Factoring RSA-250 with PRACE

Fabrice Boudot, Pierrick Gaudry, Aurore Guillevic, Nadia Heninger,
Emmanuel Thomé, **Paul Zimmermann**

November 16, 2021, CaSToRC HPC

Introduction: public-key cryptography

1976 (Diffie–Hellman, DH) and 1977 (Rivest–Shamir–Adleman, RSA)

Asymmetric means distinct public and private keys

- encryption with a public key
- decryption with a private key
- deducing the private key from the public key is a hard problem

Two hard problems:

- Integer factorization (for RSA)
- Discrete logarithm computation in a finite cyclic group (for Diffie–Hellman)

Public-key encryption: 1977, Rivest, Shamir, Adleman (RSA)

Alice **Bob**

Public-key encryption: 1977, Rivest, Shamir, Adleman (RSA)

Alice **Bob**

0. chooses public parameters:
   modulus $N = p \cdot q$
   encryption key e
   decryption key
   $d = e^{-1} \bmod (p-1)(q-1)$

Public-key encryption: 1977, Rivest, Shamir, Adleman (RSA)

Alice **Bob**

0. chooses public parameters:
   modulus $N = p \cdot q$
   encryption key e
   decryption key
   $d = e^{-1} \bmod (p-1)(q-1)$

$\xrightarrow{N,\, e}$

Public-key encryption: 1977, Rivest, Shamir, Adleman (RSA)

Alice

0. chooses public parameters:
   modulus $N = p \cdot q$
   encryption key e
   decryption key
   $d = e^{-1} \bmod (p-1)(q-1)$

$\xrightarrow{N, e}$

Bob

1. gets Alice's public key (N, e)
2. encodes m as integer in $[0, N-1]$
3. ciphertext $c = m^e \bmod N$
4. sends c to Alice

Public-key encryption: 1977, Rivest, Shamir, Adleman (RSA)

Alice

0. chooses public parameters:
   modulus $N = p \cdot q$
   encryption key e
   decryption key
   $d = e^{-1} \bmod (p-1)(q-1)$

$\xrightarrow{N, e}$

Bob

1. gets Alice's public key (N, e)
2. encodes m as integer in $[0, N-1]$
3. ciphertext $c = m^e \bmod N$
4. sends c to Alice

$\xleftarrow{c}$

Public-key encryption: 1977, Rivest, Shamir, Adleman (RSA)

Alice

0. chooses public parameters:
   modulus $N = p \cdot q$
   encryption key e
   decryption key
   $d = e^{-1} \bmod (p-1)(q-1)$

$\xrightarrow{N, e}$

Bob

1. gets Alice's public key (N, e)
2. encodes m as integer in $[0, N-1]$
3. ciphertext $c = m^e \bmod N$
4. sends c to Alice

$\xleftarrow{c}$

5. gets c from Bob
6. computes $m = c^d \bmod N$

Public-key encryption: 1977, Rivest, Shamir, Adleman (RSA)

Alice

0. chooses public parameters:
   modulus $N = p \cdot q$
   encryption key e
   decryption key
   $d = e^{-1} \bmod (p-1)(q-1)$

$\xrightarrow{N, e}$

Bob

1. gets Alice's public key (N, e)
2. encodes m as integer in $[0, N-1]$
3. ciphertext $c = m^e \bmod N$
4. sends c to Alice

$\xleftarrow{c}$

5. gets c from Bob
6. computes $m = c^d \bmod N$

It works: $m^{ed} \equiv m \bmod N$
because $ed = 1 \bmod (p-1)(q-1)$

RSA, security, attacks

Mathematical security relies on the hardness of computing d from N, e.

p, q required to compute $d = 1/e \bmod (p-1)(q-1)$

$\rightarrow$ security relies on the hardness of **integer factorization**.

Usecases:
`ssh-keygen` (linux), PGP: Enigmails on Thunderbird, Protonmail.

Note that short keys are not allowed:

```
ssh-keygen -b 512 -t rsa
Invalid RSA key length: minimum is 1024 bits
```

Factoring RSA-250

Factoring RSA modulus of 250 decimal digits (829 bits)
$N =$
214032465024074496126442307283933356300861471514475501779775492
088141802344714013664334551909580467961099285187247091458768739
626192155736304745477052080511905649310668769159001975940569345
7452230589325976697471681738069364894699871578494975937497937

Hardware (PRACE's Juwels supercomputer):
Intel Platinum 8168 at 2.7Ghz

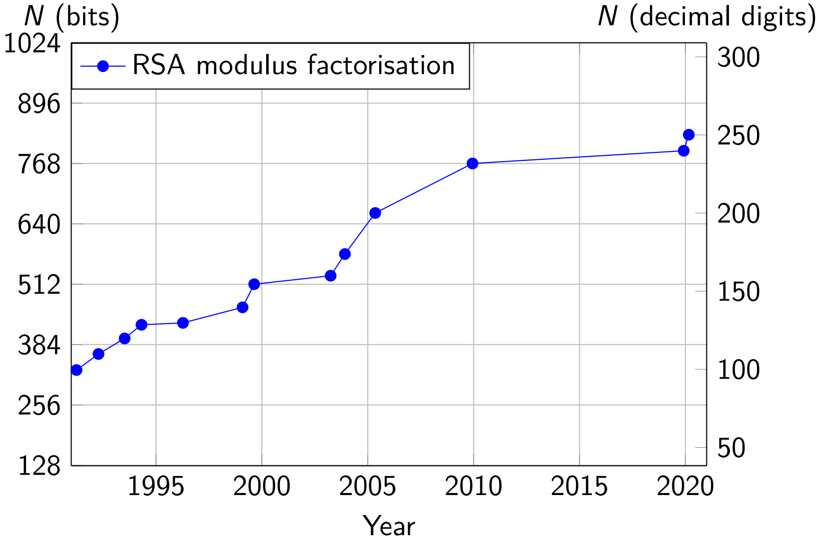
Integer factorization algorithms

- trial division: try all prime numbers up to e.g. 10^7
- ECM (Elliptic Curve Method, Lenstra 87): find medium-size factors
- Quadratic sieve: N up to 100 decimal digits
- Number Field Sieve: N larger than 100 decimal digits

Historical steps in integer factorization

- 1975, Morrison, Brillhard, continued fraction method CFRAC, factorization of $2^{128}+1$, see the *Cunningham project*
$2^{128}+1 = 340282366920938463463374607431768211457 =$
$59649589127497217 \times 5704689200685129054721$
- 1982, Pomerance, Quadratic Sieve
- 1987, Lenstra, Elliptic Curve Method (ECM)
- 1993, Buhler, Lenstra, Pomerance, General Number Field Sieve

Factorization Records with NFS



Nowadays' method: the Number Field Sieve

- developed in the 80's and 90's
- reduce the size of the numbers to be factored from $A_0\sqrt{N}$ to $A^d\sqrt[d]{N}$ for a smaller $A < A_0$ and $d \in \{3, 4, 5, 6\}$
- two huge steps: collecting relations, solving a large sparse system

Factorization with NFS: key idea

Reduce further the size of the integers to factor

Choose integer $m \approx \sqrt[d]{N}$

Write N in basis m: $N = c_0 + c_1 m + \ldots + c_d m^d$

Set $f_1(x) = c_0 + c_1 x + \ldots + c_d x^d \implies f_1(m) = 0$, set $f_0 = x - m \implies f_0(m) = 0$

Polynomials f_0, f_1 share a common root m modulo N

If f_1 is irreducible, define $\alpha \in \mathbb{C}$ a root of f_1

Factorization with NFS: key idea

Reduce further the size of the integers to factor

Choose integer $m \approx \sqrt[d]{N}$

Write N in basis m: $N = c_0 + c_1 m + \ldots + c_d m^d$

Set $f_1(x) = c_0 + c_1 x + \ldots + c_d x^d \implies f_1(m) = 0$, set $f_0 = x - m \implies f_0(m) = 0$

Polynomials f_0, f_1 share a common root m modulo N

If f_1 is irreducible, define $\alpha \in \mathbb{C}$ a root of f_1

Define a map from $\mathbb{Z}[\alpha]$ to $\mathbb{Z}/N\mathbb{Z}$

$$\begin{aligned} \phi\colon \mathbb{Z}[\alpha] &\to \mathbb{Z}/N\mathbb{Z} \\ \alpha &\mapsto m \bmod N \text{ where } f_1(m) = 0 \bmod N \end{aligned}$$

ring homomorphism $\phi(a + b\alpha) = a + bm$

$$\phi \underbrace{(a + b\alpha)}_{\text{factor in } \mathbb{Z}[\alpha]} = \underbrace{a + bm}_{\text{factor in } \mathbb{Z}} \mod N$$

Factorization with NFS: key idea

Reduce further the size of the integers to factor

Choose integer $m \approx \sqrt[d]{N}$

Write N in basis m: $N = c_0 + c_1 m + \ldots + c_d m^d$

Set $f_1(x) = c_0 + c_1 x + \ldots + c_d x^d \implies f_1(m) = 0$, set $f_0 = x - m \implies f_0(m) = 0$

Polynomials f_0, f_1 share a common root m modulo N

If f_1 is irreducible, define $\alpha \in \mathbb{C}$ a root of f_1

Define a map from $\mathbb{Z}[\alpha]$ to $\mathbb{Z}/N\mathbb{Z}$

$$\begin{array}{rcl} \phi\colon \mathbb{Z}[\alpha] & \to & \mathbb{Z}/N\mathbb{Z} \\ \alpha & \mapsto & m \bmod N \text{ where } f_1(m) = 0 \bmod N \end{array}$$

ring homomorphism $\phi(a + b\alpha) = a + bm$

$$\phi \quad \underbrace{(a + b\alpha)}_{\substack{\text{factor in } \mathbb{Z}[\alpha] \\ \text{size } A^d N^{1/d}}} \quad = \quad \underbrace{a + bm}_{\substack{\text{factor in } \mathbb{Z} \\ \text{size } AN^{1/d}}} \quad \bmod N$$

Factorization with NFS: recap

1. Polynomial selection: find two irreducible polynomials in $\mathbb{Z}[x]$ sharing a common root m modulo N
2. Relation collection: computes many smooth relations
3. Linear algebra: takes logarithms mod 2 of the relations: large sparse matrix over $\mathbb{F}_2$, computes left kernel
4. Characters: find a combination of the vectors of the kernel so that $X^2 \equiv Y^2 \bmod N$
5. Square root: computes X, Y
6. Factor N: computes $\gcd(X - Y, N)$

Factorization of RSA-250

$$
\begin{aligned}
\text{RSA-250} &= 2140324650240744961264423072839333563008614715144755017797754920\\
&\ 8814180234471401366433455190958046796109928518724709145876873 9\\
&\ 6261921557363047454770520805119056493106687691590019759405693 45\\
&\ 7452230589325976697471681738069364894699871578494975937497937,\\
p &= 6413528947707158027879019017057738908482501474294344720811685 96\\
&\ 32024532344630238623598752668347708737661925585694639798853367,\\
q &= 3337202759497815655622601060535511422794076034476755466678452 09\\
&\ 87023841729210037080257448673296881877565718986258036932062711
\end{aligned}
$$

Breaking the previous record: Why?

- Record computations needed for key-size recommendations
- Open-source software Cado-NFS
- Motivation to improve all the steps
- Testing folklore ideas competitive only for huge sizes
- Exploits improvements of ECM (Bouvier–Imbert PKC'2020)
- Scaling the code for larger sizes improves the running-time on smaller sizes

The CADO-NFS software

Record computations with the CADO-NFS software.

- Important software development effort since 2007.
- 250k lines of C/C++ code, 60k for relation collection only.
- Significant improvements since 2016.
  - improved parallelism: strive to get rid of scheduling bubbles;
  - versatility: large freedom in parameter selection;
  - prediction of behaviour and yield: essential for tuning.
- Open source (LGPL), open development model (gitlab).
  Our results can be reproduced.

Factorization 250 dd

$N = \text{RSA-250}$

Polynomial selection

$$
\begin{aligned}
f_1 &= 86130508464000x^6 \\
&\quad -66689953322631501408x^5 \\
&\quad -52733221034966333966198x^4 \\
&\quad +46262124564021437136744523465879x^3 \\
&\quad -3113627253613202265126907420550648326x^2 \\
&\quad -1721614429538740120011760034829385792019395x \\
&\quad -81583513076429048837733781438376984122961112000 \\
f_0 &= 185112968818638292881913x \\
&\quad -3256571715934047438664355774734330386901
\end{aligned}
$$

$\mathrm{Res}(f_0, f_1) = 48N$

Relations look like

small primes, special-q, large primes

✓	$5^2 \cdot 11 \cdot 23 \cdot 287093 \cdot 870953 \cdot 20179693 \cdot 28306698811 \cdot 47988583469$	$2^3 \cdot 5 \cdot 7 \cdot 13 \cdot 31 \cdot 61 \cdot 14407 \cdot 26563253 \cdot 86800081 \cdot 269845309 \cdot 802234039 \cdot 1041872869 \cdot 5552238917 \cdot 12144939971 \cdot 15856830239$
✓	$3 \cdot 1609 \cdot 77699 \cdot 235586599 \cdot 347727169 \cdot 369575231 \cdot 9087872491$	$2^3 \cdot 3 \cdot 5 \cdot 13 \cdot 19 \cdot 23 \cdot 31 \cdot 59 \cdot 239 \cdot 3989 \cdot 7951 \cdot 2829403 \cdot 31455623 \cdot 225623753 \cdot 811073867 \cdot 1304127157 \cdot 78955382651 \cdot 129320018741$
✓	$5 \cdot 1381 \cdot 877027 \cdot 15060047 \cdot 19042511 \cdot 11542780393 \cdot 13192388543$	$2^4 \cdot 5 \cdot 13 \cdot 31 \cdot 59 \cdot 823 \cdot 2801 \cdot 26539 \cdot 2944817 \cdot 3066253 \cdot 87271397 \cdot 108272617 \cdot 386616343 \cdot 815320151 \cdot 1361785079 \cdot 12322934353$
✓	$2^3 \cdot 5^2 \cdot 173 \cdot 971 \cdot 613909489 \cdot 929507779 \cdot 1319454803 \cdot 2101983503$	$2^7 \cdot 3^2 \cdot 5 \cdot 29 \cdot 1021 \cdot 42589 \cdot 190507 \cdot 473287 \cdot 31555663 \cdot 654820381 \cdot 802234039 \cdot 19147596953 \cdot 23912934131 \cdot 52023180217$
✗	$2^2 \cdot 15193 \cdot 232891 \cdot 19514983 \cdot 139295419 \cdot 540260173 \cdot 606335449$	$2^2 \cdot 3^4 \cdot 13 \cdot 19 \cdot 74897 \cdot 1377667 \cdot 55828453 \cdot 282012013 \cdot 802234039 \cdot 3350122463 \cdot 35787642311 \cdot 37023373909 \cdot 128377293101$
✗	$2^2 \cdot 5^4 \cdot 439 \cdot 1483 \cdot 13121 \cdot 21383 \cdot 67751 \cdot 452059523 \cdot 33099515051$	$2^2 \cdot 3^3 \cdot 11 \cdot 13 \cdot 19 \cdot 5023 \cdot 3683209 \cdot 98660459 \cdot 802234039 \cdot 1506372871 \cdot 4564625921 \cdot 27735876911 \cdot 32612130959 \cdot 45729461779$

small primes: abundant $\rightarrow$ dense column in the matrix
large primes: rare $\rightarrow$ sparse colum, limit to 2 or 3 on each side.

Relations look like

small primes, special-q, large primes

✓	$5^2 \cdot 11 \cdot 23 \cdot 287093 \cdot 870953 \cdot 20179693 \cdot 28306698811 \cdot 47988583469$	$2^3 \cdot 5 \cdot 7 \cdot 13 \cdot 31 \cdot 61 \cdot 14407 \cdot 26563253 \cdot 86800081 \cdot 269845309 \cdot 802234039 \cdot 1041872869 \cdot 5552238917 \cdot 12144939971 \cdot 15856830239$
✓	$3 \cdot 1609 \cdot 77699 \cdot 235586599 \cdot 347727169 \cdot 369575231 \cdot 9087872491$	$2^3 \cdot 3 \cdot 5 \cdot 13 \cdot 19 \cdot 23 \cdot 31 \cdot 59 \cdot 239 \cdot 3989 \cdot 7951 \cdot 2829403 \cdot 31455623 \cdot 225623753 \cdot 811073867 \cdot 1304127157 \cdot 78955382651 \cdot 129320018741$
✓	$5 \cdot 1381 \cdot 877027 \cdot 15060047 \cdot 19042511 \cdot 11542780393 \cdot 13192388543$	$2^4 \cdot 5 \cdot 13 \cdot 31 \cdot 59 \cdot 823 \cdot 2801 \cdot 26539 \cdot 2944817 \cdot 3066253 \cdot 87271397 \cdot 108272617 \cdot 386616343 \cdot 815320151 \cdot 1361785079 \cdot 12322934353$
✓	$2^3 \cdot 5^2 \cdot 173 \cdot 971 \cdot 613909489 \cdot 929507779 \cdot 1319454803 \cdot 2101983503$	$2^7 \cdot 3^2 \cdot 5 \cdot 29 \cdot 1021 \cdot 42589 \cdot 190507 \cdot 473287 \cdot 31555663 \cdot 654820381 \cdot 802234039 \cdot 19147596953 \cdot 23912934131 \cdot 52023180217$

small primes: abundant $\rightarrow$ dense column in the matrix
large primes: rare $\rightarrow$ sparse colum, limit to 2 or 3 on each side.

Before linear algebra: filtering step
as many cheap combinations as possible $\rightarrow$ smaller matrix

Relation collection looks like

```
1   [|||||||||||100.0%]   17 [|||||||||||100.0%]   33 [|||||||||||100.0%]   49 [|||||||||||100.0%]
2   [|||||||||||100.0%]   18 [|||||||||||100.0%]   34 [|||||||||||100.0%]   50 [|||||||||||100.0%]
3   [|||||||||||100.0%]   19 [|||||||||||100.0%]   35 [|||||||||||100.0%]   51 [|||||||||||100.0%]
4   [|||||||||||100.0%]   20 [|||||||||||100.0%]   36 [|||||||||||100.0%]   52 [|||||||||||100.0%]
5   [|||||||||||100.0%]   21 [|||||||||||100.0%]   37 [|||||||||||100.0%]   53 [|||||||||||100.0%]
6   [|||||||||||100.0%]   22 [|||||||||||100.0%]   38 [|||||||||||100.0%]   54 [|||||||||||100.0%]
7   [|||||||||||100.0%]   23 [|||||||||||100.0%]   39 [|||||||||||100.0%]   55 [|||||||||||100.0%]
8   [|||||||||||100.0%]   24 [|||||||||||100.0%]   40 [|||||||||||100.0%]   56 [|||||||||||100.0%]
9   [|||||||||||100.0%]   25 [|||||||||||100.0%]   41 [|||||||||||100.0%]   57 [|||||||||||100.0%]
10  [|||||||||||100.0%]   26 [|||||||||||100.0%]   42 [|||||||||||100.0%]   58 [|||||||||||100.0%]
11  [|||||||||||100.0%]   27 [|||||||||||100.0%]   43 [|||||||||||100.0%]   59 [|||||||||||100.0%]
12  [|||||||||||100.0%]   28 [|||||||||||100.0%]   44 [|||||||||||100.0%]   60 [|||||||||||100.0%]
13  [|||||||||||100.0%]   29 [|||||||||||100.0%]   45 [|||||||||||100.0%]   61 [|||||||||||100.0%]
14  [|||||||||||100.0%]   30 [|||||||||||100.0%]   46 [|||||||||||100.0%]   62 [|||||||||||100.0%]
15  [|||||||||||100.0%]   31 [|||||||||||100.0%]   47 [|||||||||||100.0%]   63 [|||||||||||100.0%]
16  [|||||||||||100.0%]   32 [|||||||||||100.0%]   48 [|||||||||||100.0%]   64 [|||||||||||100.0%]
Mem[|||||||||||||||||||||||||||||||||||170G/188G]   Tasks: 365, 119 thr; 65 running
Swp[                                    0K/3.72G]   Load average: 65.01 64.26 52.02
                                                    Uptime: 00:42:24
```

Relations, matrix size, core-years timings

	RSA-250
polynomial selection	130 core-years
deg f_0, deg f_1	1, 6
relation collection	2450 core-years
raw relations	8 745 268 073
unique relations	6 132 671 469
filtering	days
after singleton removal	$2\,739\,226\,048 \times 2\,620\,512\,252$
after clique removal	$1\,816\,698\,332 \times 1\,816\,698\,172$
after merge	405M rows, density 252
linear algebra	250 core-years
characters, sqrt, ind log	days

How PRACE was useful

- Access to many more computers than by our institutes (10x more)
- We asked (and obtained) an allocation of 32M core-hours
- Homogeneous processors, operating system, compilers
- Preparatory access was useful to estimate the requested time
- Support from PRACE engineers (not much used in our case)

What you should know about PRACE

There are several clusters around Europe: choose the one best suited for your application, and request a preparatory access on it.

If you are allocated say 12M core-hours for one year, you should spend 1M core-hours each month. You can still spend the hours from month $M - 1$ during month M, but not later. Start early!

Other PRACE users also have large allocations. Submit your jobs early, they might take days to start, but once submitted you make progress in the queue.

Check regularly progress of your computations. If there is an error in your submission script, you might lose millions of core-hours!

Make sure you make full use of each node. If you use a node with say 100 cores during 100 hours, you will be charged 10,000 hours, even if your program uses only one core!

Take home messages

A PRACE submission is light (23 pages in our case).

It can give a major speed up to a big computation.

All scientific domains are eligible, not reserved to numerical simulations!