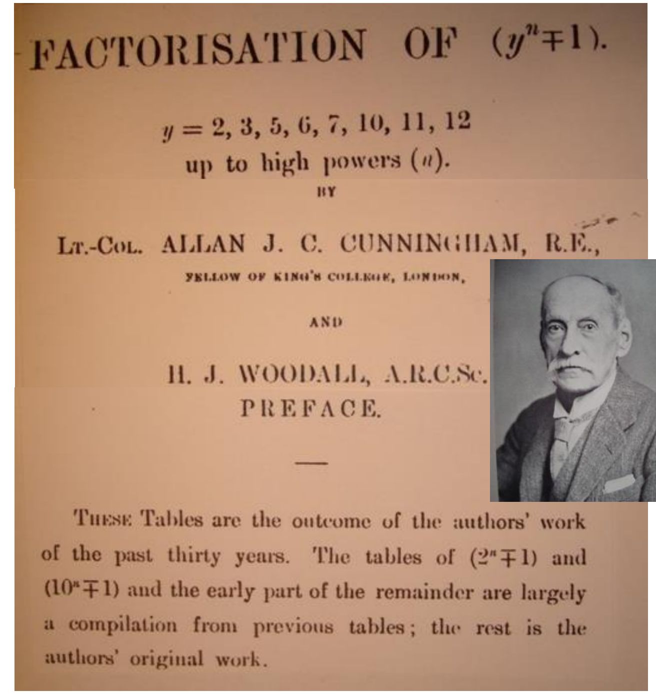
presented by: Rob Granger

* currently at NXP Semiconductors

Integer factorization: why?

traditionally:

mostly for recreational purposes



FACTORISATION OF $(y^n \mp 1)$.

$y = 2, 3, 5, 6, 7, 10, 11, 12$
up to high powers (n).

BY

Lt.-Col. ALLAN J. C. CUNNINGHAM, R.E.,
FELLOW OF KING'S COLLEGE, LONDON,

AND

H. J. WOODALL, A.R.C.Sc.

PREFACE.

These Tables are the outcome of the authors' work of the past thirty years. The tables of $(2^n \mp 1)$ and $(10^n \mp 1)$ and the early part of the remainder are largely a compilation from previous tables; the rest is the authors' original work.

more recently:

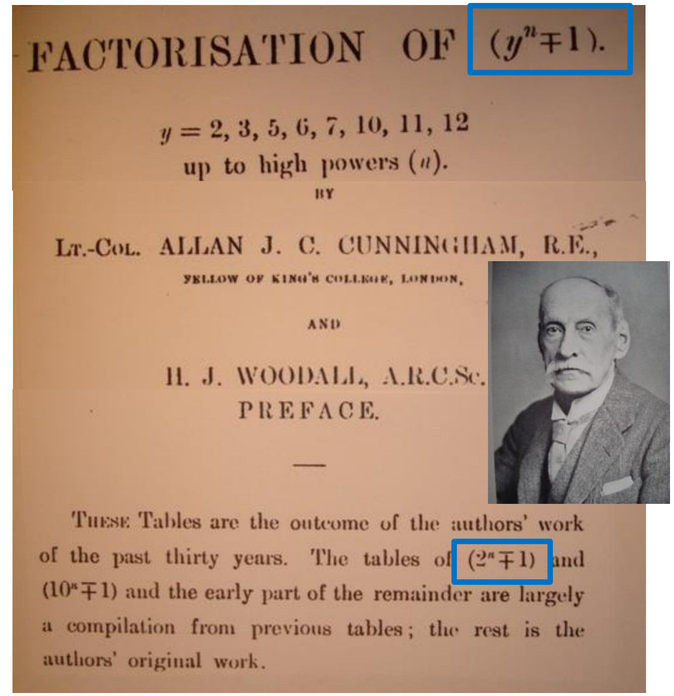
to assess the security of RSA

Integer factorization: why?

traditionally:

mostly for recreational purposes

- focus on “special” numbers



FACTORISATION OF $\boxed{(y^n \mp 1).}$

$y = 2, 3, 5, 6, 7, 10, 11, 12$

up to high powers (n).

BY

LT.-COL. ALLAN J. C. CUNNINGHAM, R.E.,

FELLOW OF KING'S COLLEGE, LONDON,

AND

H. J. WOODALL, A.R.C.Sc.

PREFACE.

THESE Tables are the outcome of the authors' work of the past thirty years. The tables of $\boxed{(2^n \mp 1)}$ and $(10^n \mp 1)$ and the early part of the remainder are largely a compilation from previous tables; the rest is the authors' original work.

more recently:

to assess the security of RSA

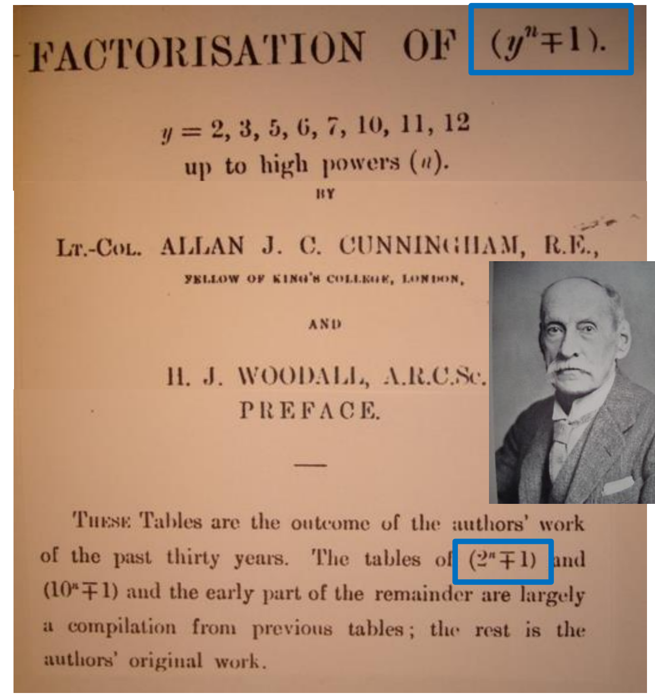
- focus on products of two large primes

Integer factorization: why?

traditionally:

mostly for recreational purposes

- focus on “special” numbers
- source of algorithmic inspiration


FACTORISATION OF $(y^n \mp 1)$.

$y = 2, 3, 5, 6, 7, 10, 11, 12$
up to high powers (n).

BY

LT.-COL. ALLAN J. C. CUNNINGHAM, R.E.,
FELLOW OF KING'S COLLEGE, LONDON,

AND

H. J. WOODALL, A.R.C.Sc.

PREFACE.

THESE Tables are the outcome of the authors' work of the past thirty years. The tables of $(2^n \mp 1)$ and $(10^n \mp 1)$ and the early part of the remainder are largely a compilation from previous tables; the rest is the authors' original work.

more recently:

to assess the security of RSA

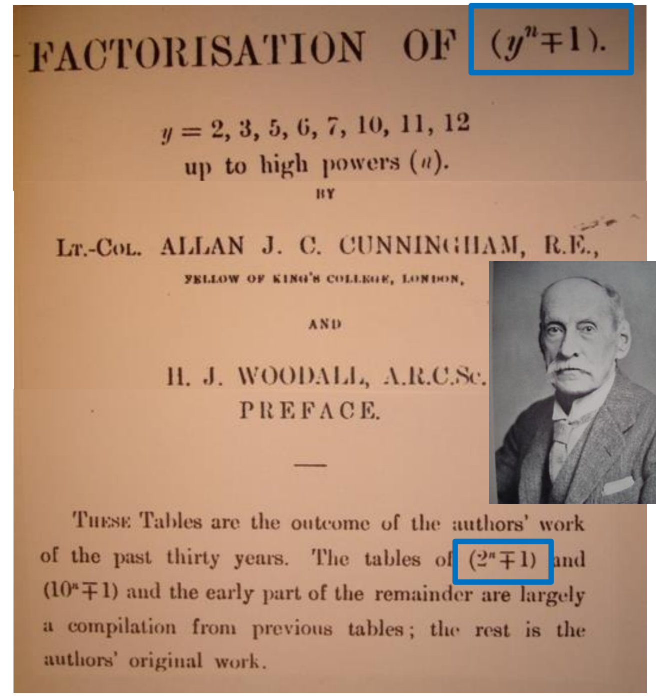
- focus on products of two large primes
- mostly applied/generalized “special” methods

Integer factorization: why?

traditionally:

mostly for recreational purposes

- focus on “special” numbers
- source of algorithmic inspiration
- currently believed to be easier:
  special number field sieve applies


FACTORISATION OF $(y^n \mp 1)$.

$y = 2, 3, 5, 6, 7, 10, 11, 12$

up to high powers (n).

BY

Lt.-Col. ALLAN J. C. CUNNINGHAM, R.E.,

FELLOW OF KING'S COLLEGE, LONDON,

AND

H. J. WOODALL, A.R.C.Sc.

PREFACE.

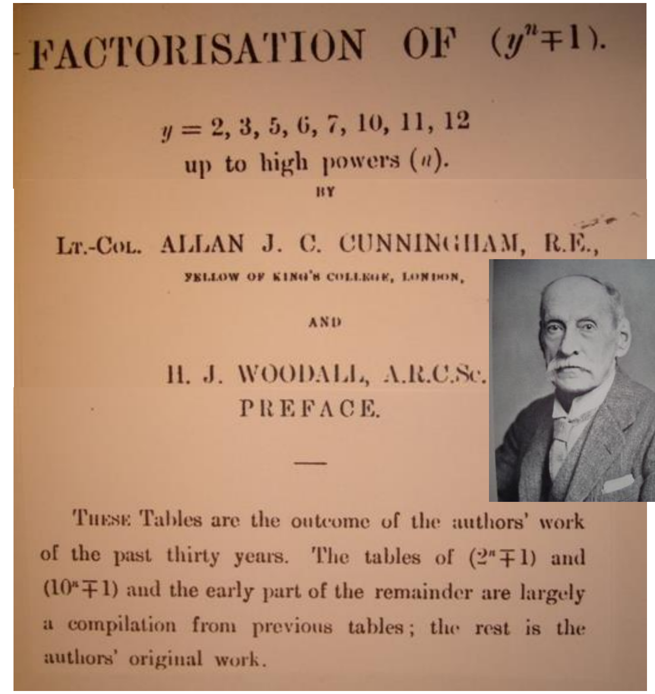
These Tables are the outcome of the authors' work of the past thirty years. The tables of $(2^n \mp 1)$ and $(10^n \mp 1)$ and the early part of the remainder are largely a compilation from previous tables; the rest is the authors' original work.

more recently:

to assess the security of RSA

- focus on products of two large primes
- mostly applied/generalized “special” methods
- currently believed to be the hardest case:
  nothing better published than number field sieve (NFS)

Integer factorization: why?


FACTORISATION OF $(y^n \mp 1)$.

$y = 2, 3, 5, 6, 7, 10, 11, 12$
up to high powers (n).

BY

LT.-COL. ALLAN J. C. CUNNINGHAM, R.E.,
FELLOW OF KING'S COLLEGE, LONDON,

AND

H. J. WOODALL, A.R.C.Sc.

PREFACE.

THESE Tables are the outcome of the authors' work of the past thirty years. The tables of $(2^n \mp 1)$ and $(10^n \mp 1)$ and the early part of the remainder are largely a compilation from previous tables; the rest is the authors' original work.

traditionally:

mostly for recreational purposes

- focus on “special” numbers
- source of algorithmic inspiration

more recently:

to assess the security of RSA

- focus on products of two large primes
- mostly applied/generalized “special” methods

exception: Coppersmith’s factorization factory

Integer factorization: why?

traditionally:

mostly for recreational purposes

- focus on “special” numbers
- source of algorithmic inspiration

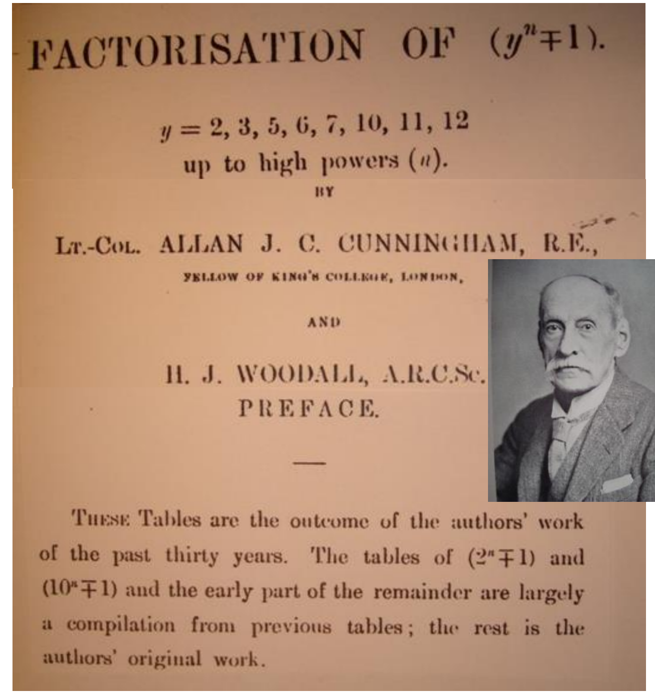
here: factorization factory
for “special” numbers

more recently:

to assess the security of RSA

- focus on products of two large primes
- mostly applied/generalized “special” methods

exception: Coppersmith’s factorization factory



FACTORISATION OF $(y^n \mp 1)$.

$y = 2, 3, 5, 6, 7, 10, 11, 12$
up to high powers (n).

BY

Lt.-Col. ALLAN J. C. CUNNINGHAM, R.E.,
FELLOW OF KING'S COLLEGE, LONDON,

AND

H. J. WOODALL, A.R.C.Sc.

PREFACE.

These Tables are the outcome of the authors' work of the past thirty years. The tables of $(2^n \mp 1)$ and $(10^n \mp 1)$ and the early part of the remainder are largely a compilation from previous tables; the rest is the authors' original work.

Why interested in “special” factoring factory?

factoring enthusiasts worldwide
are constantly burning cycles on “special” numbers

Why interested in “special” factoring factory?

factoring enthusiasts worldwide
are constantly burning cycles on “special” numbers

- can we factor these “special” numbers more efficiently by combining the effort à la Coppersmith?

Why interested in "special" factoring factory?

factoring enthusiasts worldwide
are constantly burning cycles on "special" numbers

- can we factor these "special" numbers more efficiently by combining the effort à la Coppersmith?

- potential side-effect: do we learn anything about effectiveness of Coppersmith's factorization factory to share the factoring effort of many RSA moduli?

Why interested in "special" factoring factory?

factoring enthusiasts worldwide
are constantly burning cycles on "special" numbers

- can we factor these "special" numbers more efficiently by combining the effort à la Coppersmith?

- potential side-effect: do we learn anything about effectiveness of Coppersmith's factorization factory to share the factoring effort of many RSA moduli?

attacking a single RSA modulus may not be economically viable, but factoring lots of them may become an attractive proposition if effort can be shared

our crypto salespitch

Bottom line

we can indeed effectively combine the factoring effort for well-chosen “special” numbers in current range of interest

Bottom line

we can indeed effectively combine the factoring effort for well-chosen "special" numbers in current range of interest

we saved about 50% on factorization of 17 Mersenne numbers in the 1000 to 1200 bit range

(largest so far $2^{1193}-1$; previous record $2^{1061}-1$)

Bottom line

we can indeed effectively combine the factoring effort for well-chosen “special” numbers in current range of interest

we saved about 50% on factorization of
17 Mersenne numbers in the 1000 to 1200 bit range

(largest so far $2^{1193}-1$; previous record $2^{1061}-1$)

potential for application to 1024-bit RSA is questionable

- if target moduli not known in advance: storage issues (cumbersome but surmountable for “special” numbers; but our target set was known in advance)

Bottom line

we can indeed effectively combine the factoring effort for well-chosen "special" numbers in current range of interest

we saved about 50% on factorization of 17 Mersenne numbers in the 1000 to 1200 bit range

(largest so far $2^{1193}-1$; previous record $2^{1061}-1$)

potential for application to 1024-bit RSA is questionable

- if target moduli not known in advance: storage issues (cumbersome but surmountable for "special" numbers; but our target set was known in advance)
- if target moduli known in advance: computational inefficiencies (that do not occur for "special" numbers) make regular "one-by-one" approach more efficient

Number field sieve applied to target composite n

1. polynomial selection: degree $d > 1$, integer $m \approx n^{1/d}$, radix m representation of $n = f_d m^d + \ldots + f_1 m + f_0$ leads to $f(X) = \Sigma_i f_i X^i \in \mathbf{Z}[X]$ with $f(m) \equiv 0 \bmod n$

Number field sieve applied to target composite n

1. polynomial selection: degree $d > 1$, integer $m \approx n^{1/d}$,
   radix m representation of $n = f_d m^d + \ldots + f_1 m + f_0$
   leads to $f(X) = \Sigma_i f_i X^i \in \mathbf{Z}[X]$ with $f(m) \equiv 0 \bmod n$

2. relation collection: coprime $a, b \in \mathbf{Z} \times \mathbf{Z}_{\geq 0}$ such that
   $(a-mb)(b^d f(a/b))$ factors into small primes ("smooth")

Number field sieve applied to target composite n

1. polynomial selection: degree $d > 1$, integer $m \approx n^{1/d}$, radix m representation of $n = f_d m^d + \ldots + f_1 m + f_0$ leads to $f(X) = \Sigma_i f_i X^i \in \mathbf{Z}[X]$ with $f(m) \equiv 0 \bmod n$

2. relation collection: coprime $a, b \in \mathbf{Z} \times \mathbf{Z}_{\geq 0}$ such that $(a - mb)(b^d f(a/b))$ factors into small primes ("smooth")

3. matrix step: even sum of small prime exponent vectors solves $x^2 \equiv 1 \bmod n$: $n = \gcd(x - 1, n) * \gcd(x + 1, n)$

Number field sieve applied to target composite n

1. polynomial selection: degree $d > 1$, integer $\boxed{m \approx n^{1/d}}$,
   radix m representation of $n = f_d m^d + \ldots + f_1 m + f_0$
   leads to $f(X) = \Sigma_i f_i X^i \in \mathbf{Z}[X]$ with $f(m) \equiv 0 \bmod n$
2. relation collection: coprime $a, b \in \mathbf{Z} \times \mathbf{Z}_{\geq 0}$ such that
   $(a-mb)(b^d f(a/b))$ factors into small primes ("smooth")
3. matrix step: even sum of small prime exponent vectors
   solves $x^2 \equiv 1 \bmod n$: $n = \gcd(x-1, n) * \gcd(x+1, n)$

Because $m \approx n^{1/d}$, single m works for many different n
(all with different polynomials f)

Number field sieve applied to target composite n

1. polynomial selection: degree $d > 1$, integer $\boxed{m \approx n^{1/d}}$,
   radix m representation of $n = f_d m^d + \ldots + f_1 m + f_0$
   leads to $f(X) = \Sigma_i f_i X^i \in \mathbf{Z}[X]$ with $f(m) \equiv 0 \bmod n$

2. relation collection: coprime $a, b \in \mathbf{Z} \times \mathbf{Z}_{\geq 0}$ such that
   $(a - mb)(b^d f(a/b))$ factors into small primes ("smooth")

3. matrix step: even sum of small prime exponent vectors
   solves $x^2 \equiv 1 \bmod n$: $n = \gcd(x - 1, n) * \gcd(x + 1, n)$

Because $m \approx n^{1/d}$, single m works for many different n
(all with different polynomials f)

Factorization factory idea: collect $a, b \in \mathbf{Z} \times \mathbf{Z}_{\geq 0}$ for which
$a - mb$ is smooth, use those pairs for many n and f

Number field sieve applied to target composite n

1. polynomial selection: degree $d > 1$, integer $\boxed{m \approx n^{1/d}}$, radix m representation of $n = f_d m^d + \ldots + f_1 m + f_0$ leads to $f(X) = \Sigma_i f_i X^i \in \mathbf{Z}[X]$ with $f(m) \equiv 0 \bmod n$
2. relation collection: coprime $a, b \in \mathbf{Z} \times \mathbf{Z}_{\geq 0}$ such that $(a - mb)(b^d f(a/b))$ factors into small primes ("smooth")
3. matrix step: even sum of small prime exponent vectors solves $x^2 \equiv 1 \bmod n$: $n = \gcd(x - 1, n) * \gcd(x + 1, n)$

Because $m \approx n^{1/d}$, single m works for many different n (all with different polynomials f)

Factorization factory idea: collect $a, b \in \mathbf{Z} \times \mathbf{Z}_{\geq 0}$ for which $a - mb$ is smooth, use those pairs for many n and f

if many n: amortization leads to overall savings

Factorization factory for L-bit moduli

1. $d > 1$, $m \approx 2^{L/d}$, collect $a, b \in \mathbf{Z} \times \mathbf{Z}_{\geq 0}$ s.t. $a - mb$ smooth
2. **any** $n \approx m^d$: $n = f_d m^d + \ldots + f_1 m + f_0 \Rightarrow f(m) \equiv 0 \bmod n$, locate smooth $b^d f(a/b)$, and apply matrix step as usual

Factorization factory for L-bit moduli

1. $d > 1$, $m \approx 2^{L/d}$, collect $a, b \in \mathbf{Z} \times \mathbf{Z}_{\geq 0}$ s.t. $a - mb$ smooth
2. **any** $n \approx m^d$: $n = f_d m^d + \ldots + f_1 m + f_0 \Rightarrow f(m) \equiv 0 \bmod n$, locate smooth $b^d f(a/b)$, and apply matrix step as usual

"special" numbers: does same shared-m trick apply?

Factorization factory for L-bit moduli

1. $d > 1$, $m \approx 2^{L/d}$, collect $a, b \in \mathbf{Z} \times \mathbf{Z}_{\geq 0}$ s.t. $a - mb$ smooth
2. **any** $n \approx m^d$: $n = f_d m^d + \ldots + f_1 m + f_0 \Rightarrow f(m) \equiv 0 \bmod n$, locate smooth $b^d f(a/b)$, and apply matrix step as usual

"special" numbers: does same shared-m trick apply?

"special" numbers equivalent to "nice" polynomials

(example: $F_9 \mid (2^{103})^5 + 8 \Rightarrow f(X) = X^5 + 8$ has $f(2^{103}) \equiv 0 \bmod F_9$),

thus much higher $b^d f(a/b)$ smoothness probability

Factorization factory for L-bit moduli

1. $d > 1$, $m \approx 2^{L/d}$, collect $a, b \in \mathbf{Z}\times\mathbf{Z}_{\geq 0}$ s.t. $a-mb$ smooth
2. **any** $n \approx m^d$: $n = f_d m^d + \ldots + f_1 m + f_0 \Rightarrow f(m) \equiv 0 \bmod n$, locate smooth $b^d f(a/b)$, and apply matrix step as usual

"special" numbers: does same shared-m trick apply?

"special" numbers equivalent to "nice" polynomials
(example: $F_9 | (2^{103})^5+8 \Rightarrow f(X) = X^5+8$ has $f(2^{103}) \equiv 0 \bmod F_9$),
thus much higher $b^d f(a/b)$ smoothness probability

relevant "special" numbers (such as Cunningham numbers):

- no shared m-value with enough "nice" polynomials

Factorization factory for L-bit moduli

1. $d > 1$, $m \approx 2^{L/d}$, collect $a, b \in \mathbf{Z} \times \mathbf{Z}_{\geq 0}$ s.t. $a - mb$ smooth
2. **any** $n \approx m^d$: $n = f_d m^d + \ldots + f_1 m + f_0 \Rightarrow f(m) \equiv 0 \bmod n$,
   locate smooth $b^d f(a/b)$, and apply matrix step as usual

"special" numbers: does same shared-m trick apply?

"special" numbers equivalent to "nice" polynomials
(example: $F_9 | (2^{103})^5 + 8 \Rightarrow f(X) = X^5 + 8$ has $f(2^{103}) \equiv 0 \bmod F_9$),
thus much higher $b^d f(a/b)$ smoothness probability

relevant "special" numbers (such as Cunningham numbers):
- no shared m-value with enough "nice" polynomials

$\Rightarrow$ trick does not apply without losing "special" advantage

Factorization factory for L-bit moduli

1. $d > 1$, $m \approx 2^{L/d}$, collect $a, b \in \mathbf{Z}\times\mathbf{Z}_{\geq 0}$ s.t. $a-mb$ smooth
2. **any** $n \approx m^d$: $n = f_d m^d + \ldots + f_1 m + f_0 \Rightarrow f(m) \equiv 0 \bmod n$, locate smooth $b^d f(a/b)$, and apply matrix step as usual

"special" numbers: does same shared-m trick apply?

"special" numbers equivalent to "nice" polynomials

(example: $F_9 | (2^{103})^5+8 \Rightarrow f(X) = X^5+8$ has $f(2^{103}) \equiv 0 \bmod F_9$),
thus much higher $b^d f(a/b)$ smoothness probability

relevant "special" numbers (such as Cunningham numbers):

- no shared m-value with enough "nice" polynomials

$\Rightarrow$ trick does not apply without losing "special" advantage

unless we reverse the roles of $a-mb$ and $b^d f(a/b)$:
single f may cater to several m-values (and thus n-values)

“Special” factorization factory

1. for “nice” f, collect a, $b \in \mathbf{Z} \times \mathbf{Z}_{\geq 0}$ s.t. $b^d f(a/b)$ smooth
2. **any** m for which $n = f(m)$ is relevant:
   locate smooth $a - mb$, and apply matrix step as usual

“Special” factorization factory

1. for “nice” f, collect $a, b \in \mathbf{Z}\times\mathbf{Z}_{\geq 0}$ s.t. $b^d f(a/b)$ smooth
2. **any** m for which $n = f(m)$ is relevant:
   locate smooth $a-mb$, and apply matrix step as usual

here applied to factor Mersenne numbers (2^k-1):

“Special” factorization factory

1. for “nice” f, collect $a, b \in \mathbf{Z} \times \mathbf{Z}_{\geq 0}$ s.t. $b^d f(a/b)$ smooth
2. **any** m for which $n = f(m)$ is relevant:
   locate smooth $a – mb$, and apply matrix step as usual

here applied to factor Mersenne numbers ($2^k – 1$):

$f(X) = X^8 – 2$ leads to 11 + 1 relevant composites:

- $2^{1007} – 1$ for $m = 2^{126}$
- $2^{1009} – 1$ for $m = 2^{–126}$
- $2^{1081} – 1$ for $m = 2^{–135}$

…

- $2^{1193} – 1$ for $m = 2^{–149}$
- $2^{1199} – 1$ for $m = 2^{150}$

“Special” factorization factory

1. for “nice” f, collect $a, b \in \mathbf{Z} \times \mathbf{Z}_{\geq 0}$ s.t. $b^d f(a/b)$ smooth
2. **any** m for which $n = f(m)$ is relevant:
   locate smooth $a - mb$, and apply matrix step as usual

here applied to factor Mersenne numbers ($2^k - 1$):

$f(X) = X^8 - 2$ leads to 11 + 1 relevant composites:

- $2^{1007} - 1$ for $m = 2^{126}$
- $2^{1009} - 1$ for $m = 2^{-126}$
- $2^{1081} - 1$ for $m = 2^{-135}$

…

- $2^{1193} - 1$ for $m = 2^{-149}$
- $2^{1199} - 1$ for $m = 2^{150}$

$f(X) = X^8 - 8$ leads to 6 + 7 relevant composites

“Special” factorization factory

1. for “nice” f, collect $a, b \in \mathbf{Z} \times \mathbf{Z}_{\geq 0}$ s.t. $b^d f(a/b)$ smooth
2. **any** m for which $n = f(m)$ is relevant:
   locate smooth $a - mb$, and apply matrix step as usual

here applied to factor Mersenne numbers ($2^k - 1$):

$f(X) = X^8 - 2$ leads to 11 + 1 relevant composites:

- $2^{1007} - 1$ for $m = 2^{126}$
- $2^{1009} - 1$ for $m = 2^{-126}$
- $2^{1081} - 1$ for $m = 2^{-135}$

…

- $2^{1193} - 1$ for $m = 2^{-149}$
- $2^{1199} - 1$ for $m = 2^{150}$

$f(X) = X^8 - 8$ leads to 6 + 7 relevant composites

(1+7 factored by ECM – 10+3 of 11+6 factored so far)

Nuts & bolts

1. for "nice" f, collect $a, b \in \mathbf{Z} \times \mathbf{Z}_{\geq 0}$ s.t. $b^d f(a/b)$ smooth
2. **any** m for which $n = f(m)$ is relevant:
   locate smooth $a - mb$, and apply matrix step as usual

Nuts & bolts

1. for "nice" f, collect $a, b \in \mathbf{Z} \times \mathbf{Z}_{\geq 0}$ s.t. $b^d f(a/b)$ smooth
2. **any** m for which $n = f(m)$ is relevant:
   locate smooth $a-mb$, and apply matrix step as usual

storage of Step 1 (a,b) pairs would require 70TB

$\Rightarrow$ right after generating a batch of Step 1 (a,b) pairs,
we processed it for all m-values, deleting useless pairs

Nuts & bolts

1. for "nice" f, collect $a, b \in \mathbf{Z} \times \mathbf{Z}_{\geq 0}$ s.t. $b^d f(a/b)$ smooth
2. **any** m for which $n = f(m)$ is relevant:
   locate smooth $a - mb$, and apply matrix step as usual

storage of Step 1 (a,b) pairs would require 70TB
$\Rightarrow$ right after generating a batch of Step 1 (a,b) pairs,
we processed it for all m-values, deleting useless pairs

smooth $b^d f(a/b)$-values collected using usual lattice sieving
smooth $a - mb$-values located using factorization trees
because sieving is – as predicted – too inefficient
different for regular 1024-bit factorization factory?

Nuts & bolts

1. for “nice” f, collect $a, b \in \mathbf{Z} \times \mathbf{Z}_{\geq 0}$ s.t. $b^d f(a/b)$ smooth
2. **any** m for which $n = f(m)$ is relevant:
   locate smooth $a–mb$, and apply matrix step as usual

storage of Step 1 (a,b) pairs would require 70TB
$\Rightarrow$ right after generating a batch of Step 1 (a,b) pairs,
we processed it for all m-values, deleting useless pairs

smooth $b^d f(a/b)$-values collected using usual lattice sieving
smooth $a–mb$-values located using factorization trees
because sieving is – as predicted – too inefficient
different for regular 1024-bit factorization factory?

for two numbers: additional sieving with other polynomials

Nuts & bolts

1. for "nice" f, collect $a, b \in \mathbf{Z} \times \mathbf{Z}_{\geq 0}$ s.t. $b^d f(a/b)$ smooth
2. **any** m for which $n = f(m)$ is relevant:
 locate smooth $a–mb$, and apply matrix step as usual

storage of Step 1 (a,b) pairs would require 70TB

$\Rightarrow$ right after generating a batch of Step 1 (a,b) pairs,
 we processed it for all m-values, deleting useless pairs

smooth $b^d f(a/b)$-values collected using usual lattice sieving
 smooth $a–mb$-values located using factorization trees
 because sieving is – as predicted – too inefficient
 different for regular 1024-bit factorization factory?

for two numbers: additional sieving with other polynomials

most matrix steps **not** "as usual" but with "double product"

Time spent

(after initial ECM effort – reported elsewhere)

Time spent

(after initial ECM effort – reported elsewhere)

$b^d f(a/b)$-sieving and a–mb-factorization trees

from May 22, 2010, to September 11, 2014:

about 3665+2065 = 5730 core years (2.2GHz)

Time spent

(after initial ECM effort – reported elsewhere)

$b^d f(a/b)$-sieving and $a–mb$-factorization trees
from May 22, 2010, to September 11, 2014:
about 3665+2065 = 5730 core years (2.2GHz)

matrices started December 7, 2012, four still crunching:
about 1250+ 500 = 1750 core years expected

Time spent

(after initial ECM effort – reported elsewhere)

$b^d f(a/b)$-sieving and $a–mb$-factorization trees
from May 22, 2010, to September 11, 2014:
about 3665+2065 = 5730 core years (2.2GHz)

matrices started December 7, 2012, four still crunching:
about 1250+ 500 = 1750 core years expected

largest matrix 297’605’781 × 297’606’805
with 81’028 million non-zero entries

Time spent

(after initial ECM effort – reported elsewhere)

$b^d f(a/b)$-sieving and $a\text{–}mb$-factorization trees
from May 22, 2010, to September 11, 2014:
about 3665+2065 = 5730 core years (2.2GHz)

matrices started December 7, 2012, four still crunching:
about 1250+ 500 = 1750 core years expected

largest matrix 297’605’781 × 297’606’805
with 81’028 million non-zero entries

both relation collection and matrix step
surpass RSA-768 effort by a factor of about 4

Time spent

(after initial ECM effort – reported elsewhere)

$b^d f(a/b)$-sieving and a–mb-factorization trees
from May 22, 2010, to September 11, 2014:
about 3665+2065 = 5730 core years (2.2GHz)

matrices started December 7, 2012, four still crunching:
about 1250+ 500 = 1750 core years expected

largest matrix 297’605’781 × 297’606’805
with 81’028 million non-zero entries

both relation collection and matrix step
surpass RSA-768 effort by a factor of about 4

total 7500 core years (86.7% EPFL, 12.8% MSR, 0.5% CH grid), about half of
estimated 14’000 - 21’000 core years’ individual effort

Time saved, in theory, all "heuristic expected"

$$L(c) = \exp((c+o(1))(\log(n))^{1/3}(\log(\log(n)))^{2/3}), \quad n \to \infty$$

Time saved, in theory, all "heuristic expected"

$L(c) = \exp((c+o(1))(\log(n))^{1/3}(\log(\log(n)))^{2/3})$, $n \to \infty$

NFS: factors n in time $L((64/9)^{1/3}) \approx L(1.923)$

SNFS: factors "special" n in time $L((32/9)^{1/3}) \approx L(1.526)$

Time saved, in theory, all "heuristic expected"

$L(c) = \exp((c+o(1))(\log(n))^{1/3}(\log(\log(n)))^{2/3})$, $n \to \infty$

NFS: factors n in time $L((64/9)^{1/3}) \approx L(1.923)$

SNFS: factors "special" n in time $L((32/9)^{1/3}) \approx L(1.526)$

Coppersmith factorization factory:

- after $L(2.007)$ preparation, factor n in $L(1.639)$
- advantageous if $> L(2.007-1.923) = L(0.084)$ distinct n values of about the same size must be factored

Time saved, in theory, all "heuristic expected"

$L(c) = \exp((c+o(1))(\log(n))^{1/3}(\log(\log(n)))^{2/3})$, $n \to \infty$

NFS: factors n in time $L((64/9)^{1/3}) \approx L(1.923)$
SNFS: factors "special" n in time $L((32/9)^{1/3}) \approx L(1.526)$

Coppersmith factorization factory:

- after $L(2.007)$ preparation, factor n in $L(1.639)$
- advantageous if $> L(2.007-1.923) = L(0.084)$ distinct n values of about the same size must be factored

"Special" factorization factory:

- after $L(1.615)$ preparation, factor n in $L(1.211)$
- advantageous if $> L(1.615-1.526) = L(0.089)$ distinct "special" (and suitable) n values must be factored

Time saved, in theory, all "heuristic expected"

$L(c) = \exp((c+o(1))(\log(n))^{1/3}(\log(\log(n)))^{2/3}), \; n \to \infty$

NFS: factors n in time $L((64/9)^{1/3}) \approx L(1.923)$
SNFS: factors "special" n in time $L((32/9)^{1/3}) \approx L(1.526)$

Coppersmith factorization factory:

- after $L(2.007)$ preparation, factor n in $L(1.639)$
- advantageous if $> L(2.007-1.923) = L(0.084)$ distinct n values of about the same size must be factored

"Special" factorization factory:

- after $L(1.615)$ preparation, factor n in $L(1.211)$
- advantageous if $> L(1.615-1.526) = L(0.089)$ distinct "special" (and suitable) n values must be factored
- more such n values: individual time reduces to $L(0.763)$ while preparation time $\to \infty$ (before amortization)

Conclusion

- showed practical value of Coppersmith's factorization
  factory for known set of "special" numbers
- current implementations of various smoothness tests
  suggest that new ideas are needed for
  a 1024-bit RSA factorization factory
  (for larger RSA moduli it may already be worthwhile)

Conclusion

- showed practical value of Coppersmith's factorization
  factory for known set of "special" numbers
- current implementations of various smoothness tests
  suggest that new ideas are needed for
  a 1024-bit RSA factorization factory
  (for larger RSA moduli it may already be worthwhile)
- 10011000001101011000000011110010001111011111110000000100101110000111101010010100010100001011010100001110001000100
  10111001011111111011100111000001101100001010111010110111010101000101101001110100101001001001011111101000101110 0110
  111101001111110110000100000010001110010101001000110100010110000000001101000110010011000111000010000000111000010000
  0011001
  is a 346-bit prime that divides the 1193-bit number
  1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
  1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
  1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
  1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
  1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
  1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
  1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
  1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
  1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
  1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
  1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
  111111111111111111111111111111111111111111111111111111111111
  (this is the current special number field sieve record)