About these slides

- These are slides covered in the Academic Year 2015-2016
- They will a priori not be covered this year
- Best usage : scan content to know what is in there, and consult later if you want to know more
- Please report any error / typo ! !
- Note that DLP algorithms is a very active research area today, hence the slides may already be outdated

Advanced Cryptography

DLP and Factoring Algorithms

Christophe Petit

University of Oxford

Discrete logarithms

- Given a cyclic group $(G, \circ)$ (written multiplicatively), a generator g of G and a second element $h \in G$, compute $k \in \mathbb{Z}_{|G|}$ such that $g^k = h$
- Trivial if $(G, \circ) = (\mathbb{F}_p, +)$. Why ?
- Recently broken if $(G, \circ) = (\mathbb{F}_{2^n}^*, *)$ (more generally if characteristic is not too big)
- Believed to be hard (to different extents) for $G = \mathbb{F}_p^*$ and for (well-chosen) elliptic/hyperelliptic curve groups

Integer factorization

- Given a composite number n, compute its (unique) factorization $n = \prod p_i^{e_i}$ where p_i are prime numbers
- Equivalently (why ?) : compute one non-trivial factor p_i
- Trivial if $n = p^e$
- Believed to be hard if $n = pq$ for well-chosen $p \neq q$

RSA and Diffie-Hellman

- DLP broken implies Diffie-Hellman broken
- Factorization broken implies RSA broken
- We don't know whether DH broken implies DLP broken
- We don't know whether RSA broken implies factorization broken
- Nevertheless, the best attacks against DH and RSA today are discrete log and factorization attacks

Related assumptions

- The cryptography literature includes many other, somewhat related assumptions
- Some of them are equivalent to DLP or factoring
- Some of them are strictly weaker/stronger
- Many interesting open problems
- These lectures : focus on DLP and factoring

Outline

References and Credits

- Joux, *Algorithmic Cryptanalysis*, Chapters 3,7,14,15
- Joux-Odlyzko-Pierrot, *The past, evolving present and future of discrete logarithms*
  Nice DLP algorithm picture is taken from there

Outline

Generic attacks

- DLP is trivial in some groups
- DLP seems harder in other groups
- Best attacks in a particular group often rely on specific properties of the group
- Can we find better groups ?
- How hard can DLP be in the best (hardest) groups ?

Group isomorphisms

- Any cyclic group $(G, \circ)$ of order n can be seen as $(\mathbb{Z}_n, +)$ in the following sense : there exists an invertible map $\varphi : G \to \mathbb{Z}_n$ such that $\forall x, y \in G$, we have

$$\varphi(x \circ y) = \varphi(x) + \varphi(y)$$

- Remark φ does not need to be efficiently computable
- Example : let g of order $p-1$ in $\mathbb{Z}_p^*$. Can define φ as sending any $h \in G$ to $\varphi(h) \in \mathbb{Z}_{p-1}$ such that $h = g^{\varphi(h)}$.
- Let $x' = \varphi(x)$ and $y' = \varphi(y)$. We have

$$\varphi^{-1}(x'+y') = \varphi^{-1}(\varphi(x)+\varphi(y)) = \varphi^{-1}(\varphi(x \circ y)) = x \circ y = \varphi^{-1}(x') \circ \varphi^{-1}(y')$$

DLP in the generic group model

- A DLP instance is generated in $(\mathbb{Z}_n, +)$, including a generator $g \in \mathbb{Z}_n$ and another element $h = kg \in \mathbb{Z}_n$
- A random invertible map $\theta : \mathbb{Z}_n \to \mathbb{Z}_n$ is chosen
- The map defines a group $(\mathbb{Z}_n, \circ)$ with

$$x \circ y = \theta\left(\theta^{-1}(x) + \theta^{-1}(y)\right)$$

- The attacker is NOT given g, h nor θ
- The attacker is given $\theta(g)$, $\theta(h)$ and access to **oracles**
  - $\mathcal{O}_1$: on input x, y, return $\theta\left(\theta^{-1}(x) + \theta^{-1}(y)\right)$
  - $\mathcal{O}_2$: on input x, return $\theta(-\theta^{-1}(x))$
- The attacker's goal is to compute k

Generic group model

- As θ is random, there is no special property of the group that can be exploited
- n itself is often hidden, and the attacker just receives bitstrings instead of $\mathbb{Z}_n$ elements (the size of n cannot be hidden)
- Some attacks are generic : they work for any group
  This includes exhaustive search, BSGS, Pollard's rho
- There exist much better attacks for finite fields
- Still no better attack for (well-chosen) elliptic curves

Exhaustive search

- Given $g, h \in G$ do the following

1: $k \leftarrow 1; h' \leftarrow g$
2: **if** $h' = h$ **then**
3: **return** k
4: **else**
5: $k \leftarrow k + 1$; $h' \leftarrow h'g$
6: Go to Step 2
7: **end if**

- Generic algorithm
- Time complexity $|G|$ in the worst case, $|G|/2$ on average
- Can we do better ?

Baby step, giant step (BSGS)

- Let $h = g^k$. You want to compute k.
- Let $N' = \lceil\sqrt{|G|}\rceil$
- There exist $0 \leq i, j < N'$ such that $k = jN' + i$

$$h = g^{jN'+i} \Leftrightarrow hg^{-jN'} = g^i$$

- Compute $L_B := \{g^i | i = 0, \ldots, N' - 1\}$
- Compute $L_G := \{hg^{-jN'} | j = 0, \ldots, N' - 1\}$
- Attack requires time and memory $O(\sqrt{|G|})$

Birthday paradox

- Suppose there are N_2 people in a room. What is the probability that two people have the same birthday ?
- How many people needed to have a probability larger than 50% ?
- Answer is **23** :

$$\Pr[\text{all distinct}] = 1 \cdot \frac{364}{365} \cdot \frac{363}{365} \cdot \ldots \cdot \frac{365-22}{365} < \frac{1}{2}$$

Birthday paradox

- Suppose you choose N_2 elements randomly in a set of N elements. What is the probability that two elements are equal ?
- How should N_2 be wrt N to have a probability larger than 50% ?
- Answer is $O(\sqrt{N})$:

$$\begin{aligned} \Pr[\text{all distinct}] &= 1 \cdot \frac{N-1}{N} \cdot \frac{N-2}{N} \cdot \ldots \cdot \frac{N-N_2+1}{N} \\ &\approx e^{-\frac{1}{N}} \cdot e^{-\frac{2}{N}} \cdot \ldots \cdot e^{-\frac{N_2-1}{N}} \\ &\approx e^{-\frac{N_2(N_2-1)}{N}} \end{aligned}$$

Taking $N_2 \approx \sqrt{N}$ ensures $1 - \Pr[\text{all distinct}]$ constant

Pollard's rho (iterative function)

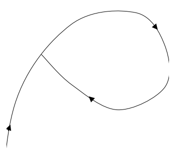
- Define G_1, G_2, G_3 of about the same size such that $G = G_1 \cup G_2 \cup G_3$ and $G_i \cap G_j = \{\}$
- Over $\mathbb{Z}_p^*$, can choose $G_1 = \{0, \ldots, \lfloor p/3 \rfloor\}$, $G_2 = \{\lfloor p/3 \rfloor + 1, \ldots, \lfloor 2p/3 \rfloor\}$, $G_3 = \{\lfloor 2p/3 \rfloor + 1, \ldots, p-2\}$
- Define a function $f : G \rightarrow G$ such that

$$\begin{cases} f(z) = zg & z \in G_1 \\ f(z) = z^2 & z \in G_2 \\ f(z) = zh & z \in G_3 \end{cases}$$

(original definition, other definitions possible)

Pollard's rho (intuition)

- Start from $g_0 := g$ and apply f recursively to get g_i
- By the way f is defined, we can keep track of a_i, b_i such that $g_i = g^{a_i} h^{b_i}$
- If f is "random enough", obtain random elements in G and a collision after $O(\sqrt{|G|})$ elements
- Collision gives DLP solution



Pollard's rho (simplest version)

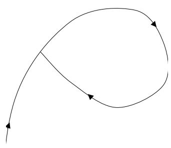
$N \leftarrow \lceil \sqrt{|G|} \rceil$
$a \leftarrow 1; b \leftarrow 0; \tilde{h} \leftarrow g; L \leftarrow \{(a, b, \tilde{h})\}$
for $k \in \{2, \ldots, N\}$ **do**
  if $\tilde{h} \in G_1$ **then** $a \leftarrow a + 1; \tilde{h} \leftarrow \tilde{h}g$
  if $\tilde{h} \in G_2$ **then** $a \leftarrow 2a; b \leftarrow 2b; \tilde{h} \leftarrow (\tilde{h})^2$
  if $\tilde{h} \in G_3$ **then** $b \leftarrow b + 1; \tilde{h} \leftarrow \tilde{h}h$
  $L \leftarrow L \cup \{(a, b, \tilde{h})\}$
end for
Find distinct $(a_i, b_i, \tilde{h}) \in L$, $i = 1, 2$
if no such elements **then abort**
return $-(a_1 - a_2)/(b_1 - b_2) \bmod |G|$

Pollard's rho analysis

- Correctness :
  - Every $(a, b, \tilde{h})$ in the list satisfies $\tilde{h} = g^a h^b$
  - $g^{a_1} h^{b_1} = g^{a_2} h^{b_2}$ implies $h = g^{-\frac{a_1 - a_2}{b_1 - b_2}}$
- Time and memory costs $N \approx \sqrt{|G|}$
- Good probability of success by birthday's paradox

Pollard's rho (improvement)

- Let $(L_1, L_1 + L_2)$ be the indices of first collision
- Then $(L_1 + j, L_1 + kL_2 + j)$ also collide
- For j, k such that $L_1 + j = kL_2$, we have $L_1 + kL_2 + j = 2(L_1 + j)$
- Now search for $(a_i, b_i, \tilde{h}_i)$ and $(a_{2i}, b_{2i}, \tilde{h}_{2i})$ such that $\tilde{h}_i = \tilde{h}_{2i}$
- Only requires constant size memory



Pohlig-Hellman

- Assume $|G| = n_1 n_2$ and let g a generator of G
- $h = g^k$ implies $h^{n_1} = (g^{n_1})^k$ where g^{n_1} generates a subgroup of order n_2
- Solving DLP in that subgroup gives k mod n_2
- Repeating for each factor and using CRT gives k

Pohlig-Hellman (example)

- Let $G = \mathbb{Z}_{13}^*$, let $g = 2$ and let $h = 7$
- We have $|G| = 12 = 2^2 \cdot 3$
- Recover k mod 2 by solving $(2^6)^k = 7^6 \bmod 13 \Leftrightarrow (-1)^k = -1 \bmod 13 \Leftrightarrow k = 1 \bmod 2$
- Write $k = 1 + 2k'$. Recover k mod 4 by solving $(2^3)^{1+2k'} = 7^3 \bmod 13 \Leftrightarrow (-1)^{k'} = -1 \bmod 13 \Leftrightarrow k' = 1 \bmod 2 \Leftrightarrow k = 3 \bmod 4$
- Recover k mod 3 by solving $(2^4)^k = 7^4 \bmod 13 \Leftrightarrow (3)^k = 9 \bmod 13 \Leftrightarrow k = 2 \bmod 3$
- Use CRT to deduce $k = 11 \bmod 12$

Outline

Outline

Generic DLP algorithms

Index Calculus for DLP : introduction
- Overview
- Example : Adleman's algorithm
- The linear algebra part

Subexponential DLP algorithms

Quasi-polynomial DLP algorithm

Factoring algorithms

Elliptic Curve Discrete Logarithm Problem

Discrete Logarithms over finite fields

- **Discrete Logarithm Problem (DLP)**
  Given G a finite cyclic group, given g a generator of G, and given $h \in G$, find k such that $h = g^k$
- Believed to be a hard problem when G is the multiplicative group of a well-chosen field
- (Formal definition of "hard" involves families of fields,. . .)

Fields used in cryptography

- $\mathbb{F}_p^*$ where p is prime : most used, believed to be secure
- $\mathbb{F}_{p^n}^*$ where p is prime and n is small (typically up to 12) : used in *pairing* applications
- $\mathbb{F}_{2^n}^*$ or $\mathbb{F}_{3^n}^*$ where n is a product of small primes : should be avoided (Pohlig-Hellman attack)
- $\mathbb{F}_{2^n}^*$ or $\mathbb{F}_{3^n}^*$ for arbitrary n : should now also be avoided, suggested before 2013 for efficiency reasons
- Remark : typically work over a prime order subgroup of $\mathbb{F}_p^*$ or $\mathbb{F}_{p^n}^*$, otherwise problems such as *decisional Diffie-Helman* are easy

L notation

$$L_Q(\alpha; c) = \exp(c(\log Q)^{\alpha}(\log\log Q)^{1-\alpha})$$

- Q is the size of the field
- $\alpha = 0 \Rightarrow L_Q(\alpha; c) = (\log Q)^c$ polynomial
- $\alpha = 1 \Rightarrow L_Q(\alpha; c) = Q^c$ exponential

Playing with L notation

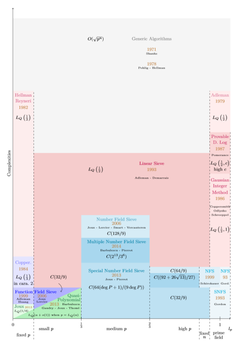
$$L_Q(\alpha; c) = \exp(c(\log Q)^{\alpha}(\log\log Q)^{1-\alpha})$$

- Approximation : ignore constant and $\log\log$ factors, write $L_Q(\alpha)$ (but beware they are very relevant in practice !)
- $L_Q(\alpha)L_Q(\beta) \approx L_Q(\max(\alpha, \beta))$
- $L_Q(\alpha, c)^k = L_Q(\alpha, kc)$ if k is constant
- $L_Q(\alpha, c)^k = L_Q(\alpha + \beta, c)$ if $k = (\log Q)^{\beta}$

State-of-the-art and History

- Write $Q = q^n$ (with q a prime power)
- State-of-the-art depends on relative size of q and n
- See Joux, Odlyzko, Pierrot. *The past, evolving present and future of discrete logarithms* `www-polsys.lip6.fr/~pierrot/papers/Dlog.pdf`

DLP algorithms for finite fields



Index calculus

- Generic framework to solve discrete logarithm problems, but some steps are group-specific
- Let g, h a DLP problem
- Define a *factor basis* $\mathcal{F} \subset G$, ensuring $\mathcal{F}$ contains a generator (most elements in G are generators)
- Can assume $g \in \mathcal{F}$, otherwise do the following :
  - Pick a generator $g' \in \mathcal{F}$
  - Compute a such that $g = (g')^a$
  - Compute b such that $h = (g')^b$
  - Compute $k = b/a \bmod |G|$
- Remark : size of $\mathcal{F}$ will be optimized for efficiency

Index calculus

- Find about $|\mathcal{F}|$ relations between factor basis elements

$$\mathcal{R}_j : \quad \prod_{f_i \in \mathcal{F}} f_i^{a_{i,j}} = 1$$

(the algorithm to compute the relations is group-specific)

- Deduce

$$\sum_{f_i \in \mathcal{F}} a_{i,j} \log_g f_i = 0$$

or

$$\begin{pmatrix} a_{1,1} & \dots & a_{|\mathcal{F}|,1} \\ \vdots & & \vdots \\ a_{1,|\mathcal{F}|} & \dots & a_{|\mathcal{F}|,|\mathcal{F}|} \end{pmatrix} \begin{pmatrix} \log_g f_1 \\ \vdots \\ \log_g f_{|\mathcal{F}|} \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}$$

Index calculus

- Use linear algebra to compute all $\log_g f_i$, the discrete logarithms of factor basis elements
- Deduce the discrete logarithm of h (This part is group-specific and may involve several steps)
- Remarks :
  - Relations often involve few elements, hence linear algebra is sparse
  - In some cases, h is included in the factor basis and the last step is avoided : linear algebra produces $\log_g h$

Outline

Leonard Adleman

Example : a naive index calculus for $\mathbb{F}_p^*$

- DLP : given $g, h \in \mathbb{F}_p^*$, find k such that $h = g^k$
- Factor basis made of **small primes**

$$\mathcal{F}_B := \{\text{primes } p_i \leq B\}$$

- **Relation search**
  - Compute $r_j := g^{a_j} h^{b_j}$ for random $a_j, b_j \in \{1, \ldots, p-1\}$
  - **If** all factors of r_j are $\leq B$, we have a relation

$$g^{a_j} h^{b_j} = \prod_{p_i \in \mathcal{F}} p_i^{e_{i,j}}$$

- **Linear algebra** produces $g^a h^b = 1$

Size of the factor basis

- By the prime number theorem,

$$|\{\text{primes } p_i \leq B\}| \approx \frac{B}{\ln B}$$

Smooth numbers

- An integer number is B-smooth if all its prime factors are smaller than B
- Define $\Psi(N, B) = \#\{B\text{-smooth numbers} \leq N\}$
- Let $u = \log N / \log B$. We have

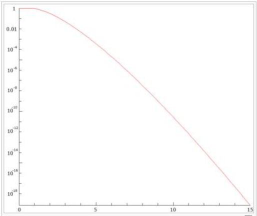
$$\frac{\Psi(N, B)}{N} = \rho(u) + O\left(\frac{1}{\log B}\right)$$

- Here ρ is the *Dickman-de Bruijn* function with

$$\rho(u) \approx u^{-u}$$

Dickman-de Bruijn function ρ

- The *Dickman-de Bruijn* function ρ satisfies $\rho(u) \approx u^{-u}$



The Dickman–de Bruijn function ρ(u) plotted on a logarithmic scale. The horizontal axis is the argument u, and the vertical axis is the value of the function. The graph nearly makes a downward line on the logarithmic scale, demonstrating that the logarithm of the function is quasilinear.

$\log \rho \approx -u \log u$
(picture source : Wikipedia)

Naive analysis of naive index calculus

- Choose $\log B \approx (\log p)^{1/2}$
- $|\mathcal{F}| \approx B/\log B \approx 2^{(\log p)^{1/2} - (\log\log p)^{-1/2}} \approx 2^{(\log p)^{1/2}}$
- $u = \log p / \log B \approx (\log p)^{1/2}$
- $\rho(u) = (\log p)^{-1/2(\log p)^{1/2}} \approx 2^{-1/2(\log p)^{1/2}(\log\log p)}$
- Number of random trials to get $|\mathcal{F}|$ relations is

$$\approx |\mathcal{F}|\rho(u)^{-1} \approx 2^{(1/2+o(1))(\log p)^{1/2}(\log\log p)}$$

- Each trial has polytime complexity in $\log p$
- Linear algebra cost is $|\mathcal{F}|^{\omega} \approx 2^{\omega(\log p)^{1/2}}$
- Total cost dominated by relation search
- $B \approx L_p(1/2; c)$ leads to slighly better cost $L_p(1/2; c')$

Playing with L notation (2)

$$L_Q(\alpha; c) = \exp(c(\log Q)^{\alpha}(\log\log Q)^{1-\alpha})$$

- Probability that an element of size $L(\alpha)$ is $L(\beta)$ smooth is

$$(L(\alpha - \beta))^{-1}$$

- If c is constant, the probability that an element of size B is B/c-smooth is constant

Same algorithm for $\mathbb{F}_{2^n}^*$

- DLP : given $g, h \in \mathbb{F}_{2^n}^*$, find k such that $h = g^k$
- Factor basis made of **small "primes"**

$$\mathcal{F}_B := \{\text{irreducible } f(X) \in \mathbb{F}_2[X] | \deg(f) \leq B\}$$

- **Relation search**
  - Compute $r_j := g^{a_j}h^{b_j}$ for random $a_j, b_j \in \{1, \ldots, p-1\}$
  - Factor $r_j \in \mathbb{F}_2[X]$ with Berlekamp's algorithm
  - If all factors $\in \mathcal{F}_B$, we have a relation $g^a h^b = \prod_{f_i \in \mathcal{F}} f_i^{e_i}$
- **Linear algebra** produces $g^a h^b = 1$

Outline

Linear algebra

- Given matrix M and vector x, find all y such that $My = x$

Gaussian elimination

- Observation : if $My = x$ then for any invertible N, we have $NMy = Nx$
- In particular, this is true when N is a matrix which
  - Swaps two rows of M
  - Multiplies one row by an invertible constant
  - Adds a multiple of one row of M to another row of M
- Gaussian elimination repeats these operations until the resulting matrix is upper triangular

Gaussian elimination

- Algorithm when M is invertible

```
1: for each column i, from i = 1 to n do
2:     Find a nonzero element in this column
3:     Swap the row of this element with row i
4:     for each row j below row i do
5:         Let c := −M_{j,i}/M_{i,i}
6:         Add c times row i to row j
           to erase the value in (j, i)
7:     end for
8: end for
```

- Adapt step 2 otherwise
- Cost is $O(n^3)$ multiplications

Resolution from Gaussian form

- Algorithm when M is invertible
  1: **for** each column i from n to 1 **do**
  2: Recover value of unknown i, using equation i and all values of previously computed unknowns $j > i$
  3: **end for**
- Adapt to determine the afine space of solutions $v + \ker M$ otherwise
- Cost is $O(n^2)$ multiplications
- Can be used to invert M in $O(n^3)$ multiplications

Sparse linear algebra

- A matrix is **sparse** if each row contains a small number of nonzero elements
- Can store larger size matrices by storing only $(i, j, M_{i,j})$ for nonzero elements $M_{i,j}$
- Gaussian elimination will kill the sparsity quickly
- Two approaches for sparse matrices :
  - Structured Gaussian elimination
  - Algorithms based on matrix-vector multiplications

Structured Gaussian elimination

- Consider the linear system $My = x$
- For the matrices M occurring in index calculus :
  - Each row contains few elements
  - The first columns contain much more elements than the last ones
- Structured Gaussian elimination involves several tricks such as removing variables that only appear once or twice
- Used as preprocessing to reduce the size in practice
- Heuristic

Lanczos algorithm

- If M is invertible, $My = x \Leftrightarrow M^t My = M^t x$ hence we can assume M is symmetric positive definite defining a scalar product $(x, y)_M := xMy^t$
- Lanczos is iterative : over the real/complex numbers, the algorithm can be stopped before the end with a reasonable approximation of the solution
- First compute a basis $\{v_i\}$ of orthogonal vectors with respect to the scalar product $(*, *)_M$,
- Then compute $\sum_{i=1}^{n} (x, v_i) v_i = \sum_{i=1}^{n} (y, v_i)_M v_i = y$
- Second part involves $O(n)$ matrix-vector multiplications, each one at $O(n)$ cost if each row contains $O(1)$ elements

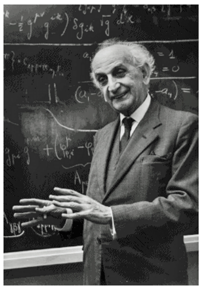
Computing the orthogonal basis

- Start from a random w_1 and $v_1 = w_1/||w_1||_M$
- Then heuristic modification of Gram-Schmidt algorithm
  1. $w_{i+1} = Mv_i$
  2. $w'_{i+1} = w_{i+1} - \sum_{j=1}^{i}(w_{i+1}, v_j)_M \cdot v_j$
  3. $v_{i+1} = w'_{i+1}/||w'_{i+1}||_M$
- Second step is in fact

$$w'_{i+1} = w_{i+1} - (w_{i+1}, v_i)_M \cdot v_i - (w_{i+1}, v_{i-1})_M \cdot v_{i-1}$$

- Likely to converge to a basis $\{v_1, \ldots, v_n\}$ over the reals ; needs some adjustment for small characteristic finite fields

Cornelius Lanczos



Wiedemann algorithm

- Reconstruct the **minimal polynomial** of M : smallest degree polynomial f such that $f(M) = 0$
- If $f(\alpha) = \sum_{i=0}^{d} f_i \alpha^i$, then $I = -\frac{1}{f_0}\sum_{i=1}^{d} f_i M^i$ then

$$x = -\frac{1}{f_0}\sum_{i=1}^{d} f_i M^i x = M\left(-\frac{1}{f_0}\sum_{i=1}^{d} f_i M^{i-1} x\right)$$

- We deduce y such that $My = x$

Wiedemann algorithm (2)

- Main idea to compute minimal polynomial :
  - Construct $(a, M^i x)$ for a random vector a and $i = 0, \ldots, 2n-1$
  - Use Berlekamp-Massey's algorithm to compute the linear recurrence in this sequence
- The whole algorithm requires $O(n)$ matrix-vector products
- Recent discrete log records use Block Wiedemann `http://caramel.loria.fr/p180.txt`

Outline

Generic DLP algorithms

Index Calculus for DLP : introduction

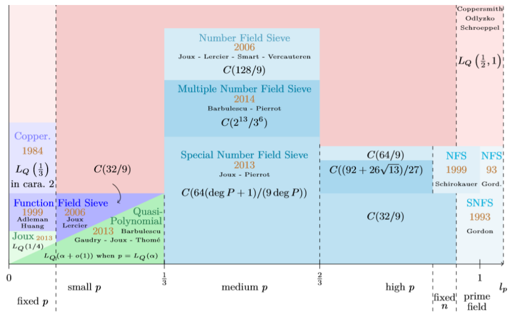
Subexponential DLP algorithms
- Coppersmith
- Function Field Sieves
- Number Field Sieves

Quasi-polynomial DLP algorithm

Factoring algorithms

Elliptic Curve Discrete Logarithm Problem

Further subexponential DLP algorithms



Source : `www-polsys.lip6.fr/~pierrot/papers/Dlog.pdf`

Outline

Generic DLP algorithms

Index Calculus for DLP : introduction

Subexponential DLP algorithms
- Coppersmith
- Function Field Sieves
- Number Field Sieves

Quasi-polynomial DLP algorithm

Factoring algorithms

Elliptic Curve Discrete Logarithm Problem

Don Coppersmith

Remember : basic algorithm for $\mathbb{F}_{2^n}^*$

- DLP : given $g, h \in \mathbb{F}_{2^n}^*$, find k such that $h = g^k$
- Factor basis made of **small "primes"**

$$\mathcal{F}_B := \{\text{irreducible } f(X) \in \mathbb{F}_2[X] \mid \deg(f) \leq B\}$$

- **Relation search**
  - Compute $r_j := g^{a_j} h^{b_j}$ for random $a_j, b_j \in \{1, \ldots, p-1\}$
  - Factor $r_j \in \mathbb{F}_2[X]$ with Berlekamp's algorithm
  - If all factors $\in \mathcal{F}_B$, we have a relation $g^a h^b = \prod_{f_i \in \mathcal{F}} f_i^{e_i}$
- **Linear algebra** produces $g^a h^b = 1$

Coppersmith's algorithm for $\mathbb{F}_{2^n}$

- Idea : reduce factor basis to polynomials of degree $n^{1/3}$ (vs. $n^{1/2}$) by ensuring all r_j have degree $n^{2/3}$ (vs. n)
- We have $\mathbb{F}_{2^n} \approx \mathbb{F}_2[x]/(p(x))$ for any irreducible p
  Choose $p(x) = x^n + q(x)$ where $\deg q \leq n^{2/3}$
- Let $k = 2^e \approx n^{1/3}$, let $d \approx n^{1/3}$
- Let $h \approx n^{2/3}$ least integer larger than n/k
- Let $r(x) = x^{hk} \bmod p(x) = q(x)x^{hk-n}$
  with $\deg r < k + \deg q \approx n^{2/3}$

Coppersmith's algorithm for $\mathbb{F}_{2^n}$

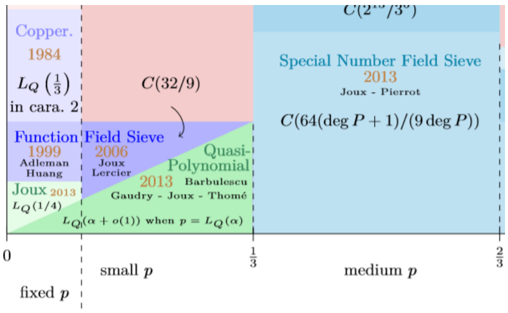
- Factor basis are elements with degree smaller than d, where d smallest integer $\geq n^{1/3}$
- Relations will be of the form $d(x) = (c(x))^k$
  for c, d smooth, where c constructed in a special way
- Relation search
  - Take $a(x)$ and $b(x)$ coprime with degrees d
  - Take $c(x) = a(x)x^h + b(x)$ degree $O(n^{2/3})$
  - Take $d(x) = (c(x))^k \bmod p$
  - We have $d(x) = r(x)(a(x))^k + (b(x))^k$ degree $O(n^{2/3})$
  - If both c and d are smooth, we get a relation
  - Probability $O(2^{-n^{1/3}-\epsilon})$

Individual logarithms

- For increasing i, until m_i and n_i are smooth enough
  - Use continued fractions/ Euclide algorithm to write $h(x)x^i = m_i(x)/n_i(x)$ with $\deg m_i, \deg n_i \leq n/2$
  - Check smoothness of m_i and n_i
  - Continue until both are $O(n^{2/3})$ smooth
- For each factor m
  - Choose $a(x)$ and $b(x)$ coprime random such that $m|c$ where $c(x) = a(x)x^h + b(x)$
  - Let $d(x) = (c(x))^k \bmod p(x)$ as above
  - If d and c/m are smooth enough, we either iterate on all (smaller degree) factors or we write m in the factor basis

Outline

Function Field Sieves



Source : www-polsys.lip6.fr/~pierrot/papers/Dlog.pdf

Adleman-Huang

- We want to solve DLP in $\mathbb{F}_{p^n}$, where p is constant
- Set smoothness bound $d \approx n^{1/3}$
- Define $f(x) = x^n + q(x)$ where $\deg q < n^{2/3}$
- Let $k \approx n^{1/3}$, let h least integer larger than n/k, and let $\delta = hk - n$
- Let $m(x) = x^h$ and $H(x,y) = y^k + x^\delta q(x)$
- We have a homomorphism

$$\Phi : \frac{\mathbb{F}_p[x,y]}{(H(x,y))} \to \frac{\mathbb{F}_p[x]}{(f(x))} : (x,y) \to (x, m(x))$$

Factor basis and relations

- Factor basis is $\mathcal{F} = \mathcal{F}_1 \cup \mathcal{F}_2$ where
  1. $\mathcal{F}_1$ contains all irreducible polynomials of degree at most d over $\mathbb{F}_p$,
  2. $\mathcal{F}_2 = \{r + ms \mid N(r + ys) \in \mathcal{F}_1\}$
     (Here $N(r + ys) = r^k H(x, -s/r)$ is function field norm)
- To find a relation, take random couples of polynomials (a, b) both of degrees about $n^{1/3}$, until both
  1. $am + b$ is d-smooth
  2. $N(ay + b)$ is d-smooth

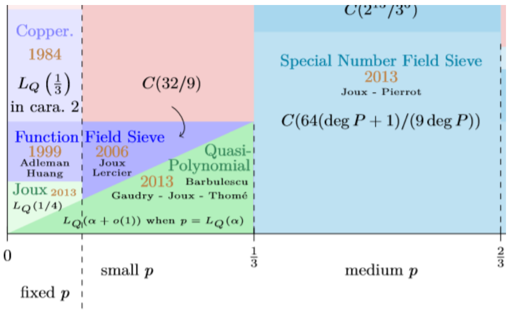
Adleman-Huang (2)

- From each such couple deduce a relation

$$\sum_{P_i \in \mathcal{F}_1} e_i \log P_i = \sum_{Q_i \in \mathcal{F}_2} f_i \log Q_i$$

- Remark : $\deg(am + b) \approx \deg N(ay + b) \approx n^{2/3}$ so probability that a random couple (a, b) gives a relation is about $L_p(1/3)^{-1}$
- Individual logarithms as in Coppersmith's algorithm

Joux-Lercier



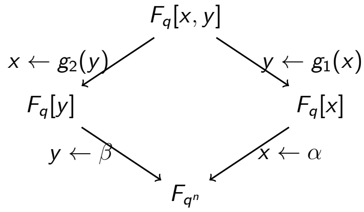
Source : www-polsys.lip6.fr/~pierrot/papers/Dlog.pdf

Joux-Lercier

- We want to solve DLP in $\mathbb{F}_{q^n}$, where $q = L_{q^n}(1/3)$
- Find polynomials g_1, g_2 of degrees $d_1, d_2 \approx n^{1/2}$ over $\mathbb{F}_q$ s.t. $g_2(g_1(x)) + x$ has an irreducible factor I of degree n
- Letting $y = g_1(x)$, we see that $g_1(-g_2(y)) - y$ has an irreducible factor I' of degree n
- If $\alpha \in \mathbb{F}_{p^n}$ is a root of I then $\beta = g_1(\alpha)$ is a root of I'
- If $\beta \in \mathbb{F}_{p^n}$ is a root of I' then $\alpha = -g_2(\beta)$ is a root of I

Joux-Lercier

- We have the following commutative diagram of homomorphisms



Factor basis and relations

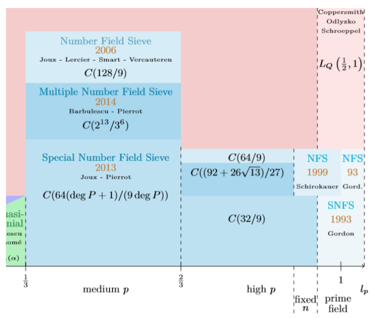
- Factor basis is $\mathcal{F} = \mathcal{F}_1 \cup \mathcal{F}_2$ where
  - $\mathcal{F}_1 =$ images of degree 1 polynomials in $F_q[y]$ by $y \leftarrow \beta$
  - $\mathcal{F}_2 =$ images of degree 1 polynomials in $F_q[x]$ by $x \leftarrow \alpha$
- To find relations, pick random $h(x, y) = xy + bx + cy + d$ until both $h(g_2(y), y)$ and $h(x, g_1(x))$ split completely
- Splitting probability $\frac{1}{d_1!} \cdot \frac{1}{d_2!} \approx 2^{-n^{1/2} \log n} \approx L_{q^n}(1/3)$
- Size of $\mathcal{F}$ is also $L_{q^n}(1/3)$

Individual logarithms

- Let $h \in \mathbb{F}_q[x]$ for which we want to compute DL
- Compute $x^i h(x)$ until the result is moderately smooth
- For each factor h', find $a, b \in \mathbb{F}_q[x]$ such that
  - Degrees not too large, about deg h'
  - $h'(x) \mid (a(x)g_1(x) + b(x))$
  - $(a(x)g_1(x) + b(x)) \;/\; h'(x)$ smoother enough
  - $a(g_2(y))y + b(g_2(y))$ smooth enough
- Alternatively decrease the factors on each side, until all factors on both sides are linear

Outline

Number Field Sieves



Source : www-polsys.lip6.fr/~pierrot/papers/Dlog.pdf

Gordon

- We want to solve DLP in $\mathbb{F}_p$, where p is prime
- Choose $m \approx L_p(2/3)$
- Let $p = \sum_{i=0}^{d} f_i m^i$ with $d \approx (\log p)^{1/3}$
- Let $f(x) = \sum_{i=0}^{d} f_i x^i$
- We have a ring homomorphism

$$\varphi \ : \ \mathbb{Q}[x]/(f(x)) \to \mathbb{F}_p \ : \ x \to m$$

Factor basis and relations

- Let $B \approx L_p(1/3)$ be a smoothness bound
- Factor basis is $\mathcal{F} = \mathcal{F}_1 \cup \mathcal{F}_2$ where
  - $\mathcal{F}_1 = \{\text{primes smaller than } B\}$
  - $\mathcal{F}_2 = \{\text{degree 1 prime ideals } \nu \mid N(\nu) \in \mathcal{F}_1\}$
- Search for pairs (a, b) with $a \approx b \approx L_p(1/3)$ such that $a + bm \in \mathcal{F}_1$ and $a + bx \in \mathcal{F}_2$
- Note that
  $a + bm \approx N(a + bx) = (-b)^d f(-a/b) \approx L_p(2/3)$
  so smoothness probability is $L_p(1/3)$

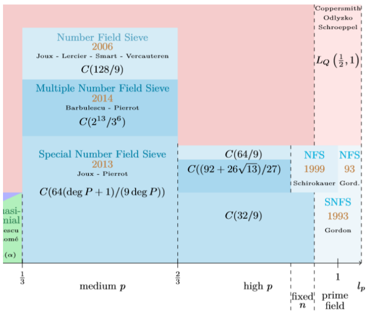
Individual logarithms

- Suppose we want DL of a particular h
- First compute $x^i h$ until the result is $L_p(2/3)$ smooth
- For each factor h_i,
  - Generate $L_p(1/3)$-smooth $\ell_i \approx h_i$, let $m_i = h_i \ell_i$,
    let $f_i(x)$ such that $f_i(m_i) = 0 \bmod p$,
    until $N_i(x) = f_i(0)$ is $L_p(1/3)$ smooth
  - Search for pairs (a, b) with $a \approx b \approx L_p(1/3)$ such that $a + bm_i \in \mathcal{F}_1$ and $N_i(a + bx)$ is $L_p(1/3)$ smooth.
    Repeat and eliminate factors not in $\mathcal{F}_1$

Technicalities

- Need to cancel units appearing in the relations
  $\Rightarrow$ add these units to the factor bases
- If the class number of $\mathbb{Q}[x]/(f(x))$ is $h > 1$ then need to remove non-principal ideals from the relations
  $\Rightarrow$ implicitly take h powering of the equations to get principal ideals

Joux-Lercier-Smart-Vercauteren



Source : www-polsys.lip6.fr/~pierrot/papers/Dlog.pdf

Joux-Lercier-Smart-Vercauteren

- We want to solve DLP in $\mathbb{F}_{p^n}$, where $p = L_{p^n}(2/3)$
- Choose $f_1 \in \mathbb{Z}[x]$ of degree n with small coefficients, with a root m modulo p
- Let $f_2 = f_1 + p$
- Define number fields $K_i = \mathbb{Q}[x]/(f_i(x))$
- We have two homomorphisms

$$\varphi_i : K_i \to \mathbb{F}_{p^n} : x \to m$$

Factor basis and relations

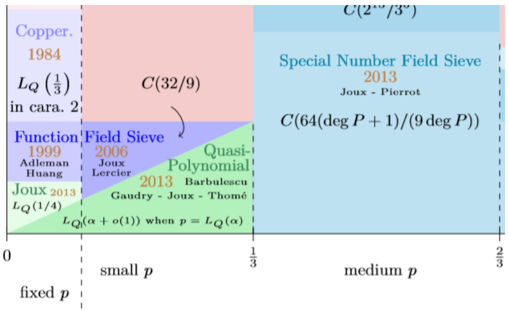
- Let $B \approx L_{p^n}(1/3)$ be a smoothness bound
- Let $\mathcal{F}_0 = \{$primes smaller than $B\}$
- Factor basis is $\mathcal{F} = \mathcal{F}_1 \cup \mathcal{F}_2$ where
  - $\mathcal{F}_1 = \{c + dm \mid N_1(c + dx) = (-d)^k f_1(-c/d) \in \mathcal{F}_0\}$
  - $\mathcal{F}_2 = \{c + dm \mid N_2(c + dx) = (-d)^k f_2(-c/d) \in \mathcal{F}_0\}$
- Search for pairs (a, b) with $a \approx b \approx L_{p^n}(1/3)$ such that both $N_i(a + bx) \in \mathcal{F}_0$
- Note that $N_i(a + bx) \approx L_{p^n}(2/3)$ so smoothness probability is $L_p(1/3)$

Remarks

- Individual logarithms as in Gordon, alternating descent in K_1 and K_2
- If K_i has a non-trivial automorphism group $Aut(K)$ (for example if it is Galois) then corresponding part of factor basis can be reduced by a factor $\#Aut(K)$
- Multiple number field sieve uses more than 2 number fields in parallel

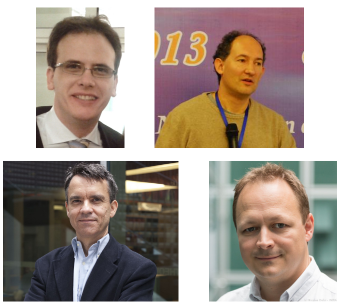
Outline

Quasi-polynomial DLP algorithm !



Source : www-polsys.lip6.fr/~pierrot/papers/Dlog.pdf

Barbulescu, Gaudry, Joux, Thomé



Sparse medium subfield representation

- A finite field K admits a sparse medium subfield representation if
  - $K = \mathbb{F}_{q^{2k}}$ for some prime power q
  - There exist $h_0, h_1 \in \mathbb{F}_{q^2}[X]$ with small degrees, such that $X^q h_1(X) - h_0(X)$ has a degree k irreducible factor I
- In practice we can find h_1, h_2 of degrees at most 2
- The polynomial I is used to define $\mathbb{F}_{q^{2k}} = \mathbb{F}_{q^2}[X]/(I(X))$
- Elements in such field will be seen as polynomials of degree less than k over $\mathbb{F}_{q^2}$

Quasi-polynomial DLP algorithm !

- If K admits a sparse medium subfield representation then (initially under various heuristics, now getting cleaner) any discrete logarithm in K can be computed in time bounded by $\max(q,k)^{O(\log k)}$
- If $q \approx k$ then $q = O(\log|K|)$ hence complexity $q^{O(\log q)} = 2^{O((\log\log|K|)^2)}$ quasi-polynomial in $\log|K|$
- If $|K| = p^n$ with characteristic $p = (\log|K|)^{O(1)}$ then set $q = p^{\lceil \log_p n \rceil}$ and work in extension field $L = \mathbb{F}_{q^{2n}}$, still quasi-polynomial
- If $q = L_{q^{2k}}(\alpha)$ then complexity $L_{q^{2k}}(\alpha)^{O(\log\log q^{2k})}$

Key proposition

Let $K = \mathbb{F}_{q^{2k}}$ with a sparse medium subfield representation. Under various heuristics,

1. There is an algorithm (polynomial time in q and k) which given an element of K as a polynomial $P \in \mathbb{F}_{q^2}[X]$ with $2 \leq \deg P \leq k-1$, returns an expression with at most $O(q^2 k)$ terms
$$\log P = e_0 \log h_1 + \sum e_i \log P_i$$
where $\deg P_i \leq \lceil \frac{1}{2} \deg P \rceil$ and $e_i \in \mathbb{Z}$
2. There is an algorithm (polynomial time in q and k) which returns $\log h_1$ and $\log(X+a)$ for all $a \in \mathbb{F}_{q^2}$

Using the Key proposition

- Given $P \in K$ we use first part to obtain
$$\log P = e_0 \log h_1 + \sum e_i \log P_i$$
where $\deg P_i \leq \lceil \frac{1}{2} \deg P \rceil$
- Apply first part recursively on each P_i
- Eventually
$$\log P = e_0 \log h_1 + \sum_{a \in \mathbb{F}_{q^2}} e_a \log(X+a)$$
- Apply second part to get $\log P$

Using the Key proposition (2)

- The procedure constructs a tree with arity $O(q^2 k)$ and $O(\log k)$ levels
- Number of nodes is $(q^2 k)^{O(\log k)}$
- Each node has a cost polynomial in k and q

Main ideas in Key proposition

- Systematic equation
$$X^q - X = \prod_{\alpha \in \mathbb{F}_q} (X - \alpha)$$
- Sparse field representation
$$I | (h_1 X^q - h_0) \Rightarrow X^q = \frac{h_0}{h_1} \bmod I$$
- Replace X by $m \cdot P$ in systematic equation, where
$$m \cdot P = \frac{aP + b}{cP + d} \quad \text{and} \quad m = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in SL(2, \mathbb{F}_{q^2})$$

Sketch of the algorithm

- Given P, substitute X by $m \cdot P$ for various m, so that products $P(X) - \alpha$ appear on the RHS
- Use the sparse field representation to reduce the degree on the LHS to about the degree of P
- Keep the relation if all factors of the LHS have degree smaller than $\lceil \frac{1}{2} \deg P \rceil$
- Combine the relations with linear algebra to eliminate all factors $P(X) - \beta$ with $\beta \neq 0$
- For second part : take $P(X) = X$

Remarks

- Relations obtained are identical for all $m = \lambda m'$ with $\lambda \in \mathbb{F}_{q^2}$ and $m \in SL(2, \mathbb{F}_q)$, and more generally we pick m in distinct cosets of $PGL(\mathbb{F}_{q^2})/PGL(\mathbb{F}_q)$
- Probability that a random polynomial of degree D is $D/2$-smooth is constant
- Analysis involves several heuristic assumptions ;
they are likely to be fine, if not then we are likely to refine them and deduce a better algorithm

Outline

Integer factorization

- Given a composite number n, compute its (unique) factorization $n = \prod p_i^{e_i}$ where p_i are prime numbers
- Equivalently : compute one non-trivial factor p_i
- We will assume $n = pq$, where p and q are primes

Factorization vs Discrete logarithms

- Discrete logarithm and factoring algorithms are similar
- Exceptions (?)
  - Quasi-polynomial time algorithm for discrete logarithms in small to medium characteristic
  - Elliptic curve factorization method
- Hardness of large characteristic field discrete logarithms and integer factorization is comparable today

Outline

Generic DLP algorithms

Index Calculus for DLP : introduction

Subexponential DLP algorithms

Quasi-polynomial DLP algorithm

Factoring algorithms
- Basic Algorithms and Quadratic Field Sieve
- (General) Number Field Sieve
- Elliptic Curve Factorization Method

Elliptic Curve Discrete Logarithm Problem

Sieve of Eratosthenes

- Compute all primes up to $\sqrt{n}$ using a sieve
- Try to factor n by each of them
- Complexity $O(\sqrt{n})$

- Remark : sieve can also be used to quickly find all smooth numbers in an interval

Pollard's rho

- Idea : find x and y such that $\gcd(x-y,n)=p$ in other words $x=y \bmod p$ but $x \neq y \bmod n$
- Define some "pseudorandom" iteration function f
- Compute iterates x_i and x_{2i}
- Simultaneously compute $\gcd(x_i - x_{2i}, n)$
- By birthday's paradox, $x_i = x_{2i} \bmod p$ after $O(p^{1/2})$ trials on average, and $x_i = x_{2i} \bmod n$ after $O(n^{1/2})$ trials on average
- Hence we succeed after $O(p^{1/2})$ trials on average

Pollard's $p-1$ method

- A number $x = \prod p_i^{e_i}$ is B-powersmooth if $p_i^{e_i} < B$
- The method assumes $p-1$ is B-powersmooth
- Let s be the product of all $p_i^{e_i} < B$
- By assumption $(p-1)|s$, hence $g^s = 1 \bmod p$
- We deduce $\gcd(g^s - 1, n) = p$
- Only works if some factor p such that $p-1$ smooth !
- Compute gcd with square-and-multiply algorithm

Carl Pomerance

Quadratic Field Sieve : Rough version

- A congruence $x^2 = y^2 \bmod n$ such that $x \neq \pm y \bmod n$ implies that $\gcd(x-y,n)$ is a non trivial factor of n
- Set a smoothness bound $B \approx L_n(1/2)$
- Factor basis $\mathcal{F} = \{\text{primes smaller than } B\}$
- Pick random x_i until you find a relation

$$x_i^2 \bmod n = \prod_{s_j \in \mathcal{F}} s_j^{e_{ij}}$$

(probability is about $L_n(1/2)^{-1}$)
- Repeat until you have $|\mathcal{F}|$ relations

Quadratic Field Sieve : Rough version (2)

- For each i write the exponents e_{ij} in a row vector
- Perform linear algebra modulo 2 on these vectors to find a_i such that $\sum_{i=1}^{n} e_{ij} a_i = 2b_j$ even
- Deduce a congruence

$$\left(\prod_i x_i^{a_i}\right)^2 = \prod_i \left(x_i^2\right)^{a_i} = \prod_i \left(\prod_{s_j \in \mathcal{F}} s_j^{e_{ij}}\right)^{a_i} = \left(\prod_{s_j \in \mathcal{F}} s_j^{b_j}\right)^2$$

- Only 2 congruences needed on average

Improvements

- Choose x slightly bigger than $\sqrt{n}$ such that

$$x^2 \bmod n = x^2 - n = (\sqrt{n} + t)^2 - n = 2t\sqrt{n} + t^2$$

  is about the size of $\sqrt{n}$
- Sieving : instead of testing smoothness with trial divisions, build a basis of smooth numbers of the form $x^2 - n$ by extending the sieve of Erathostenes

Outline

(General) Number Field Sieve

- Original idea by Pollard, later developed by many authors
- Eventually led to discrete logarithm algorithms as well
- Let $d \approx (\log n)^{1/3}$ and $m \approx \lceil n^{1/d} \rceil$
- Write $n = \sum_{i=0}^{d} f_i m^i$
- Let $f(x) = \sum_{i=0}^{d} f_i x^i$
- We have a ring homomorphism

$$\varphi \ : \ \frac{\mathbb{Q}(x)}{(f(x))} \to \mathbb{Z}_n \ : \ x \to m$$

Factor basis and relations : rough idea

- Define smoothness bound $B \approx L_n(1/3)$
- Define factor basis $\mathcal{F} = \mathcal{F}_1 \cup \mathcal{F}_2$ where
  - $\mathcal{F}_1 =$ set of primes smaller than B
  - $\mathcal{F}_2 = \{a + bm \mid a, b \in \mathbb{Z}, N(a + bx) \in \mathcal{F}_1\}$
    (here $N(a + bx) = (b)^d f(a/b)$ is the number field norm)
- Generate pairs (a, b) with $a, b \approx L_n(1/3)$ until both $a + bm$ and $N(a + bx)$ are B-smooth
- Deduce a relation from each such pair
- Use linear algebra to get x, y such that $x^2 = y^2 \bmod n$
- Complexity $\approx L_n(1/3)$

Technicalities

- As such the number field side of equation may not be a square after linear algebra : only its norm is a square
- $\mathbb{Z}[x]$ may not be the full ring of integers
- Need to deal with units
- Need to deal with non-unique factorization / ideal class group when class number $h > 1$
- All issues solved by Adleman :
  - Fix a random set of $O(\log n)$ primes q_i
  - Consider multiplicative characters extending Legendre symbols $\chi_{q_i}(ax + b) = \left(\frac{am+b}{q_i}\right)$
  - Include $(\chi_{q_i}(ax + b))_i$ in each exponent relation

Remarks

- Instead of generating a, b randomly, fix random a values and sieve on b for each fixed a
- Initially various heuristics, but now rigorous bound for complexity of finding $x^2 = y^2 \bmod n$ (yet we cannot prove $x \neq \pm y \bmod n$!)
- Exact constant more efficient for Mersenne-like numbers (Special Number Field Sieve) than arbitrary numbers (General Number Field Sieve)
- Improved constant using several number fields in parallel (Coppersmith's trick)

Further readings

- Pomerance, *A Tale of Two Sieves*
- Buhler, Lenstra, Pomerance, *Factoring integers with the Number Field Sieve*

Outline

Pollard's $p - 1$ method

- A number $x = \prod p_i^{e_i}$ is B-powersmooth if $p_i^{e_i} < B$
- The method assumes $p - 1$ is B-powersmooth
- Let s be the product of all $p_i^{e_i} < B$
- By assumption $(p - 1)|s$, hence $g^s = 1 \bmod p$
- We deduce $\gcd(g^s - 1, n) = p$
- Only works if some factor p such that $p - 1$ smooth !

Elliptic curve factorization method

- Idea : generalize previous method when neither $p - 1$ nor $q - 1$ are smooth
- The group order $\#E(\mathbb{F}_p)$ of an elliptic curve can be smooth even when $p - 1$ is not !

Elliptic curve addition law

- Let $E : y^2 = x^3 + a_4 x + a_6$
- Let $P_1 = (x_1, y_1)$, $P_2 = (x_2, y_2)$ two points on the curve
- The chord-and-tangent rules lead to addition law formulae : for example we have $P_1 + P_2 = (x_3, y_3)$ where $\lambda = \frac{y_2 - y_1}{x_2 - x_1}$, $\nu = \frac{y_1 x_2 - y_2 x_1}{x_2 - x_1}$, $x_3 = \lambda^2 - x_1 - x_2$, $y_3 = -\lambda x_3 - \nu$
- These formulae involve divisions
- Over $\mathbb{F}_p$, a division by 0 means P_3 is point at infinity
- Over $\mathbb{Z}_n$, a division fails if $(x_2 - x_1)$ is not invertible
- A failure reveals a factor of n !

Elliptic curve factorization method

1. Choose E and $P = (x, y) \in E(\mathbb{Z}_n)$
2. Let B be a smoothness bound on $\#E(\mathbb{Z}_p)$ for $p|n$
3. Compute $s = \prod p_i^{e_i}$ where $p_i^{e_i} \leq B$
4. We have $[s]P = 0 =$ "point at infinity" modulo p but $[s]P \neq 0$ in $\mathbb{Z}_n$
5. Try to compute $[s](P)$: a division by p must occur and produce an error
6. When a division by some d fails, compute

$$\gcd(d, n) \neq 1$$

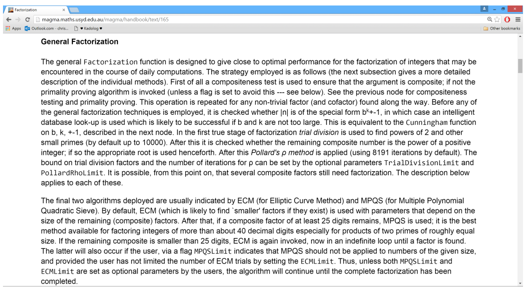
Elliptic curve factorization method

- For a random curve, we expect $\#E(\mathbb{F}_p)$ to be $\pm$ uniformly distributed in

$$\#E(\mathbb{F}_p) \in [(p+1) - 2\sqrt{p}, (p+1) + 2\sqrt{p}]$$

- Let $B \approx L_p(1/2)$ so that smoothness probability is about $(L_p(1/2))^{-1}$
- Repeat with random curves until you get a factor
- Remark : runtime depends on the smallest factor
- In practice, the method is used as subroutine to factor middle-size integers when $\log_2 n \approx 60 - 80$ bits

Factorization in practice : Magma



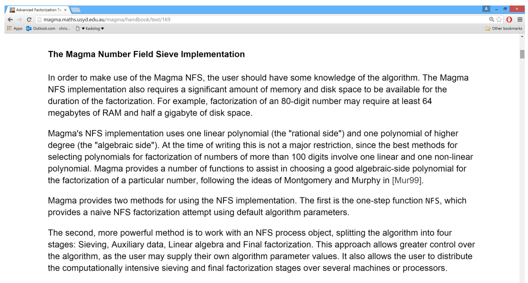
General Factorization

The general `Factorization` function is designed to give close to optimal performance for the factorization of integers that may be encountered in the course of daily computations. The strategy employed is as follows (the next subsection gives a more detailed description of the individual methods). First of all a compositeness test is used to ensure that the argument is composite; if not the primality proving algorithm is invoked (unless a flag is set to avoid this --- see below). See the previous node for compositeness testing and primality proving. This operation is repeated for any non-trivial factor (and cofactor) found along the way. Before any of the general factorization techniques is employed, it is checked whether |n| is of the special form b^k+-1, in which case an intelligent database look-up is used which is likely to be successful if b and k are not too large. This is equivalent to the `Cunningham` function on b, k, +-1, described in the next node. In the first true stage of factorization *trial division* is used to find powers of 2 and other small primes (by default up to 10000). After this it is checked whether the remaining composite number is the power of a positive integer; if so the appropriate root is used henceforth. After this *Pollard's ρ method* is applied (using 8191 iterations by default). The bound on trial division factors and the number of iterations for ρ can be set by the optional parameters `TrialDivisionLimit` and `PollardRhoLimit`. It is possible, from this point on, that several composite factors still need factorization. The description below applies to each of these.

The final two algorithms deployed are usually indicated by ECM (for Elliptic Curve Method) and MPQS (for Multiple Polynomial Quadratic Sieve). By default, ECM (which is likely to find `smaller' factors if they exist) is used with parameters that depend on the size of the remaining (composite) factors. After that, if a composite factor of at least 25 digits remains, MPQS is used; it is the best method available for factoring integers of more than about 40 decimal digits especially for products of two primes of roughly equal size. If the remaining composite is smaller than 25 digits, ECM is again invoked, now in an indefinite loop until a factor is found. The latter will also occur if the user, via a flag `MPQSLimit` indicates that MPQS should not be applied to numbers of the given size, and provided the user has not limited the number of ECM trials by setting the `ECMLimit`. Thus, unless both `MPQSLimit` and `ECMLimit` are set as optional parameters by the users, the algorithm will continue until the complete factorization has been completed.

- No Number Field Sieve involved by default

Factorization in practice : Magma



The Magma Number Field Sieve Implementation

In order to make use of the Magma NFS, the user should have some knowledge of the algorithm. The Magma NFS implementation also requires a significant amount of memory and disk space to be available for the duration of the factorization. For example, factorization of an 80-digit number may require at least 64 megabytes of RAM and half a gigabyte of disk space.

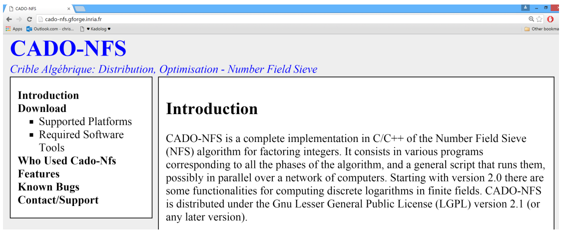
Magma's NFS implementation uses one linear polynomial (the "rational side") and one polynomial of higher degree (the "algebraic side"). At the time of writing this is not a major restriction, since the best methods for selecting polynomials for factorization of numbers of more than 100 digits involve one linear and one non-linear polynomial. Magma provides a number of functions to assist in choosing a good algebraic-side polynomial for the factorization of a particular number, following the ideas of Montgomery and Murphy in [Mur99].

Magma provides two methods for using the NFS implementation. The first is the one-step function `NFS`, which provides a naive NFS factorization attempt using default algorithm parameters.

The second, more powerful method is to work with an NFS process object, splitting the algorithm into four stages: Sieving, Auxiliary data, Linear algebra and Final factorization. This approach allows greater control over the algorithm, as the user may supply their own algorithm parameter values. It also allows the user to distribute the computationally intensive sieving and final factorization stages over several machines or processors.

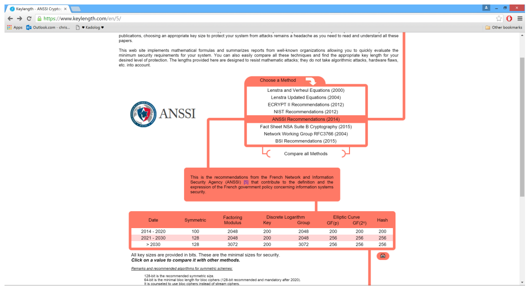
- May require expert knowledge to use properly

Factorization in practice : CADO-NFS



- Probably best available software today !

Recommended key lengths



- Check `www.keylength.com` for updates !

Outline

Generic DLP algorithms

Index Calculus for DLP : introduction

Subexponential DLP algorithms

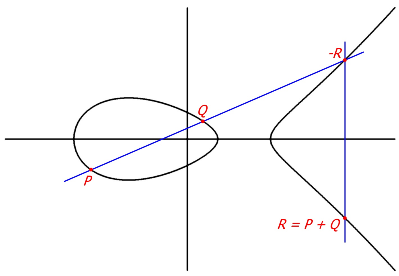
Quasi-polynomial DLP algorithm

Factoring algorithms

Elliptic Curve Discrete Logarithm Problem

Elliptic curves

- Set of rational points satisfying some cubic equation
- Group structure given by chord and tangent rule



Elliptic curve discrete logarithm problem (ECDLP)

- Given E over a finite field K,
  Given $P \in E(K)$, given $Q \in G := < P >$,
  Find $k \in \mathbb{Z}$ such that $Q = kP$.
- In practice K is often a prime field, a binary field with prime extension, or $\mathbb{F}_{p^n}$ with n relatively small
- Common belief : best algorithms are generic ones
  (at least for the parameters used in practice)
  160-bit ECDLP $\approx$ 2048-bit DLP or factoring

Reductions to simpler DLP

- Idea : transfer ECDLP to a "simpler" DLP problem through a group homorphism
- **MOV reduction** if $|G|$ divides $q^m - 1$ [MOV93]
  Use pairings to transfer ECDLP to DLP on K^m
- Polynomial time for **anomalous curves** [SA98,S98,S99]
  Transfer ECDLP to a p-adic elliptic logarithm if $|G| = |K|$
- **Weil descent** for some curves over $\mathbb{F}p^n$ [GS99,GHS00]
  Transfer ECDLP to the Jacobian of an hyperelliptic curve
- Only work for specific families

Remember : Index calculus

- General method to solve discrete logarithm problems
  1. Define a **factor basis** $\mathcal{F} \subset G$
  2. **Relation search** : find about $|\mathcal{F}|$ **relations**
  $$a_i P + b_i Q = \sum_{P_j \in \mathcal{F}} e_{ij} P_j$$
  3. Do **linear algebra** modulo $|G|$ on the relations to get
  $$aP + bQ = 0$$
- Define $\mathcal{F}$ s.t. there is an "efficient" algorithm for Step 2
- Balance relation search and linear algebra

Index calculus : success stories

- **Finite fields :** Adleman [A79,A94], Coppersmith [C84], Adleman and Huang [AH99], Joux [J13], Barbulescu-Gaudry-Joux-Thomé [BGJT13]
  Subexponential complexity for any field
  Quasipolynomial for small to medium characteristic fields
- **Hyperelliptic curves :**
  Adleman-DeMarrais-Huang [ADH94], Enge [E00], Gaudry [G00], Gaudry-Thomé-Thériault-Diem [GTTD07]
  Subexponential for large genus ; beats BSGS if $g \geq 3$
- **Elliptic curves :** no algorithm at all until 2005

Index calculus for elliptic curves

- For finite fields, **small "primes"** are a natural factor basis
  - Every element factors uniquely as a product of primes
  - "Good" probability that random elements are smooth
- Similarly for elliptic curves, we will need
  1. A definition of "small" elements
  2. An algorithm to decompose general elements into (potentially) small elements
- First partial solutions given by Semaev [S04]

Summation polynomials [S04]

- Relate the x-coordinates of points that sum to O
- $S_r(x_1, \ldots, x_r) = 0$
  $\Leftrightarrow \quad \exists (x_i, y_i) \in E(\bar{K})$ s.t. $(x_1, y_1) + \cdots + (x_r, y_r) = O$
- Recursive formulae :
  $S_2(x_1, x_2) = x_1 - x_2$
  $S_3(x_1, x_2, x_3) = \ldots$ (depends on E)
  $S_r(x_1, \ldots, x_r) =$
  $Res_X\left(S_{r-k}(x_1, \ldots, x_{m-k-1}, X), S_{k+2}(x_{r-k}, \ldots, x_r, X)\right)$
- S_r has degree 2^{r-2} in each variable
  Symmetric set of solutions

Semaev's variant of index calculus

- Semaev's variant of index calculus :
  - **Factor basis** :
    define $\mathcal{F}_V := \{(x, y) \in E | \mathbf{x} \in \mathbf{V}\}$ where $V \subset K$
  - **Relation search** : for each relation,
    Compute $(X_i, Y_i) := a_i P + b_i Q$ for random a_i, b_i
    Find $\mathbf{x_j} \in \mathbf{V}$ with $\mathbf{S_{m+1}(x_1, \ldots, x_m, X_i) = 0}$
    Find the corresponding y_j
- **Semaev's observation** : ECDLP reduced to solving summation's polynomial with constraints $x_i \in V$
- For $K = \mathbb{F}_p$, Semaev proposed $V := \{x < B\}$ but he could not solve summation polynomials

Focus on composite fields [G09,D11]

- For $K := \mathbb{F}_{q^n}$, Gaudry and Diem proposed $V := \mathbb{F}_q$
- Finding relations amounts to finding $\mathbf{x_j} \in \mathbb{F}_\mathbf{q}$ with $\mathbf{S_{n+1}(x_1, \ldots, x_n, X_i) = 0}$
- See $\mathbb{F}_{q^n}$ as a vector space over $\mathbb{F}_q$
- See polynomial equation $S_{n+1} = 0$ over $\mathbb{F}_{q^n}$ as a system of n polynomial equations in n variables over $\mathbb{F}_q$
- System can be solved with generic algorithms using complexity polynomial in Bézout bound $O(2^{n^2})$
- Gives $L(2/3)$ algorithm when $n \approx \sqrt{\log q} \approx (\log q^n)^{1/3}$

ECDLP : state-of-the-art

- We have an $L(2/3)$ algorithm to solve ECDLP over fields $\mathbb{F}_{q^n}$ if q and n have the right size
- In applications we are interested in ECDLP over either prime fields, or $\mathbb{F}_{2^n}$ with extension degree n prime
- Some algorithms have been suggested in those cases, but their complexity is unknown

Binary case [D11b,FPPR12]

Let $K := \mathbb{F}_{2^n}$. Fix $n' < n$ and $m \approx n/n'$

- **Factor basis** :
  Choose a **vector subspace** V of $\mathbb{F}_{2^n}$ with dimension n'
  Define $\mathcal{F}_V := \{(x,y) \in E | x \in V\}$
- **Relation search** : find about $2^{n'}$ relations. For each one,
  Compute $(X_i, Y_i) := a_i P + b_i Q$ for random a_i, b_i
  Find $x_j \in V$ with $S_{m+1}(x_1, \ldots, x_m, X_i) = 0$
  Find the corresponding y_j
- **Linear algebra** between the relations

Finding relations : Weil descent

- Finding relations amounts to
  Finding $\mathbf{x_i} \in \mathbf{V}$ with $\mathbf{S_{m+1}(x_1, \ldots, x_m, X) = 0}$
- Let $\{v_1, \ldots, v_{n'}\}$ be a basis of V
  Define $x_{ij} \in \mathbb{F}_2$ such that $x_i = \sum_{j=1}^{n'} x_{ij} v_j$

$$S_{m+1}\left(\sum_{j=1}^{n'} x_{1j} v_j, \ldots, \sum_{j=1}^{n'} x_{n'j} v_j, X\right) = 0$$

- See $\mathbb{F}_{2^n}$ as a vector space over $\mathbb{F}_2$
- The polynomial equation over $\mathbb{F}_{2^n}$ corresponds to a **system** of polynomial equations over $\mathbb{F}_2$

Complexity of characteristic 2 algorithm

- Computing S_{m+1} with resultants : cost 2^{t_1} where

$$t_1 \approx m(m+1)$$

- Finding $2^{n'}$ relations : total cost 2^{t_2} where

$$t_2 \approx n' + \log T_R$$

  where $T_R(m, n', n)$ is **time to compute one relation**
- (Sparse) linear algebra on relations : cost $2^{\omega' t_3}$ where

$$t_3 \approx \log m + \log n + \omega' n'$$

Complexity of characteristic 2 algorithm

- Conjectured to be subexponential based on a heuristic assumption on Groebner Basis algorithms behavior and experimental results [PQ12]
- Original assumption perhaps too optimistic
- Still an open problem

ECDLP over Prime Fields

- No vector space available to define the factor basis
- Find a rational map $L = \circ_{j=1}^{n'} L_j$ with a large zero set
- Define a factor basis $\mathcal{F} = \{(x,y) \in E(K) | L(x) = 0\}$
- Each relation search now amounts to solving

$$\begin{cases} S_{m+1}(x_{11}, \ldots, x_{m1}, X) = 0 & \\ x_{i,j+1} = L_j(x_{i,j}) & i = 1, \ldots, m; j = 1, \ldots, n' - 1 \\ 0 = L_{n'}(x_{i,n'}) & i = 1, \ldots, m. \end{cases}$$

- Complexity is an open problem

Outline

Conclusion on (EC)DLP and factoring

- Very active field of research, with recent breakthroughs
- Research challenges
  - Find new algorithms for these problems
  - Analyze existing algorithms
  - Consider related problems
- Come to me if interested in a project in the area
- Recommended key sizes : `www.keylength.com`