HARD PROBLEMS IN NUMBER THEORY ARISING FROM CRYPTOGRAPHY

KRISTIN LAUTER

1. Introduction

RSA is an algorithm for public-key cryptography, based loosely on the hardness of factoring. Suppose you have a public number $N = pq$, with p, q primes that are secret and roughly the same size. Another piece of information is a public exponent $e : 2^{16} - 1$. The idea is as follows: if you can factor N, then you can find a modular inverse for e in the multiplicative group $\mathbb{Z}/N\mathbb{Z}$. So to encrypt a message m, you can take $m^e \pmod N$ as the encryption, and if you know $N = pq$, then you can compute $\phi(N) = (p-1)(q-1)$ and an integer d such that $ed \equiv 1 \pmod{\varphi(N)}, (m^e)^d \equiv m \pmod N$. So this shows you that given encryption of this form, if you can factor, then you can decrypt.

Suppose you want to factor N: you could try to write it as

$$N = x^2 - y^2 = (x-y)(x+y),$$

and this would be very convenient. Unfortunately, this is pretty hard to do and pretty rare.

For example,

$$629 = 729 - 100 = (27-10)(27+10).$$

This is an extremely lucky example, as 27 is one of the first squares you'd likely try.

The **quadratic sieve method** was introduced by Carl Pomerance in 1981. The idea is to find squares of integers mod N: can try to generate a list of quadratic relations by taking a list of squares, looking at them modulo N, and putting them together, until you end up with a square on one side.

Example: $N = 1649$.

$$41^2 \equiv 32 \pmod{1649}$$
$$43^2 \equiv 200 \pmod{1649}$$

So $41^2 23^2 \equiv 80^2 \pmod{1649}$. Take GCD with 1649: $(41 \cdot 43 - 80, 1649) = 17$ and $(41 \cdot 43 + 80, 1649) = 97$.

The quadratic sieve has a heuristic running time that is sub-exponential:

$$L_N(1/2, c) = e^{(c+o(1))(\log N)^{\frac{1}{2}}(\log\log N)^{\frac{1}{2}}}.$$

In practice, today, $N \sim 2^{2048}$.

Number field sieve: has running time $L(\frac{1}{3}, c_1)$, heuristically better than the quadratic sieve. This was used for the record factorization of RSA 768.

Another factoring method is the Elliptic Curve Method (ECM), which can be useful in finding small factors. The runtime is proportional to the size of the factors it finds. (This is why we pick p, q to have roughly the same size.) ECM was used to find a 73-digit prime factor of $2^{1181} - 1$, using 200 Sony Playstations.

Discrete Logarithm Problem: Suppose we have a cyclic group $G = \langle g \rangle$. Key exchange between Alice and Bob:

$$\text{Alice } (a) \xrightarrow{g^a} \text{Bob } (b)$$

$$\xleftarrow[1]{g^b}$$

Computational Diffie-Hellman: given $g^a, g^b \to g^{ab}$. Decision version of DH: $(t, s, u) \to 0$. In some groups, DDH is easy (e.g., elliptic curves because of Weil pairing). These assumptions are used for building various protocols in cryptography.

Discrete Log Problem (DLP): let $y = g^a$. Given y, g, find a.

One of the first groups proposed: $G = (\mathbb{Z}/p\mathbb{Z})^* = \langle g \rangle$. What you need is for your group to have roughly prime order.

Index calculus attack: $p = 71$, take $B = 11$, and we have

$$\begin{aligned} 2^6 &\equiv -7 \pmod{71} \\ 7^2 &\equiv -2 \cdot 11 \pmod{71} \\ 11^2 &\equiv 5^2 \cdot 2 \pmod{71} \\ 2^2 &\equiv 5^2 \cdot 3 \pmod{71} \\ 3^4 &\equiv 2 \cdot 5 \pmod{71}, \end{aligned}$$

where we've stopped because we've reached 5 relations with 5 primes. This gives a linear system that one can solve for the exponents.

Something misleading about this: we indiscriminately too small primes, small powers, and factored. But you might not want to factor each time, and there are better ways to go about doing this.

One of the ways to do the index calculus: minimum parameter sizes, $p \sim 2^{2048}$. The record discrete log computation: 700-bit. This is the minimum you'd take in practice if you wanted to implement one of these systems.

2. Elliptic curve cryptography

Key: advantage we have with ECC is we don't have subexponential things to attack them. The best known attack for elliptic curve are square-root attacks (they're exponential attacks that run in square root of group order). So in practice, can take elliptic curve $E/\mathbb{F}_q$. We write down a short Weierstrass form of the curve $y^2 = x^3 + ax + b, a, b \in \mathbb{F}_q$, characteristic of $\mathbb{F}_q$ not 2,3. Can take group $G = E(\mathbb{F}_q)$ to be the group in the Discrete Log Problem. We want $\#E(\mathbb{F}_q) = N$, where N is a prime or something close to a prime (e.g., or a product of 2 primes).

So how do we generate elliptic curves with a certain number of primes?

Schoof-Elkies-Algorithm for point-counting: can randomly take elliptic curve over finite field, use SEA to count points.

Is there something better? **CM** (complex multiplication) **method** for generating curves with a given number of points.

How does this work? If $\#E(\mathbb{F}_q) = N$, by Hasse's theorem, we know

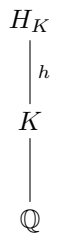
$$N = q + 1 - t,$$

where t is the trace of Frobenius. Frobenius is an operator on the points of an elliptic curve. Let π denote the Frobenius endomorphism. Its characteristic polynomial is

$$x^2 - tx + q,$$

where $\pi + \overline{\pi} = t, \pi\overline{\pi} = q$. So if $d = t^2 - 4q$, $\pi \in \mathbb{Q}(\sqrt{d}) = K$. So now we're looking for a curve E that has CM by K.

Explicit class field theory:



The Hilbert class field H_K is generated by the j-invariant of E, $j(E)$, where E has CM by K. $j(E)$ has minimal polynomial

$$H(X) = \prod_{E_i \text{ CM by } \mathcal{O}_K} (X - j(E_i)) \in \mathbb{Z}[X].$$

If we could compute this polynomial, then we could find the curves we wanted.

So suppose q is totally split over H_K. Then to compute the polynomial, reduce mod primes above q and piece together the information. The difficulty here is that $h, j(E_i)$ are large. As the discriminant of K (which is d) grows, the degree h goes to infinity, and the size of $j(E_i)$ grows out of control. The complexity is $e^{\sqrt{d}\log d \text{ or } (\log d)^2}$ depending on whether or not you assume GRH. We have that $h \sim \sqrt{d}$ and $d \sim q \sim 2^{256}$. So now we're looking at numbers that are $e^{2^{128}}$: these numbers are very big!

A couple things we can do to ameliorate this situation:

- Agashe-Lauter-Venkatesan (2001): CRT method for computing $H(X)$. Idea here is that instead of doing high-precision evaluation of modular forms (to recognize $j(E)$), try to compute the H's modulo lots of small primes. Find $H(X) \pmod{\ell}$ for ℓ small. Do this for enough ℓ: explicit CRT. $H(x) \pmod{q}$ (Dan Bernstein's thesis). Only need to compute one coefficient at a time, so storage is $\sqrt{d}$ bits instead of d bits.
- Sutherland (based on BBEL): $h \sim 10^{15}$, only possible because of CRT method.

A few words on **pairing-based cryptography**: Boneh-Franklin in 2001 solved a longstanding problem in identity-based encryption. The hardest of the "hard problems" is pairing inversion. Take, for example, the Weil pairing. Let $E/\mathbb{F}_q$ be an elliptic curve, m an integer. Look at $E[m] \ni P, Q$ and let k be the embedding degree. The Weil pairing is

$$e_m(P, Q) = f_Q(P)/f_P(Q) = u \in \mathbb{F}_{q^k}^*.$$

Problem 2.1 (Pairing inversion)**.** Given u, m, P, find Q.