# Computation of a 768-bit prime field discrete logarithm

C. Diem, T. Kleinjung, A. K. Lenstra, C. Priplata, C. Stahlke

EPF Lausanne (Switzerland), Universität Leipzig (Germany)


# Paris, 1st April 1776

# Paris, 1st April 1776

Marie-Sophie Germain

Germain primes $q$, $p = 2q + 1$

# Paris, 1st April 1776

Marie-Sophie Germain

Germain primes $q$, $p = 2q + 1$

Example: $q = [2^{765}\pi] + 31380$, $p = [2^{766}\pi] + 62762$

# Two centuries later

## Diffie-Hellman key exchange

Public data: prime $p$ and generator $g$ of $\mathbf{F}_p^\times$

1. Alice chooses private random $a$ and computes $A = g^a$.
2. Bob chooses private random $b$ and computes $B = g^b$.
3. Alice sends $A$ to Bob; Bob sends $B$ to Alice.
4. Alice computes common secret $S = g^{ab}$ as $S = B^a$.
5. Bob computes common secret $S = g^{ab}$ as $S = A^b$.

# Two centuries later

## Diffie-Hellman key exchange

Public data: prime $p$ and generator $g$ of $\mathbf{F}_p^\times$

1. Alice chooses private random $a$ and computes $A = g^a$.
2. Bob chooses private random $b$ and computes $B = g^b$.
3. Alice sends $A$ to Bob; Bob sends $B$ to Alice.
4. Alice computes common secret $S = g^{ab}$ as $S = B^a$.
5. Bob computes common secret $S = g^{ab}$ as $S = A^b$.

## Discrete logarithm problem (DLP)

Given a prime $p$, a generator $g$ of $\mathbf{F}_p^\times$ and a target $h \in \mathbf{F}_p^\times$ find an integer $x$ (denoted by $\log_g h$) such that $g^x = h$.

# Two centuries later

## Diffie-Hellman key exchange

Public data: prime $p$ and generator $g$ of $\mathbf{F}_p^\times$

1. Alice chooses private random $a$ and computes $A = g^a$.
2. Bob chooses private random $b$ and computes $B = g^b$.
3. Alice sends $A$ to Bob; Bob sends $B$ to Alice.
4. Alice computes common secret $S = g^{ab}$ as $S = B^a$.
5. Bob computes common secret $S = g^{ab}$ as $S = A^b$.

## Discrete logarithm problem (DLP)

Given a prime $p$, a generator $g$ of $\mathbf{F}_p^\times$ and a target $h \in \mathbf{F}_p^\times$ find an integer $x$ (denoted by $\log_g h$) such that $g^x = h$.

Investigate how secure this is for, say, 768-bit prime fields:

## Our challenge

Solve the DLP for $p = [2^{766}\pi] + 62762$, $g = 11$, $h = [2^{766}e]$, i.e., find $x$ such that

$$11^x \equiv h \pmod{p}.$$

# Factoring

Number field sieve (NFS):

1. **Polynomial selection:** Find two polynomials $f_1, f_2 \in \mathbf{Z}[x]$ with a common zero $m$ modulo $N$ (and some conditions).
   Denote by $F_1, F_2$ the corresponding homogeneous polynomials.
2. **Sieving:** Choose $L$ and find sufficiently many pairs $a, b \in \mathbf{Z}$ (relations) such that $F_1(a, b)$ and $F_2(a, b)$ factor into primes $\leq L$.
3. **Matrix step:** Construct a matrix from these relations.
   Solve this system of linear equations modulo 2.

Each solution gives rise to a congruence $c^2 \equiv d^2 \pmod{N}$, and $\gcd(c + d, N)$ is a proper divisor of $N$ with probability $\geq \frac{1}{2}$.

# Discrete logarithms

Number field sieve (NFS):

1. **Polynomial selection:** Find two polynomials $f_1, f_2 \in \mathbf{Z}[x]$ with a common zero $m$ modulo $p$ (and some conditions).
   Denote by $F_1, F_2$ the corresponding homogeneous polynomials.
2. **Sieving:** Choose $L$ and find sufficiently many pairs $a, b \in \mathbf{Z}$ (relations) such that $F_1(a, b)$ and $F_2(a, b)$ factor into primes $\leq L$.
3. **Matrix step:** Construct a matrix from these relations.
   Solve this system of linear equations modulo $q$.

The solution vector of the matrix step gives (virtual) logarithms of (some of) the prime ideals $\leq L$ modulo $q$.
Using these logarithms $g^x \equiv h \pmod{p}$ can be solved via *descent* (later).

# Differences between factoring and the DLP

Fundamental difference:

For each integer there is just one factoring problem, whereas for each prime there are many DPLs.
DLP instances with the same prime $p$ share the three main NFS steps.
Applicable to cryptosystems in which the same prime $p$ is used by all parties (as is often featured in some standards).

# Differences between factoring and the DLP

Fundamental difference:

For each integer there is just one factoring problem, whereas for each prime there are many DPLs.
DLP instances with the same prime $p$ share the three main NFS steps.
Applicable to cryptosystems in which the same prime $p$ is used by all parties (as is often featured in some standards).

Differences between factoring-NFS and DLP-NFS:

- One has more freedom in polynomial selection for DLP-NFS (Joux-Lercier method).
- The matrix step modulo $q$ is about $\log_2 q$ times more complex than modulo 2.
- There are some other, minor differences.

# Extrapolating from RSA-768 to 768-bit DLP

RSA-768 timings:

| Main steps | time | wall clock time | memory | comments |
|---|---|---|---|---|
| Pol. selection | 40 years | 5 months | $< 10$ MB | low priority |
| Sieving | 1500 years | 2 years | 1-2 GB | very parallel |
| Matrix step ($193\text{M} \times 193\text{M}$) | 75 years | 4 months | 200 GB | only 8 tasks |

# Extrapolating from RSA-768 to 768-bit DLP

RSA-768 timings:

| Main steps | time | wall clock time | memory | comments |
|---|---|---|---|---|
| Pol. selection | 40 years | 5 months | $< 10$ MB | low priority |
| Sieving | 1500 years | 2 years | 1-2 GB | very parallel |
| Matrix step ($193\text{M} \times 193\text{M}$) | 75 years | 4 months | 200 GB | only 8 tasks |

Naive extrapolation to 768-bit DLP:

| Main steps | time | |
|---|---|---|
| Pol. selection | 40 years | |
| Sieving | 1500 years | |
| Matrix step | 50000 years | (about 767 times 75 years) |

# Rebalancing

Problem: How can the effort for the matrix step be reduced?

# Rebalancing

Problem: How can the effort for the matrix step be reduced?

Solution: By adapting parameters one looks for better (but rarer) relations, which are supposed to produce a smaller matrix.

(smaller factor bases, only two large primes per polynomial)

# Rebalancing

Problem: How can the effort for the matrix step be reduced?

Solution: By adapting parameters one looks for better (but rarer) relations, which are supposed to produce a smaller matrix.

(smaller factor bases, only two large primes per polynomial)

Consequences:

- The sieving time increases.
- The time for the matrix step decreases (smaller matrix).

# Rebalancing

Problem: How can the effort for the matrix step be reduced?

Solution: By adapting parameters one looks for better (but rarer) relations, which are supposed to produce a smaller matrix.

(smaller factor bases, only two large primes per polynomial)

Consequences:

- The sieving time increases.
- The time for the matrix step decreases (smaller matrix).
- An unexpected side-effect occurred:
  One can apply tricks to speed up sieving (halving the running time).

# Rebalancing

Problem: How can the effort for the matrix step be reduced?

Solution: By adapting parameters one looks for better (but rarer) relations, which are supposed to produce a smaller matrix.

(smaller factor bases, only two large primes per polynomial)

Consequences:

- The sieving time increases.
- The time for the matrix step decreases (smaller matrix).
- An unexpected side-effect occurred:
  One can apply tricks to speed up sieving (halving the running time).

Unrelated to the above,
one can (and we did) use the Joux-Lercier polynomial selection method.
It reduces the complexity of sieving and of the matrix step.

# Timeline

| | |
|---|---|
| Early 2015 | Experiments with 512-bit and 640-bit DLPs |
| | Polynomial selection for 768-bit DLP |

# Timeline

| | |
|---|---|
| Early 2015 | Experiments with 512-bit and 640-bit DLPs |
| | Polynomial selection for 768-bit DLP |
| May 2015 | Sieving started for 768-bit DLP |

# Timeline

| | |
|---|---|
| Early 2015 | Experiments with 512-bit and 640-bit DLPs |
| | Polynomial selection for 768-bit DLP |
| May 2015 | Sieving started for 768-bit DLP |
| August 2015 | First matrix built (about 80 million, far too big) |

# Timeline

| | |
|---|---|
| Early 2015 | Experiments with 512-bit and 640-bit DLPs |
| | Polynomial selection for 768-bit DLP |
| May 2015 | Sieving started for 768-bit DLP |
| August 2015 | First matrix built (about 80 million, far too big) |
| November 2015 | Feasible matrix (about 34 million) |

# Timeline

| | |
|---|---|
| Early 2015 | Experiments with 512-bit and 640-bit DLPs |
| | Polynomial selection for 768-bit DLP |
| May 2015 | Sieving started for 768-bit DLP |
| August 2015 | First matrix built (about 80 million, far too big) |
| November 2015 | Feasible matrix (about 34 million) |
| December 2015 | End of sieving, matrix size 23.5 million |
| | Matrix step started |

# Timeline

| | |
|---|---|
| Early 2015 | Experiments with 512-bit and 640-bit DLPs |
| | Polynomial selection for 768-bit DLP |
| May 2015 | Sieving started for 768-bit DLP |
| August 2015 | First matrix built (about 80 million, far too big) |
| November 2015 | Feasible matrix (about 34 million) |
| December 2015 | End of sieving, matrix size 23.5 million |
| | Matrix step started |
| May 2016 | Matrix step completed |

# NFS computation

Computation:

- It took about 1 year wall clock time (4th May 2015 - 18th May 2016).
- It took about 5000 core years.
- It could be reduced to 3000-4000 core years (perhaps further).

# NFS computation

Computation:

- It took about 1 year wall clock time (4th May 2015 - 18th May 2016).
- It took about 5000 core years.
- It could be reduced to 3000-4000 core years (perhaps further).

Result:

- We have (virtual) logarithms for about 23.5 million prime ideals.
- This leads to the logarithms for some of the small primes.

# Individual logarithms

Precomputation (not essential, but useful):

0. Extend 23.5 million logarithms to a bigger database,
   for example: all logarithms for prime ideals of norm $< 2^{35}$.
   (This took about 200 core years.)

# Individual logarithms

Precomputation (not essential, but useful):

0. Extend 23.5 million logarithms to a bigger database,
   for example: all logarithms for prime ideals of norm $< 2^{35}$.
   (This took about 200 core years.)

For each target $h$ compute $\log_g h$ as follows:

1. Write the logarithm of target $h$ as sum of
   logarithms of not too big prime ideals (e.g. $< 100$ bits)
   (average time 40 core hours, very parallel), and
2. descent prime ideals to smaller ones until all are in the database
   (average time 3 core hours, some parallelism).

# Individual logarithms

Precomputation (not essential, but useful):

0. Extend 23.5 million logarithms to a bigger database,
   for example: all logarithms for prime ideals of norm $< 2^{35}$.
   (This took about 200 core years.)

For each target $h$ compute $\log_g h$ as follows:

1. Write the logarithm of target $h$ as sum of
   logarithms of not too big prime ideals (e.g. $< 100$ bits)
   (average time 40 core hours, very parallel), and
2. descent prime ideals to smaller ones until all are in the database
   (average time 3 core hours, some parallelism).

This can be improved in various ways (rebalancing, improving software).

# Summary

- Comparison between factoring $N$ and the DLP for $p \quad (p \approx N)$: Although solving similarly-sized matrices is about $\log_2 p$ times more complex, solving the DLP is *not* $\log_2 p$ times harder than factoring.
- After having spent a few thousand core years for each "interesting" 768-bit prime, it is relatively easy to compute discrete logarithms in the corresponding prime fields.