Factorisation and Discrete-Logarithm Algorithms

Federico Pintore [1]

[1] Mathematical Institute,
University of Oxford

Outline

1. Factorization algorithms
2. Generic discrete logarithm algorithms
3. Discrete logarithms over finite fields

Outline

Integer factorization

Problem: Given a composite number N, which is the product of two n-bit primes, compute one of its factors.

Trial Divison: try every prime number up to $\sqrt{N}$. Running time is, at worst, $O(\sqrt{N} \cdot \text{polylog}(\text{N}))$.

Can we do better?

Pollard's rho

- It can be used to factor any arbitrary integer $N = pq$.
- Idea: find a **good** pair (x, y) s.t. $x = y \pmod{p}$ but $x \neq y \pmod{N}$.
- This implies that $\gcd(x - y, N) = p$, and therefore a non-trivial factor of N is obtained.
- Define some "pseudorandom" iteration function f (a standard choice would be $f(x) = x^2 + 1 \mod N$. It has the property that, if $x = x' \pmod{p}$, then $f(x) = f(x') \pmod{p}$.)
- At step i-th, compute x_i, x_{2i} and $\gcd(x_i - x_{2i}, N)$.
- By birthday's bound, a pair (x_i, x_{2i}) s.t. $x_i = x_{2i} \pmod{p}$ is expected to be found after $O(p^{1/2})$ trials on average.

Pollard's Rho

```
Input: integer N (a product of two n-bit primes)
a := b ← Z*_N
for i ∈ {2, ..., 2^{n/2}} do
    a := f(a)
    b := f(f(b))
    p := gcd(a − b, N)
    if p ∉ {1, N} then
        return p.
    end if
end for
```

Pollard's $p-1$ and Elliptic curve factorization methods

- Pollard's $p-1$ is an effective method if $p-1$ has only "small" prime factors.
- Elliptic-curve factorisation method generalises it when neither $p-1$ nor $q-1$ are smooth.
- The group order $\#E(\mathbb{F}_p)$ of an elliptic curve E can be smooth even when $p-1$ is not!
- Choosing *strong primes* for RSA, i.e. $p-1$ and $q-1$ both have large prime factors, can help against Pollard's $p-1$, but not against Elliptic-curve factorisation method or Number Field Sieve.

Quadratic Sieve Algorithm

- It runs in sub-exponential time in the length of N. Good choice for numbers up to about 300 bits long.
- Try to factor 8051. $8051 = 90^2 - 7^2 = (90-7)(90+7) = 83 \times 97$.
- **Idea**: find a, b s.t. $a^2 = b^2 \pmod N$ but $a \neq \pm b \pmod N$. Hence $\gcd(a-b, N)$ gives one non trivial factor of N.

Quadratic Sieve Algorithm

- Fix some bound $B \in \mathbb{N}$, and let $\mathcal{F} = \{p_1, \ldots, p_k\}$ the set of primes less than or equal to B.
- Search for integers x_i, where $x_1 = \left\lceil \sqrt{N} \right\rceil, x_2 = \left\lceil \sqrt{N} \right\rceil + 1, \ldots$, s.t. $q_i := x_i^2 \pmod N$ is B-smooth, and factor them.
- Find a subset S of $\{q_i\}_i$ such that the product of its elements is a square, i.e.

$$\prod_{j \in S} q_j = \prod_{\ell=1}^{k} p_\ell^{\sum_{j \in S} e_{j,\ell}} \quad s.t. \quad \sum_{j \in S} e_{j,\ell} = 0 \pmod 2 \quad \forall \ell \in \{1, \ldots, k\}$$

- S can be found using linear algebra.

Quadratic Sieve Algorithm

- Define the matrix of exponents (modulo 2) as follows:

$$\begin{pmatrix} e_{1,1} \pmod 2 & e_{1,2} \pmod 2 & \dots & e_{1,k} \pmod 2 \\ \vdots & \vdots & \ddots & \vdots \\ e_{m,1} \pmod 2 & e_{m,2} \pmod 2 & \dots & e_{m,k} \pmod 2 \end{pmatrix}$$

- If $m = k + 1$, then there exists a nonempty subset S of rows that sum to the zero vector modulo 2.

Quadratic Sieve Algorithm

- Take $N = 377753$. We can compute the following:

$$620^2 \bmod N = 17^2 \cdot 23$$
$$621^2 \bmod N = 2^4 \cdot 17 \cdot 29$$
$$645^2 \bmod N = 2^7 \cdot 13 \cdot 23$$
$$655^2 \bmod N = 2^3 \cdot 13 \cdot 17 \cdot 29$$

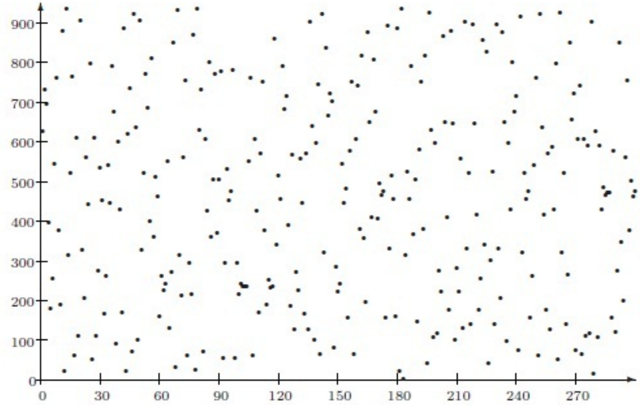
$$(620 \cdot 621 \cdot 645 \cdot 655 \pmod N)^2 = (2^7 \cdot 13 \cdot 17^2 \cdot 23 \cdot 29)^2 \pmod N$$
$$\Rightarrow 127194^2 = 45335^2 \pmod N$$

Since $127194 \neq \pm 45335 \pmod N$,
$\gcd(127194 - 45335, 377753) = 751$ gives a non trivial factor of N.

Outline

Why Discrete Logarithm?

A graph of $f(x) = 627^x \pmod{941}$ for $x = 1, 2, 3, \ldots$



Discrete logarithms

- Trivial if $(G, \circ) = (\mathbb{F}_p, +)$. Why?
- Recently broken if $(G, \circ) = (\mathbb{F}_{2^n}^*, *)$
  (more generally if characteristic is small)
- Believed to be hard for $G = \mathbb{F}_p^*$
  and harder for (well-chosen) elliptic curve groups

Discrete logarithms

- Trivial if $(G, \circ) = (\mathbb{F}_p, +)$. Why?
- Recently broken if $(G, \circ) = (\mathbb{F}_{2^n}^*, *)$
  (more generally if characteristic is small)
- Believed to be hard for $G = \mathbb{F}_p^*$
  and harder for (well-chosen) elliptic curve groups

Discrete logarithms

- Trivial if $(G, \circ) = (\mathbb{F}_p, +)$. Why?
- Recently broken if $(G, \circ) = (\mathbb{F}_{2^n}^*, *)$
  (more generally if characteristic is small)
- Believed to be hard for $G = \mathbb{F}_p^*$
  and harder for (well-chosen) elliptic curve groups

Generic group model

- Algorithms do not exploit any special properties of the encodings of the group elements, other than the fact that each group element is encoded as a unique binary string.
- For instance, the attacker just receives bitstrings instead of $\mathbb{Z}_n$ elements (n itself is often hidden but the size of n cannot be hidden).
- Operations on group elements are performed using an oracle that provides access to the group operations.
- 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 ← 1; h' ← g
2: if h' = h then
3:     return k
4: else
5:     k ← k + 1; h' ← h'g
6:     Go to Step 2
7: end if
```

- Generic algorithm
- Time complexity $|G|$ in the worst case
- Can we do better?

Pohlig-Hellman

- They observed that Dlog in a group $\mathbb{G}$ is as hard as the Dlog in the largest subgroup of prime order in $\mathbb{G}$.
- This applies in any arbitrary finite abelian group.
- Assume $|\mathbb{G}| = N = n_1 n_2$ 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.
- Assuming that we can solve DLP in that subgroup, this would give us $k \bmod n_2$.
- Repeating the same thing for each factor of N and using CRT would give us k.

Pohlig-Hellman

- Let $\mathbb{G} = \langle g \rangle$ of order $N = \#\mathbb{G} = \prod_{i=1}^{\ell} p_i^{e_i}$
- Given $h = g^x$, we want to first find $x \mod p_i^{e_i}$ and then use CRT to recover it mod N.
- There is a group isomorphism $\phi : \mathbb{G} \to C_{p_1^{e_1}} \times \cdots \times C_{p_\ell^{e_\ell}}$.
- Define the projection map $\phi_{p_i} : \mathbb{G} \to C_{p_i^{e_i}}$ where $\phi_{p_i}(g) = g^{N/p_i^{e_i}}$. ϕ_{p_i} is a group homomorphism, i.e., if $h = g^x$ in $\mathbb{G}$, then $\phi_{p_i}(h) = \phi_{p_i}(g)^x$ in $C_{p_i^{e_i}}$.
- Solving the discrete logarithm in $C_{p_i^{e_i}}$ reduces to solving e_i discrete logarithm in the group C_{p_i} following an inductive procedure.
- Given $h' = g^{x'} \in C_{p_i^{e_i}}$, we write $x' = x_0 + x_1 p_i + \cdots + x_{e_i - 1} p_i^{e_i - 1}$ and then find $x_0, x_1, \ldots, x_{e_i - 1}$ in turn.

Baby step, giant step (BSGS)

- Given a public cyclic group $\mathbb{G} = \langle g \rangle$, now we can assume that $\mathbb{G}$ has a prime order p.
- Given $h \in \mathbb{G}$, find the value of k s.t. $h = g^k$.
- Let $N' = \lceil \sqrt{|\mathbb{G}|} \rceil$
- 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 each $\mathcal{O}\left(|\mathbb{G}|^{1/2}\right)$
- Can we do better in terms of space requirement and still obtain a time complexity of $\mathcal{O}\left(\sqrt{|\mathbb{G}|}\right)$

Baby step, giant step (BSGS)

- Given a public cyclic group $\mathbb{G} = \langle g \rangle$, now we can assume that $\mathbb{G}$ has a prime order p.
- Given $h \in \mathbb{G}$, find the value of k s.t. $h = g^k$.
- Let $N' = \lceil \sqrt{|\mathbb{G}|} \rceil$
- 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 each $\mathcal{O}\left(|\mathbb{G}|^{1/2}\right)$
- Can we do better in terms of space requirement and still obtain a time complexity of $\mathcal{O}\left(\sqrt{|\mathbb{G}|}\right)$

Baby step, giant step (BSGS)

- Given a public cyclic group $\mathbb{G} = \langle g \rangle$, now we can assume that $\mathbb{G}$ has a prime order p.
- Given $h \in \mathbb{G}$, find the value of k s.t. $h = g^k$.
- Let $N' = \lceil \sqrt{|\mathbb{G}|} \rceil$
- 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 each $\mathcal{O}\left(|\mathbb{G}|^{1/2}\right)$
- Can we do better in terms of space requirement and still obtain a time complexity of $\mathcal{O}\left(\sqrt{|\mathbb{G}|}\right)$

Baby step, giant step (BSGS)

- Given a public cyclic group $\mathbb{G} = \langle g \rangle$, now we can assume that $\mathbb{G}$ has a prime order p.
- Given $h \in \mathbb{G}$, find the value of k s.t. $h = g^k$.
- Let $N' = \lceil \sqrt{|\mathbb{G}|} \rceil$
- 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 each $\mathcal{O}\left(|\mathbb{G}|^{1/2}\right)$
- Can we do better in terms of space requirement and still obtain a time complexity of $\mathcal{O}\left(\sqrt{|\mathbb{G}|}\right)$

Baby step, giant step (BSGS)

- Given a public cyclic group $\mathbb{G} = \langle g \rangle$, now we can assume that $\mathbb{G}$ has a prime order p.
- Given $h \in \mathbb{G}$, find the value of k s.t. $h = g^k$.
- Let $N' = \lceil \sqrt{|\mathbb{G}|} \rceil$
- 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 each $\mathcal{O}\left(|\mathbb{G}|^{1/2}\right)$
- Can we do better in terms of space requirement and still obtain a time complexity of $\mathcal{O}\left(\sqrt{|\mathbb{G}|}\right)$

Baby step, giant step (BSGS)

- Given a public cyclic group $\mathbb{G} = \langle g \rangle$, now we can assume that $\mathbb{G}$ has a prime order p.
- Given $h \in \mathbb{G}$, find the value of k s.t. $h = g^k$.
- Let $N' = \lceil \sqrt{|\mathbb{G}|} \rceil$
- 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 each $\mathcal{O}\left(|\mathbb{G}|^{1/2}\right)$
- Can we do better in terms of space requirement and still obtain a time complexity of $\mathcal{O}\left(\sqrt{|\mathbb{G}|}\right)$

Pollard's Algorithms

- John Pollard, a famous name in factoring/Dlog algorithms in the 20th century.
- Known for $(P-1)$ method, Rho-method, Number Field Sieve.
- The idea in the Rho method is to find a collision in a random mapping.
- Using the birthday paradox *naively* is no better than Baby-Step/Giant-Step method in terms of space/time requirements.
- Similar to the improved birthday paradox attack on hash functions, we can use Floyd's cycle finding algorithm, i.e. given (x_i, x_{2i}), we compute

$$(x_{i+1}, x_{2i+2}) = (f(x_i), f(f(x_{2i})))$$

- We stop when $x_\ell = x_{2\ell}$

Pollard's rho

- Define the sets G_1, G_2, G_3 of about the same size such that $G = G_1 \cup G_2 \cup G_3$ and $G_i \cap G_j = \{\}$, assuming that $1 \notin G_2$.
- Over $\mathbb{Z}_p^*$, one 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 random walk $f : G \to G$ such that

$$x_{i+1} = f(x_i) = \begin{cases} hx_i & x_i \in G_1 \\ x_i^2 & x_i \in G_2 \\ gx_i & x_i \in G_3 \end{cases}$$

Pollard's rho

- Given $g, h = g^x$, we start from $x_0 := 1$ and apply f recursively to get $\{x_i, x_{2i}\}_i$.
- By the way f is defined, we can keep track of (x_t, a_t, b_t) such that $x_t = g^{a_t} h^{b_t}$, where

$$a_{i+1} = \begin{cases} a_i \\ 2a_i \mod p \\ a_i + 1 \mod p \end{cases}, b_{i+1} = \begin{cases} b_i + 1 \mod p & x_i \in G_1 \\ 2b_i \mod p & x_i \in G_2 \\ b_i & x_i \in G_3 \end{cases}$$

- We stop when a collision is found, i.e. $x_\ell = x_{2\ell}$, therefore $x = \dfrac{a_{2\ell} - a_\ell}{b_\ell - b_{2\ell}} \mod p$.
- If f is "random enough", then we should find the Dlog in expected time $\mathcal{O}\left(\sqrt{|G|}\right)$.

Pollard's rho

$N \leftarrow \lceil \sqrt{|G|} \rceil$
$a_1 = 0; b_1 = 0; x_1 = 1$
$(x_2, a_2, b_2) = f(x_1, a_1, b_1)$
for $k \in \{2, \ldots, N\}$ **do**
  $(x_1, a_1, b_1) = f(x_1, a_1, b_1)$
  $(x_2, a_2, b_2) = f(f(x_2, a_2, b_2))$
  if $x_1 = x_2$ **break;**
end for
if $b_1 = b_2 \mod p$ **then**
  return $\perp$
else
  return$(a_2 - a_1)/(b_1 - b_2) \mod p$
end if

Pollard's Rho: example

Example (Smart's book)

Consider $\mathbb{G} = \langle g \rangle$, a subgroup of $\mathbb{F}_{607}^*$ of order $p = 101$, with $g = 64$. Given $h = 122 = 64^x$. Solve for x.
We split $\mathbb{G}$ into three sets S_1, S_2, S_3 as follows:

$$S_1 = \{x \in \mathbb{F}_{607}^* : x \leq 201\}$$

$$S_2 = \{x \in \mathbb{F}_{607}^* : 202 \leq x \leq 403\}$$

$$S_3 = \{x \in \mathbb{F}_{607}^* : 404 \leq x \leq 606\}$$

Pollard's Rho: example

Example

A collision is found when $i = 14$, this implies that $g^0 h^{12} = g^{64} h^6$, so $[12x = 64 + 6x \mod 101]$ and therefore $x = 78$.

More from Pollard

- Pollard's Lambda Method: similar to the Rho method in that it uses deterministic random walk, but it is particularly designed to the cases where we know that the Dlog lies in a particular interval.
- Parallel Pollard's Rho: designed to be able to use computing resources of different sites across the internet.

Outline

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

(simplified) Index Calculus for $\mathbb{F}_p^*$

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

$$\mathcal{F}_B := \{\text{primes } p_i \leq B\} = \{p_1, \ldots, p_k\}$$

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

$$g^{a_i} = \prod_{p_j \in \mathcal{F}} p_j^{e_{i,j}} \qquad (1)$$

- **Linear algebra** Once we have $\ell \geq k$ linearly independent equations similar to equations (1), we solve $\bmod (p-1)$ for $\log_g p_i$, $i = 1, \ldots, k$.
- Search for t such that $[g^t \cdot h \mod p]$ is B-smooth. Once found, solve for $\log_g h \mod (p-1)$.

Size of the factor basis

- By the prime number theorem,

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

- Fact: 30% of all numbers have no prime factors above their square root. Surprisingly, a large proportion of numbers can be built out of so few primes!

Complexity Analysis

- How to choose an optimal B: If B is large, then it is more likely that the generated elements are B-smooth, but then testing that they are B-smooth is more difficult now. Therefore, we need to balance the cost!
- In order to choose an optimal B, we also need to know the probability that a random integer that is smaller than N is B-smooth.
- We will assume that the cost of generating relations dominates the overall complexity of Algorithm, i.e. assume that the linear algebra is negligible in terms of time complexity.
- We will simply use the trial-division to factor over $\mathcal{F}_B$.

Complexity Analysis

- A number is B-smooth if all its prime factors are smaller than B.
- Define $\Psi(N, B) = \#\{B\text{-smooth numbers} \leq N\}$.
- The probability that a positive integer $m \leq N$ is B-smooth is approximately equal to $\frac{1}{N} \cdot \Psi(N, B)$.
- The Canfield-Erdos-Pomerance Theorem: Let $u = \frac{\log N}{\log B}$, we have $\frac{1}{N} \cdot \Psi(N, B) = u^{-u+o(u)}$. This is the *Dickman-de Bruijn* function ρ, i.e. $\rho(u) \approx u^{-u}$.
- The expected number of random trials of choosing numbers in $[1; N]$ to find one that is B-smooth is $\approx u^u$

Complexity Analysis

- Let $|\mathcal{F}_B| = k$, the expected running time of the algorithm is

$$\approx \underbrace{(k+1)}_{\text{nb of relations}} \cdot \underbrace{u^u}_{\text{expected nb of trials}} \cdot \underbrace{k}_{\text{nb of trial divisions}} \cdot \underbrace{M(\log N)}_{\text{time for a trial division}} \quad (2)$$

$$\approx B^2 \cdot u^u \quad \textit{drop the logarithmic factors, where } k \approx \frac{B}{\log B} \quad (3)$$

$$= N^{2/u} \cdot u^u \quad (4)$$

- We want to minimize $f(u) = N^{2/u} \cdot u^u$. If we set $f'(u) = 0$, we need a u s.t. $u^2 \log u \approx 2 \log N$.
- Let $u = 2\sqrt{\frac{\log N}{\log \log N}}$, we then get $u^2 \log u = 2 \log N + o(\log N)$

Complexity Analysis

- Back to our bound B:

$$
\begin{aligned}
B &= N^{1/u} \\
&= exp(\frac{1}{u}\log N) \\
&= exp(\frac{1}{2}\sqrt{\log N \log\log N)} \\
&= L_N(1/2, 1/2)
\end{aligned}
$$

- Note that $u^u = L_N(1/2, 1)$, therefore $B^2 u^u = L_N(1/2, 2)$.
- The cost of the linear algebra step is bounded by $\tilde{O}(B^3)$, i.e. $L_N(1/2, 3/2)$.

Further Reading (1)

- Andrew Granville.
  Smooth numbers: computational number theory and beyond.
  Algorithmic number theory: lattices, number fields, curves and cryptography, 44:267–323, 2008.

- Antoine Joux, Andrew Odlyzko, and Cécile Pierrot.
  The past, evolving present, and future of the discrete logarithm.
  In *Open Problems in Mathematics and Computational Science*, pages 5–36. Springer, 2014.

- Hendrik W Lenstra Jr.
  Factoring integers with elliptic curves.
  Annals of mathematics, pages 649–673, 1987.

Further Reading (2)

Carl Pomerance.
Smooth numbers and the quadratic sieve.
Algorithmic Number Theory, Cambridge, MSRI publication, 44:69–82, 2008.

Carl Pomerance.
A tale of two sieves.
Biscuits of Number Theory, 85, 2008.

Victor Shoup.
Lower bounds for discrete logarithms and related problems.
In *Advances in Cryptology—EUROCRYPT'97*, pages 256–266. Springer, 1997.

Further Reading (3)

Andrew Granville.
Smooth numbers: computational number theory and beyond.
Algorithmic number theory: lattices, number fields, curves and cryptography, 44:267–323, 2008.

Antoine Joux, Andrew Odlyzko, and Cécile Pierrot.
The past, evolving present, and future of the discrete logarithm.
In *Open Problems in Mathematics and Computational Science*, pages 5–36. Springer, 2014.

Hendrik W Lenstra Jr.
Factoring integers with elliptic curves.
Annals of mathematics, pages 649–673, 1987.

Further Reading (4)

- Carl Pomerance.
  Smooth numbers and the quadratic sieve.
  Algorithmic Number Theory, Cambridge, MSRI publication, 44:69–82, 2008.

- Carl Pomerance.
  A tale of two sieves.
  Biscuits of Number Theory, 85, 2008.

- Victor Shoup.
  Lower bounds for discrete logarithms and related problems.
  In *Advances in Cryptology—EUROCRYPT'97*, pages 256–266. Springer, 1997.