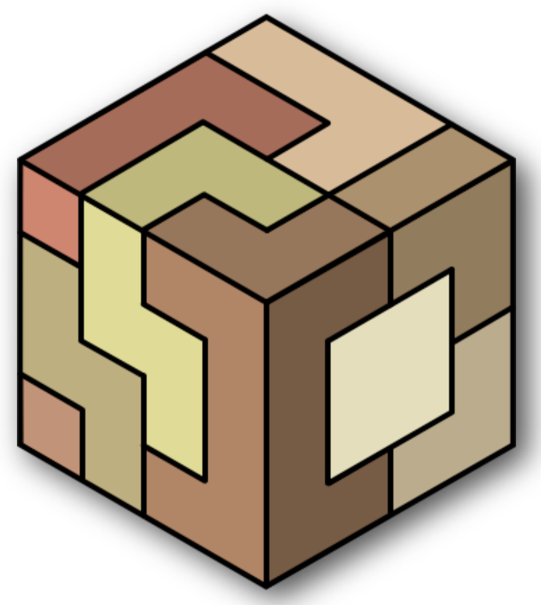
An overview of integer factorization

From the dark ages to the modern times



jerome.milan (at) lix.polytechnique.fr

March 2010

The Dark Ages

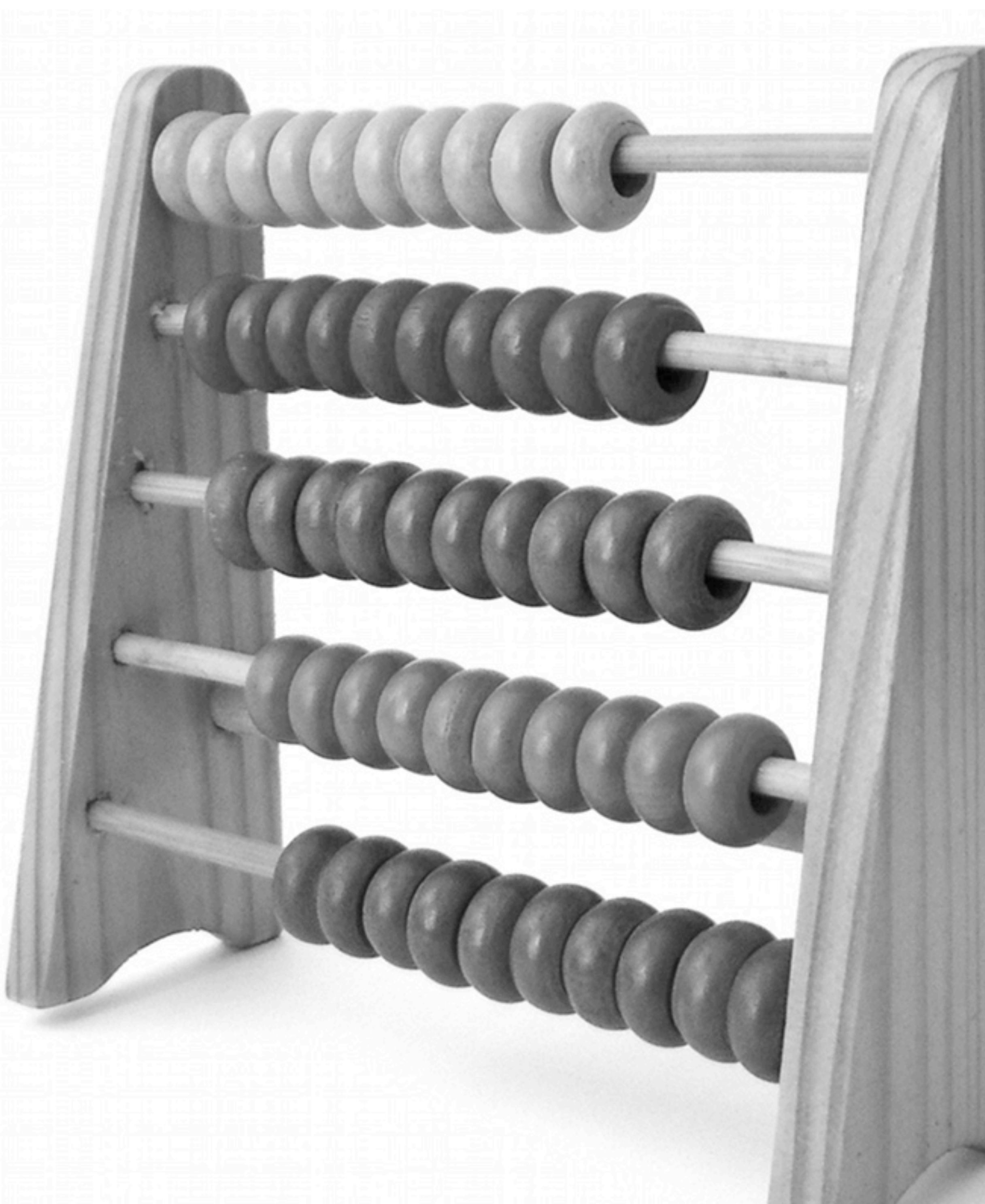
Fermat's method

The p−1 method

The p+1 method

Pollard's Rho

SQUFOF



Fermat's method

- Fermat – Around 1643, in a letter to Mersenne
- Write $N = p_1 p_2$ as $N = x^2 - y^2 = (x+y)(x-y)$
  - If N is not a square then $x \geq \lfloor \sqrt{N} \rfloor + 1$

Fermat's method (simplest form)

Input: integer N to factor
Output: a factor p of N

1. $m \leftarrow \lfloor \sqrt{N} \rfloor + 1$

2. while (true)

  if $m^2 - N$ is a square **then**

    return $m + \sqrt{m^2 - N}$

  else

    $m \leftarrow m + 1$

Fermat's method

- Basic enhancements
  - Replace squarings by additions

$$(m+1)^2 - N = m^2 - N + (2m+1)$$

  - Better square detection test (*e.g.* a square $\equiv_{16} 0, 1, 4$ or 9)
    - Deduce sieve on x

- Runs in $O\left(\dfrac{(\sqrt{N}-p_1)^2}{2p_1}\right)$ with best case in $O(\sqrt{N})$

- Latter enhancements
  - R. Lehman (1974) in $O(N^{1/3})$
  - J. McKee (1999) heuristically in $O(N^{1/4})$
  - R. Erra/C. Grenier (2009) in polynomial time if $|p_1 - p_2| < N^{1/3}$

The p−1 method

- Published by Pollard in 1974 (previously known by D.N & D.H Lehmer)

- Based on Fermat's little theorem
  - If $\gcd(x, p) = 1$ then $x^{p-1} = 1 \bmod p$

- Let $y = \prod_{i=0}^{r} p_i^{e_i}$ and $B \in \mathbb{N}$
  - y is B-smooth $\equiv \forall i \in [0, r], p_i \leq B$
  - y is B-power smooth $\equiv \forall i \in [0, r], p_i^{e_i} \leq B$

- A special-purpose algorithm
  - Succeeds if a $p_i - 1$ is B-power smooth, for some bound B
  - Runs in $O(B \cdot \log B \cdot \log^2 N)$

The p−1 method

Pollard's p−1 (first stage)

Input: integer N to factor
bound B_1

Output: a factor p of N or failure

1. Choose x coprime with N
2. **for** $i = 1..B_1$ **do** // *Compute* $x^{B_1!} \bmod N$
   - $x \leftarrow x^i \bmod N$
3. $p \leftarrow \gcd(x-1, N)$
4. **if** $(p \neq 1)$ and $(p \neq N)$
   - **return** p

   else
   - **return** failure

The p−1 method

- First stage example
  - $N = 421 \times 523$
  - $B = 7 \qquad x = 3$
  - $\gcd(x^{B!} - 1, N) = 421$
  - $420 = 2^2 \times 3 \times 5 \times 7$ (420 is 7-power smooth)

- Optional second stage
  - If $p - 1$ not B_1-power smooth, first stage will fail
  - Second stage allows one factor of $p - 1$ to be in $[B_1, B_2]$
    - Compute $\gcd((x^{B!})^q - 1, N)$ for all primes q in $[B_1, B_2]$
      - Standard continuation, FFT continuation, etc.

The p−1 method

Pollard's p−1 (second stage – standard continuation)

Input: integer N to factor
$y = x^{B_1!} \bmod N$ from first stage
bound B_2

Output: a factor p of N or failure

1. [Precomputations]

Let $\{q_1, q_2 \ldots q_k\}$ be the primes in $[B_1, B_2]$
$y_i \leftarrow y^{q_{i+1}-q_i} \bmod N$ for all $i \in [1, k]$

2. [Gcds]

$z \leftarrow y_1^{q_1} \bmod N$
for $j = 1..k$ **do**
  $p \leftarrow \gcd(z-1, N)$
  if $(p \neq 1)$ and $(p \neq N)$ **return** p
  $z \leftarrow z \times y_i \bmod N$
return failure

The p+1 method

- H.C. Williams – 1982
  - Similar to p−1 but succeeds if p+1 is B_1-power smooth
- Suppose $p = \prod_{i=1}^{k} q_i^{e_i} - 1$
- Lucas sequence $V_k(P,Q)$
  - Let α,β roots of $x^2 - Px + Q$ and $\Delta = P^2 - 4Q$
    - $V_k(P,Q) \equiv \alpha^k + \beta^k$
  - Fact 1. If $(\gcd(\Delta, N) = 1)$ and $((\Delta/p) = -1)$ then
    p divides $V_{B_1!}(P,Q) - 2$
  - Fact 2. There is an efficient algorithm to compute $V_k(P,1)$
    using recursive formulae

The p+1 method

Williams' p+1 (first stage)

Input: integer N to factor
bound B_1

Output: a factor p of N or failure

1. Choose P_0 so that $\gcd(P_0^2 - 4, N) = 1$
2. $P_m \leftarrow V_{B_1!}(P_0, 1)$ *// There's an efficient algo for that*
   // using recursion formulae
3. $p \leftarrow gcd(N, P_m - 2)$
4. **if** $(p \neq 1)$ and $(p \neq N)$
   return p
   else
   return failure

The p+1 method

- What if $(\Delta/p) \neq -1$?
  - Degrades as a slow version of Pollard's p−1
  - Failure – retry with another P_0

- Second stage with bound B_2
  - Will work if $p = p_L \prod_{i=1}^{k} q_i^{e_i} - 1$ with p_L prime in $[B_1, B_2]$
  - Similar to Pollard's p−1 but computes gcd(N, T_j) where T_j is a combination of Lucas sequences

- In practice, slower than p−1

Pollard's Rho

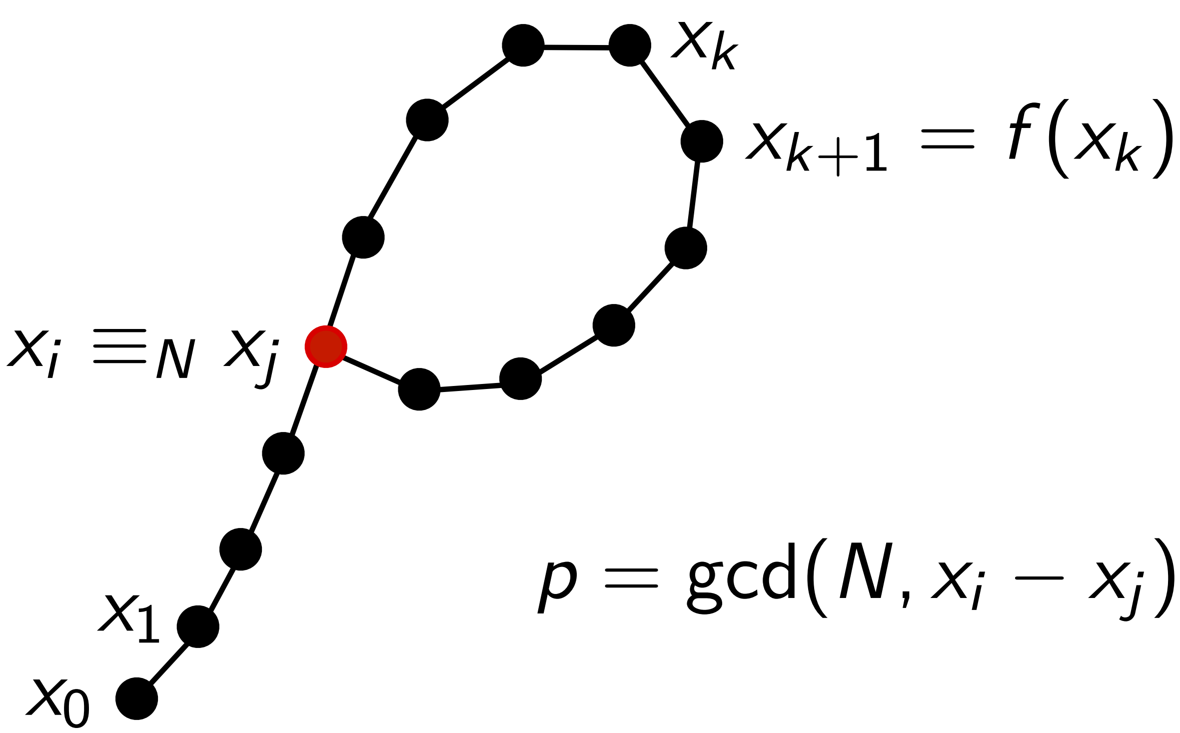
- John Pollard – 1975

- Special-purpose algorithm
  - Better when N has small factors
  - Complexity: $O\left(\sqrt{p}\right)$ with p a factor of N

- Based on birthday paradox
  - Randomly pick $x_1, x_2, x_3 \ldots$ in $[0, N]$
  - Collision expected after $\simeq \sqrt{\pi N/2}$ samples

Pollard's Rho

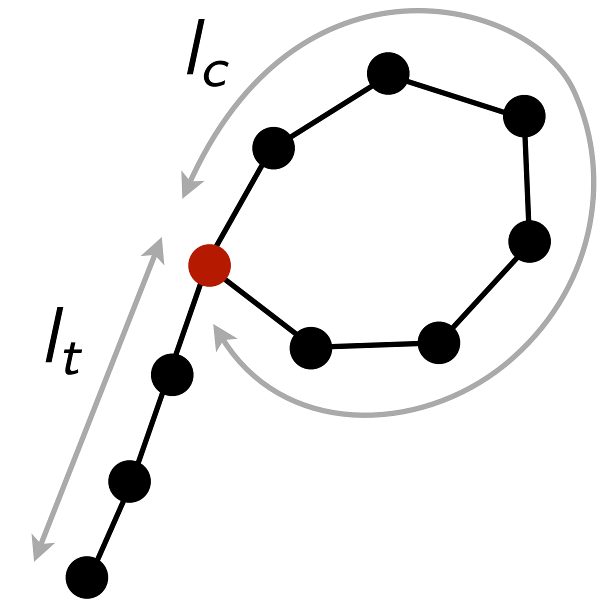
- Idea – Find self-collision in pseudo random walk $f \bmod N$



- Suppose $N = p_1 p_2 \ldots p_k$
  - If $x_i = x_j \bmod p$ then $\gcd(N, x_i - x_j)$ may give a factor
  - Collision expected after $O(\sqrt{p_1})$ iterations only

Pollard's Rho

- Floyd's cycle finding algorithm
  - Only compare x_i and x_{2i}
  - $x_{2i} = x_i \iff l_c | i$ and $i \geq l_t$
  - $\exists i, l_t \leq i < l_t + l_c$ such that $x_{2i} = x_i$



- Variants – Brent, Nivasch, distinguished points, etc.

- Open question – which function f?
  - Usually, $f(x) = ax^2 + b \mod N$

Pollard's Rho

Pollard's Rho (with Floyd's cycle finding)

Input: integer N to factor
pseudo random walk function f

Output: a factor p of N

1. [init]

$x_i \leftarrow 1$

$x_{2i} \leftarrow 1$

2. while $(g = 1)$ or $(g = N)$

$x_i \leftarrow f(x_i)$

$x_{2i} \leftarrow f(f(x_{2i}))$

$g \leftarrow \gcd(N, x_{2i} - x_i)$

3. return g

SQUFOF – Square Form Factorization

- D. Shanks – around 1975
  - Discovered while investigating CFRAC's shortcomings
- Based on infrastructure of real quadratic fields
- Quadratic forms $F(x, y) = ax^2 + bxy + cy^2 \equiv (a, b, c)$
  - ρ standard reduction operator
    - Expressed with continued fraction formalism
- The forms $(a_i, b_i, c_i) = \rho^i(a_0, b_0, c_0)$ are on a cycle
  - Look for (a_i, b_i, c_i) and $(a_{i+1}, b_{i+1}, c_{i+1})$ with $b_i = b_{i+1}$
    - Yields simple relation giving a factor of $\Delta = b^2 - 4ac$

SQUFOF – Square Form Factorization

$$q_0 = \lfloor N \rfloor \quad , \quad q_i = \left\lfloor \frac{q_0 + P_i}{Q_i} \right\rfloor \text{ for } i > 0 \tag{1}$$

$$P_0 = 0 \quad , \quad P_1 = q_0 \tag{2}$$

$$P_i = q_{i-1}Q_{i-1} - P_{i-1} \text{ for } i > 1 \tag{3}$$

$$Q_0 = 1 \quad , \quad Q_1 = N - q_0^2 \tag{4}$$

$$Q_i = Q_{i-2} - q_{i-1}(P_{i-1} - P_i) \text{ for } i > 1 \tag{5}$$

Moreover we have the pivotal equality:

$$N = P_m^2 + Q_{m-1}Q_m \tag{6}$$

The principal cycle of reduced forms is given by the set of forms $\rho^i(F_0) = ((-1)^{(i-1)}Q_{i-1},\ 2P_i,\ (-1)^i Q_i)$ with the principal form $F_0 = (1, 2q_0, q_0^2 - N)$.

Using (1) to (5), reverse cycle through the quadratic forms $G_i = \rho^\imath(G_0) = ((-1)^{(i-1)}S_{i-1},\ 2R_i,\ (-1)^i S_i)$ to find a symmetry point, *e.g.* a pair of forms G_m, $G_m + 1$ with $R_m = R_{m+1}$ (this happens for $m \approx n/2$). Using (3), (5) write $R_m = t_m S_m/2$ and since $N = R_m^2 + S_{m-1}S_m$, we obtain a factorization of N: $N = S_m \cdot (S_{m-1} + S_m t_m^2/4)$.

SQUFOF – Square Form Factorization

- Complexity $O(N^{1/4})$
- Theory is really complicated
  - But very easy to implement
- Manipulate numbers of size $2\sqrt{N}$ at most
  - Particularly interesting for double-precision numbers
- Often used in QS or NFS implementation to factor residues

SQUFOF – Square Form Factorization

SQUFOF

Input: integer N to factor
Output: a factor p of N or failure

1. [Find square form]

$F \leftarrow F_0$
while $F \neq$ square form — *// Abort and return failure*
  $F \leftarrow \rho(F)$ — *// if takes too long*

2. [Inverse square root]

$G_0 \leftarrow \rho(\sqrt{F})$
$G_1 \leftarrow \rho(G_0)$

3. [Find symmetry point] — *// Needs about half the*
while (no symmetry point) — *// number of iterations*
  $G_0 \leftarrow \rho(G_0)$ — *// needed in step 1.*
  $G_1 \leftarrow \rho(G_1)$

4. [Deduce factors] — *// Simple relation with G_0 and G_1*

The modern times

ECM

CFRAC

QS & variants

NFS

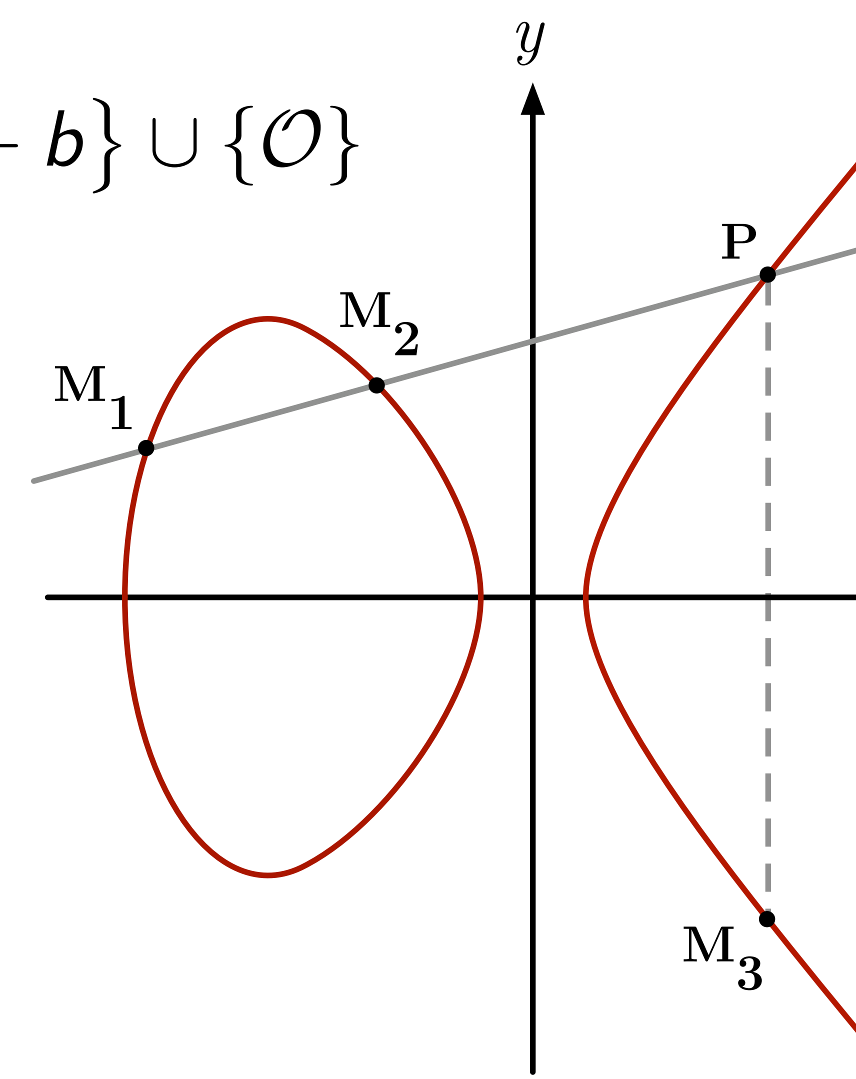
ECM – Elliptic Curve Method

- H.W. Lenstra (1985) + later improvements (Brent, Montgomery)
- Special-purpose algorithm
  - Given $N = p_1 \dots p_k$, runs asymptotically in $L_{p_1}(1/2, \sqrt{2}) \cdot M(N)$
    - $L_x(\alpha, c) = \exp((c + o(1)) \cdot (\log x)^{\alpha} \cdot (\log \log x)^{1-\alpha})$
    - $M(N) \equiv$ cost of multiplication mod N
- In a nutshell, ECM = "p−1 on elliptic curves"
  - p−1 succeeds if $\#\mathbb{Z}_p^*$ is B_1-power smooth
  - ECM succeeds if $\#E(\mathbb{F}_p)$ is B_1-power smooth
    - Retry with another curve if failure

ECM – Elliptic Curve Method

- Crude reminders
  - $E_{a,b}(\mathbb{F}_p) = \{(x, y) \in \mathbb{F}_p \times \mathbb{F}_p : y^2 = x^3 + ax + b\} \cup \{\mathcal{O}\}$
  - Defines a group
    - Chord and tangent group law
  - Hasse theorem

    $$p + 1 - 2\sqrt{p} \leq \#E_{a,b}(\mathbb{F}_p) \leq p + 1 + 2\sqrt{p}$$



- Elliptic "pseudocurve" $E_{a,b}(\mathbb{Z}_N)$
  - Not a group!
  - There are points P_i and Q_i for which $P_i + Q_i$ is not defined
    - Failure to find inverse mod N gives a factor

ECM – Elliptic Curve Method

ECM (first stage)

Input: integer N to factor
bound B_1

Output: a factor p of N or failure

1. [Choose elliptic curve E and initial point Q_0]

// Several strategies are possible.
// Popular are Suyama's curve parameterization and
// Montgomery's point representation

2. Compute $Q \leftarrow [B_1!]\ Q_0$

// If $\#E(\mathbb{F}_{p_1})$ is B_1-power smooth this computation
// will fail – non inversible element x in $\mathbb{Z}_N$

if $(\nexists\ x^{-1} \bmod N)$ **then return** $\gcd(x, N)$

3. return failure

// Try again with another curve
// or with another bound B_1

ECM – Elliptic Curve Method

- Again, second stage with bound B_2
  - Will work if $\#E(\mathbb{F}_p) = p_L \prod_{i=1}^{k} q_i^{e_i}$ with p_L prime in $[B_1, B_2]$
- Idea
  - Let $\{q_{k+1}, q_{k+2} \ldots q_l\}$ be the primes in $[B_1, B_2]$
  - Precompute $R_i = [q_{i+1} - q_i]\ Q$ for all i in $[k+1, l]$
  - Compute

$$Q \leftarrow [q_{k+1}]Q$$

$$Q \leftarrow Q + [R_1]Q$$

$$Q \leftarrow Q + [R_2]Q$$

- Several variants (birthday paradox, standard continuation)

Congruence of squares methods

- Basic idea: Kraïtchik in the 1920s
  - Find U, V so that $U^2 = V^2 \bmod N$
  - Then $\gcd(U - V, N)$ yields a (nontrivial?) factor of N
- Two stages
  - Find congruences
    - Collect $F + \epsilon$ relations r_i of type $x_i^2 = y_i \bmod N$
    - Factor the y_i on a factor base $\mathcal{B} = \{p_1, p_2 \dots p_F\}$
      - $y_i = \prod_{i=1}^{k} p_i^{e_i}$
      - $e_i^* = e_i \bmod 2$
    - Each relation = a row in a $(F + \epsilon) \times F$ matrix $\mathcal{M}$
      - $[e_1^*, e_2^*, e_3^* \dots e_F^*]$
  - Solve linear system
    - Compute kernel of $\mathcal{M}$
      - Gives collections $\{r_i\}_j$ for which $\prod_i y_i = V^2$

Congruence of squares methods

- Relation selection – keep only smooth y_i
  - Trial division
  - Early abort strategy (Pomerance, 1982)
    - Trial divide with a fraction of the p_i (e.g. primes $\leq \sqrt{p_F}$)
    - Abort if cofactor greater than a given bound
    - Multiple steps possible
  - Smoothness detection batch
    - Accumulate several candidates and test in batch
      - Franke, Kleinjung, Morain & Wirth (2004)
      - Bernstein (2004)

Congruence of squares methods

- Large prime variations
  - Allow $y_i = L \prod_i p_i^{e_i}$ with $L = p_{L_1} p_{L_2} \ldots p_{L_{LP}}, \; p_{L_i} > p_F$
  - Usually
    - $LP = 1$ (single large prime variation)
      - Easy to implement
    - $LP = 2$ (double large prime variation)
      - Harder

L_2
$L_2 \; L_4$
L_4

$L_2 \; L_5$
$L_3 \; L_5$
L_3

$L_3 \; L_7$
$L_6 \; L_9$
$L_6 \; L_7$
$L_3 \; L_9$

Congruence of squares methods

- General purpose methods
- One idea, several algorithms
  - CFRAC
  - QS & derivatives
  - NFS
- Main difference is the way the $x_i^2 = y_i \bmod N$ are generated

CFRAC – Continued Fraction Factorization

- Morrison & Brillhart (1975) from ideas from Lehmer & Powers
- A general factoring method
  - Runs in $L_N(1/2, \sqrt{2})$
- Based on continued fraction expansion of $\sqrt{N}$
  - Look for $x_i^2 = y_i \bmod N$ with y_i "small"
  - $x_i^2 = y_i + kN = y_i + k' d^2 N$
  - $(x_i/d)^2 - N = y_i/d^2$ is "small" $\Rightarrow (x_i/d) \simeq \sqrt{N}$
- Let $\langle a_i/b_i \rangle_{\sqrt{N}}$ be the i-th continued fraction convergent to $\sqrt{N}$
  - $|a_i^2 - b_i^2 N| < 2\sqrt{N}$
    - In some sense, the smallest residues possible

CFRAC – Continued Fraction Factorization

- Problem
  - The sequence $\{\langle a_i/b_i\rangle_{\sqrt{N}}\}_i$ is periodic
    - Use a correctly chosen multiplier k and factor kN
- Choosing a multiplier & the factor base
  - $p|y_i \Rightarrow (\frac{kN}{p}) = 1$
  - $\mathcal{B} = \{q_1, q_2 \dots q_F\}$ with $(kN/q_i) = 1$
    - Choose k so that $\mathcal{B}$ contains "lots of" small primes

CFRAC – Continued Fraction Factorization

CFRAC (high level description)

Input: integer N to factor

Output: a factor p of N or failure

1. [Select multiplier k and factor base $\mathcal{B}$]

// Balance size of k and number of small primes in $\mathcal{B}$

2. [Generate relations]

// Expand $\sqrt{kN}$ to generate congruence relations

3. [Select relations]

// Keeps relations with y_i smooth or a product of
// a smooth number with a few large primes

4. [Linear algebra]

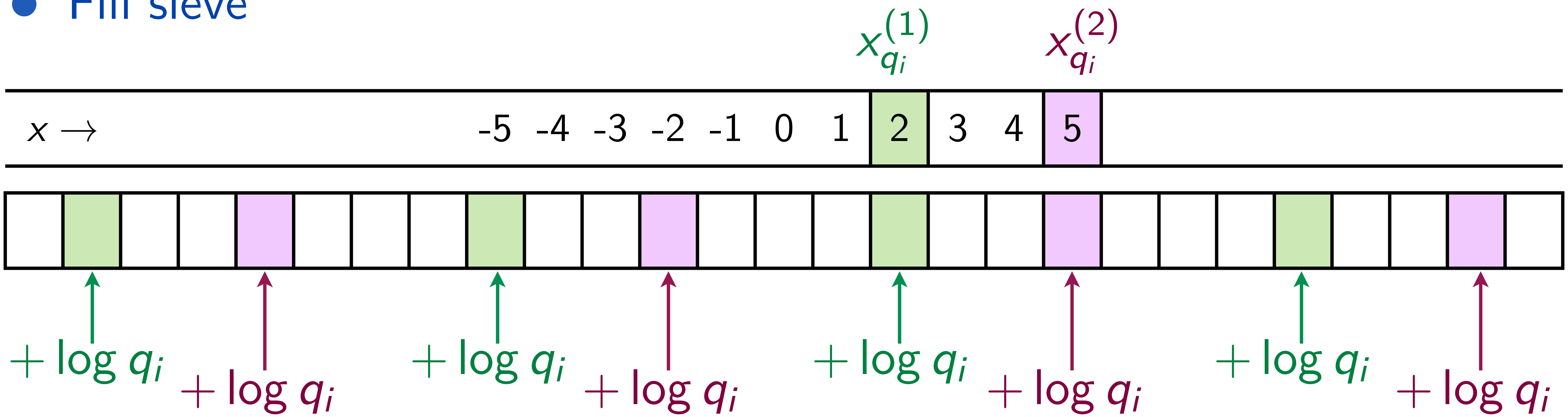
// Compute gcd(N, U-V) for each solution found

QS – Quadratic Sieve

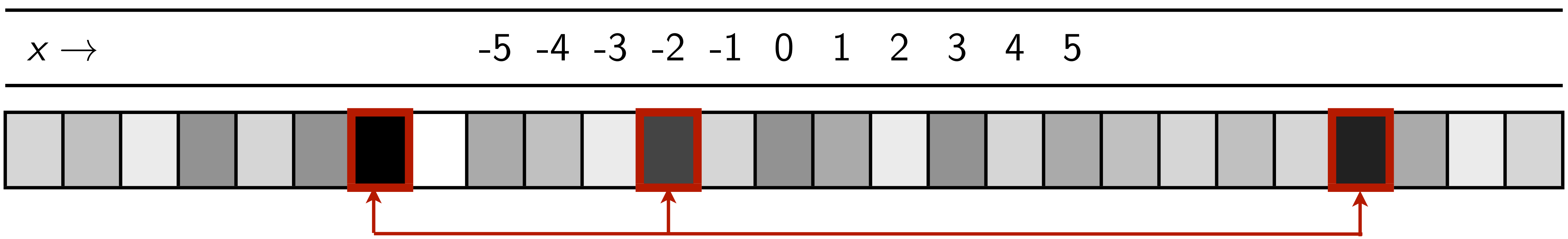
- Pomerance – 1982
- Use of a sieve to quickly discard non smooth residues
- Runs in $L_N(1/2, 1)$
- $g(x) = (x + \lfloor\sqrt{N}\rfloor)^2 - N = u^2 - N$
  - $g(x)$ is $\mathcal{B}$-smooth $\Rightarrow$ relation $g(x) = u^2 \bmod N$
  - $p|g(x) \Rightarrow p|g(x + m.p), m \in \mathbb{Z} \longrightarrow$ sieve
- Sieving
  - Solve $x^2 = N \bmod q_i$ for all $q_i \in \mathcal{B} \longrightarrow x_{q_i}^{(1)}$ and $x_{p_i}^{(2)}$
  - Sieve with $\{x_{q_i}^{(1)}, x_{q_i}^{(2)}\}_i$ and keep potentially smooth $g(x)$ for surviving values of x only

QS – Quadratic Sieve

- Fill sieve



- Scan sieve



Only check if these $g(x_i)$ are smooth

Only the $g(x_i)$ for which Sieve$[x_i] \geq \tau$ are eligible for a smoothness test

MPQS – Multiple polynomial Quadratic Sieve

- Problem with QS
  - $g(x)$ grows linearly (for small x)

- The Multiple Polynomial QS (MPQS)
  - Use several polynomials $g_{a,b}(x) = (a.x + b)^2 - N$
    - Switch polynomial when $g_{a,b}(x)$ gets too large
      - Effectively sieve in interval $[-M, M]$
  - Polynomial initialization problem
    - Need to compute $\{x_{q_i}^{(1)}, x_{q_i}^{(2)}\}_i$ = the solutions to $g_{a,b}(x) = 0 \bmod q_i$ for each new polynomial
      - Can become a bottleneck
  - Faster than QS but same complexity $L_N(1/2, 1)$

SIQS – Self Initializing Quadratic Sieve

- The Self Initializing Quadratic Sieve (SIQS)
  - Choose family $\{g_{a,b_i}\}$ such that $g_{a,b_{i+1}}$ can be quickly initialized from g_{a,b_i}
- In a nutshell
  - Choose $a = \prod_{i=0}^{s} p_i$ so that $a \simeq \sqrt{2N}/M$ *(to minimize $g_{a,b}(x)$)*
  - We want $b^2 - N = ka$ *(since then $a | g_{a,b}(x)$)*
    - Gives 2^s values for b but only 2^{s-1} are suitable
  - Fully initialize g_{a,b_0} *(i.e. compute $\{x_{q_i}^{(1)}, x_{q_i}^{(2)}\}_i$)*
  - The $2^{s-1} - 1$ other g_{a,b_i} can be derived from $g_{a,b_{i-1}}$
  - If more polynomial needed, choose another a

SIQS – Self Initializing Quadratic Sieve

SIQS (seen from the ionosphere)

Input: integer N to factor

Output: a factor p of N or failure

1. [Select multiplier k and factor base $\mathcal{B}$]

2. [Polynomial initialization]

// Choose $a = \prod_{i=0}^{s} p_i \simeq \sqrt{2N}/M$

// 1 full poly-init g_{a,b_0} *for* $2^{s-1} - 1$ *fast poly-init* g_{a,b_i}

3. [Fill Sieve]

// Sieve with the $\{x_{q_i}^{(1)}, x_{q_i}^{(2)}\}_i$

4. [Scan sieve]

// Scan the sieve, keeps x_i *for which* $\mathrm{Sieve}[x_i] \geq \tau$
// and perform smoothness detection on $g_{a,b}(x_i)$.
// If not enough relations, goto step 2

5. [Linear algebra & factor deduction]

// Standard to all congruences of square methods

NFS – Number Field Sieve

- Special NFS (Pollard, 1988)
  - Numbers of the form $c_1 a^n + c_2 b^n$
- General NFS (Buhler/Lenstra/Pomerance, 1990)
  - Arbitrary numbers
- The fastest methods known
  - SNFS
    - $L_N(1/3, \sqrt[3]{32/9})$ in time
    - $L_N(1/3, \sqrt[3]{32/9})^{1/2}$ in space
  - GNFS
    - $L_N(1/3, \sqrt[3]{64/9})$ in time
    - $L_N(1/3, \sqrt[3]{64/9})^{1/2}$ in space

NFS – Number Field Sieve

- Basic GNFS in a nutshell (and from high up there)
  - Monic irreducible polynomial $f \in \mathbb{Z}[x]$ of degree d
    - $m \in \mathbb{Z}_N$ so that $f(m) \equiv 0 \bmod N$
    - $\alpha \in \mathbb{C}$ so that $f(\alpha) = 0$
  - Ring morphism
    - $\phi : \mathbb{Z}[\alpha] \to \mathbb{Z}_N$

      $\sum_{i=0}^{d-1} a_i \alpha^i \mapsto \left(\sum_{i=0}^{d-1} a_i m^i\right) \bmod N$
  - Consider pairs $\theta_i, \phi(\theta_i)$ so that
    - $\theta_1 \ldots \theta_k = \gamma^2$ in $\mathbb{Z}[\alpha]$ (algebraic side)
    - $\phi(\theta_1) \ldots \phi(\theta_k) = v^2 \bmod N$ in $\mathbb{Z}_N$ (rational side)

NFS – Number Field Sieve

- Let $\phi(\gamma) = u \bmod N$
  - $u^2 = \phi(\gamma)^2 = \phi(\gamma^2) = \phi(\theta_1 \ldots \theta_k) = \phi(\theta_1) \ldots \phi(\theta_k) = v^2 \bmod N$
  - $u^2 = v^2 \bmod N$
    - computing algebraic square root γ from γ^2 not trivial
  - Now, look for $\theta_i = a_i + b_i\alpha$ with $\gcd(a_i, b_i) = 1$

    $$\phi(\theta_i) = a_i + b_i m$$
- Sieving rational side
  - $g(x) = x + m$
  - Let r be a root of g mod p_i
    - $p_i | b \cdot g(a/b) \Leftrightarrow a = rb \bmod p_i$
      - Sieve along a for each b

NFS – Number Field Sieve

- Sieving algebraic side
  - Note $\alpha, \alpha_2 \ldots \alpha_d$ the complex roots of f
    - $$\begin{aligned} \|a + b\alpha\| &= (a + b\alpha) \ldots (a + b\alpha_d) \\ &= b^d (a/b + \alpha) \ldots (a/b + \alpha_d) \\ &= b^d f(a/b) \end{aligned}$$
    - $p_i | b^d f(a/b) \Leftrightarrow a = rb \bmod p_i$ with r a root of $f \bmod p_i$
      - Sieve along a for each b
      - Take intersection with rational sieve survivors
- Linear algebra + deducing factors
  - As in other congruence of square methods...
    - Modulo algebraic square root problem not trivial

NFS – Number Field Sieve

NFS (seen from the Moon)

Input: integer N to factor

Output: a factor p of N or failure

1. **[Polynomial selection]**
   // Select f and g (usually g of degree 1)
2. **[Sieving]**
   // Sieve for the two polynomials f and g
3. **[Filtering]**
   // Prepare the matrix for linear algebra
4. **[Linear algebra]**
   // Usually block Wiedemann or block Lanczos
5. **[Square roots]**
   // Algebraic square root non trivial

NFS – Number Field Sieve

- Warning
  - Lots of details swept under the rug!
    - In particular for $x \in \mathbb{Z}[\alpha]$, $||x|| = a^2 \not\Rightarrow x = b^2$

- Lots of enhancements
  - Polynomial selection methods
  - Lattice sieving

NFS – Number Field Sieve

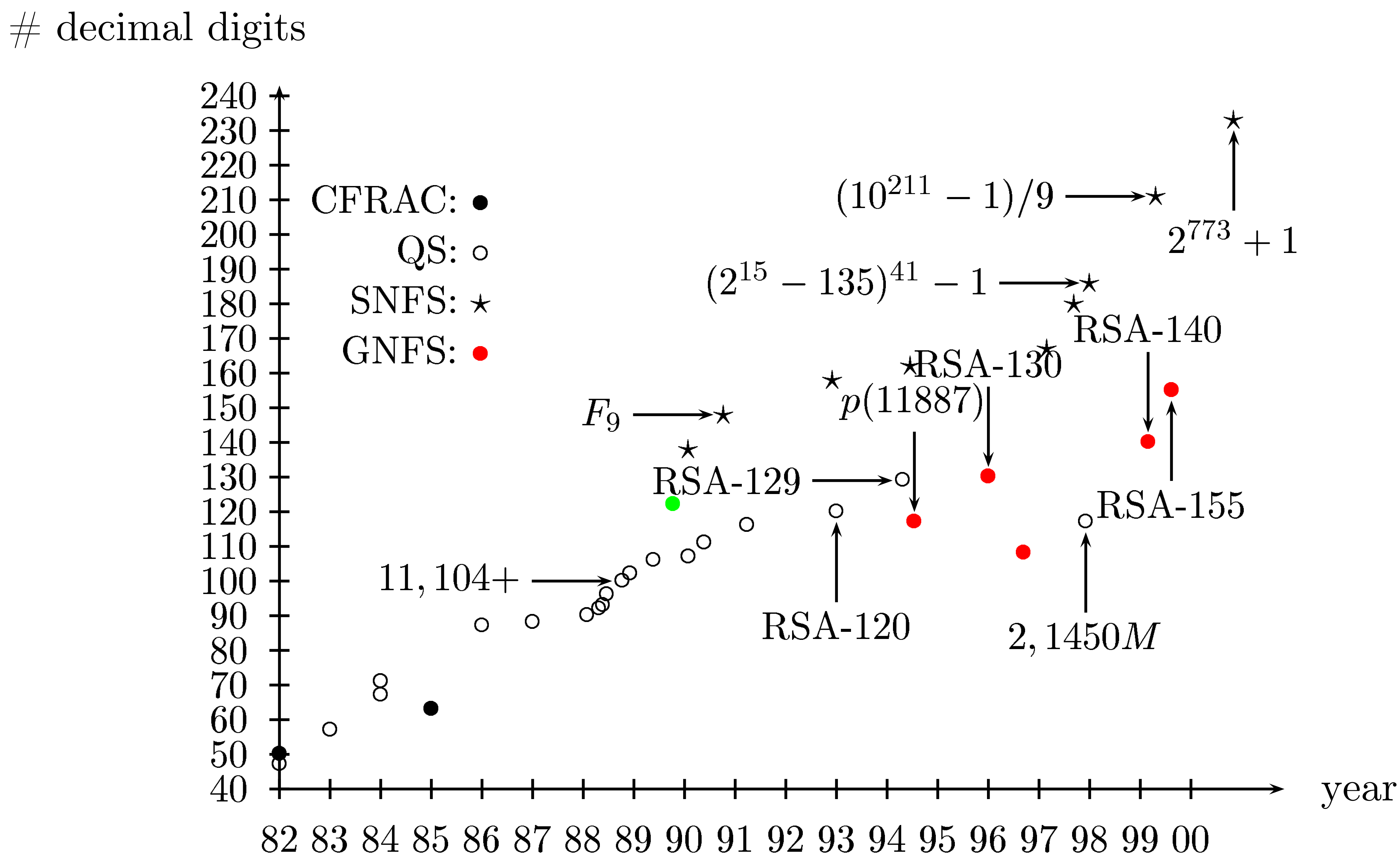
- Special NFS
  - 1990 : 9th Fermat number $F_9 = 2^{512} + 1$
  - 2000 : $2^{773} + 1$
  - 2007 : $2^{1039} - 1$
- General NFS
  - 1999 : RSA-155
  - 2009 : RSA-768 (232 decimal digits)
    - Estimated to be about 10 times harder than $2^{1039} - 1$

NFS – Number Field Sieve

- Factoring RSA-768 (232 decimal digits)
  - Polynomial selection – 6 months / 80 cores
  - Sieving – 24 months / hundreds of cores
    - $64 \cdot 10^9$ relations (5 TB)
  - Filtering – 20 days / 2 cores + 10 TB disk space
  - Linear algebra – 3 months / 600 cores (estimation)
    - $193 \cdot 10^6 \times 193 \cdot 10^6$ matrix (105 GB)
    - Block Wiedermann, up to 1 TB RAM needed
  - Square roots – A few hours / 12 cores

From "Factorization of a 768-bit RSA modulus", Kleinjung et al., 2010

Factorization records



From "Thirty Years of Integer Factorization", F. Morain, 2001

Factorization records

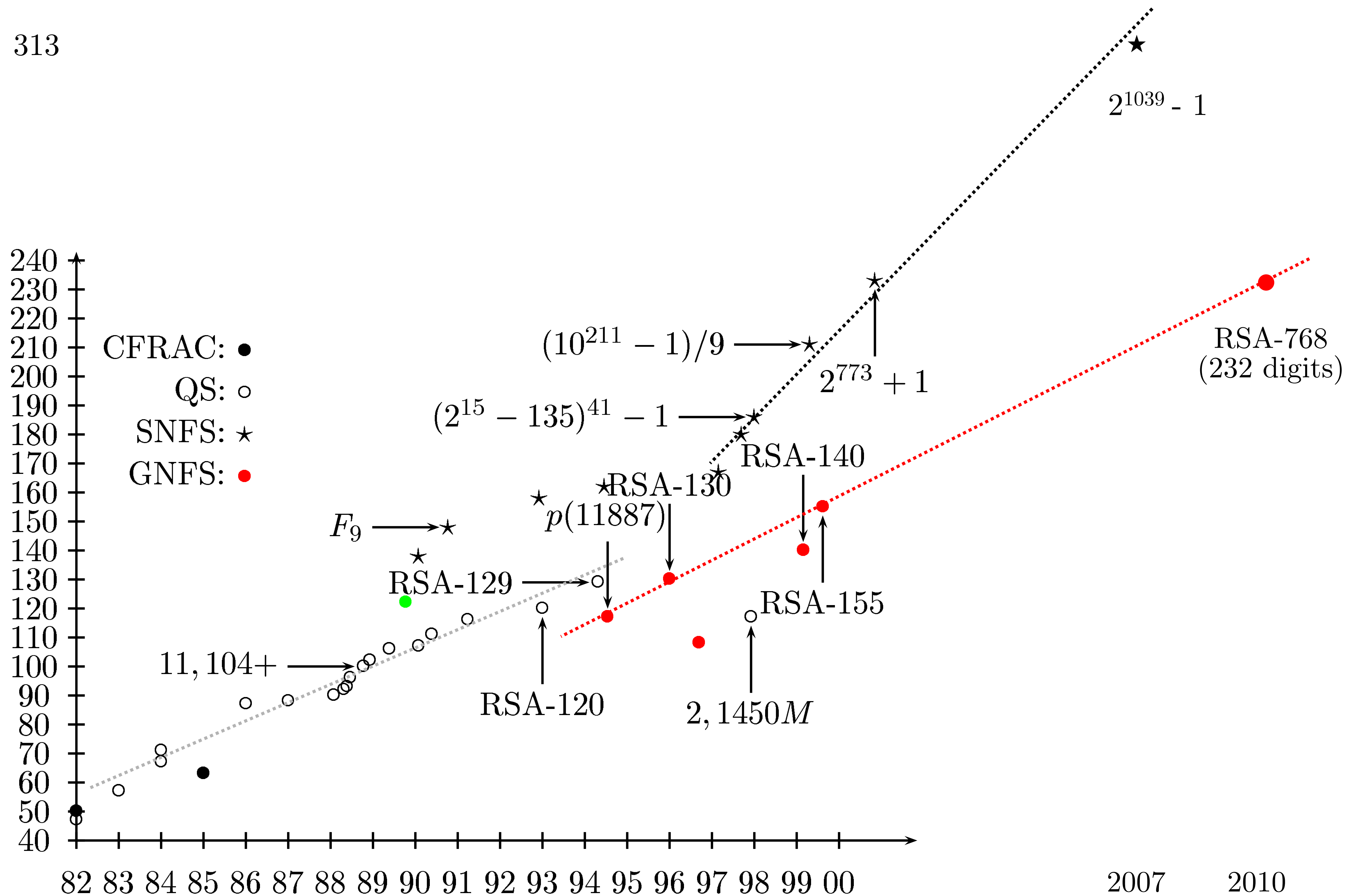


FIGURE 1. Size in bits of the factored numbers depending on the year.

References

Mathematics of Public Key Cryptography
Steven Galbraith
http://www.isg.rhul.ac.uk/~sdg/crypto-book/crypto-book.html

Prime numbers — A computational perspective
Richard Crandall & Carl Pomerance
http://www.springer.com/mathematics/numbers/book/978-0-387-25282-7

Prime numbers and Computer Methods for Factorization
Hans Riesel
http://www.springer.com/birkhauser/mathematics/book/978-0-8176-3743-9

Theorems on factorization and primality testing
John Pollard
Proceedings of the Cambridge Philosophical Society, vol. 76, issue 03, p. 521, 1974.

Thirty Years of Integer Factorization
François Morain
http://algo.inria.fr/seminars/sem00-01/morain.ps

A p+1 method of factoring
Hugh Williams
Mathematics of Computation, vol. 39, p. 225-234, 1982.

References

Square form factorization
Jason Gower & Samuel Wagstaff Jr.
Mathematics of computation, vol. 77, no. 261, pp. 551-588, 2008.

Factoring integers with elliptic curves
Hendrik Lenstra
Annals of Mathematics, vol. 126, pp. 649-673, 1987.

Speeding the Pollard and Elliptic Curve Methods of Factorization
Peter Montgomery
Mathematics of Computation, vol. 48, pp. 243–264, 1987.

A method of factoring and the factorization of F_7
Michael Morrison & John Brillhart
Mathematics of Computation, vol. 29, no. 129, pp. 183–205, 1975.

Factoring integers with the self-initializing quadratic sieve
Scott Contini
http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.26.6924

The number field sieve
Arjen Lenstra *et al.*
http://www.std.org/~msm/common/nfspaper.pdf

References

A Tale of Two Sieve
Carl Pomerance
http://www.ams.org/notices/199612/pomerance.pdf

Factorization of a 768-bit RSA modulus
Thorsten Kleinjung *et al.*
http://eprint.iacr.org/2010/006.pdf