Factorising n by φ(n)

Mathematical Cryptography,
Lecture 21

Contents

Factorising n by φ(n)

Some choices of witnesses to compositeness of n in the Miller-Rabin test allowe to factorise n:

- Choice $a \in \mathbb{Z}_n^+ \setminus \mathbb{Z}_n^*$ gives the factor $d = \gcd(a, n) > 1$.
- Choice $a \in K_n \setminus L_n$ generates a non-trivial square root of 1 (i.e., $c \neq \pm 1$, where $c^2 = 1$ in $\mathbb{Z}_n$), which gives the factors $d_{1,2} = \gcd(c \pm 1, n) > 1$.

We show that the knowledge of the factorisation of n is equivalent to the knowledge of $\varphi(n)$ and we use the same technique as in the proof of estimating number of false witnesses in the Miller-Rabin test.

Factorising n by φ(n)

Proposition

The problem of factorising n is equivalent to the knowledge of $\varphi(n)$, or from knowledge of one of these facts, the other one can be calculated in polynomial time.

- From $n = \prod_{i=1}^{r} p_i^{e_i}$ we have $\varphi(n) = \prod_{i=1}^{r} p_i^{e_i - 1}(p_i - 1)$.
- For $n = pq$, from $\varphi(n)$ we can compute p and q as solutions of the quadratic equation $x^2 - (n + 1 - \varphi(n))x + n = 0$. Since $\varphi(n) = (p-1)(q-1) = n - (p+q) + 1$, we know sum and product of two solutions.
- We design a polynomial algorithm that calculates factorisation of any n from knowledge of $\varphi(n)$ (or another multiple of the exponent of the group $\mathbb{Z}_n^*$).

Factorising n by φ(n)

Exponent of the group $\mathbb{Z}_n^*$

The exponent of the group $\mathbb{Z}_n^*$ is the smallest $m > 0$ such that $a^m = 1$ for all $a \in \mathbb{Z}_n^*$. It is denoted by $\lambda(n)$ (the Carmichael function) and the following formulas hold:

- $\lambda(\prod_{i=1}^r p_i^{e_i}) = \mathrm{lcm}(\lambda(p_1^{e_1}), \ldots, \lambda(p_r^{e_r}))$
- $\lambda(p^e) = \varphi(p^e) = p^{e-1}(p-1)$ for primes $p > 2$
- $\lambda(2^e) = \frac{\varphi(2^e)}{2} = 2^{e-2}$ for $e \geq 3$, $\lambda(4) = 2$, $\lambda(2) = 1$.

Consequence

- $\lambda(n) \mid \varphi(n)$ for every n, thus $\varphi(n)$ is a multiple of the exponent of the group $\mathbb{Z}_n^*$.
- $\lambda(n)$ is even for every $n > 2$.
- If $d \mid n$, then $\lambda(d) \mid \lambda(n)$.

Factorising n by φ(n)

Algorithm for finding a factor of n by λ(n)

Input: $n > 1$ odd, where $n \neq p^e$ for a prime p,
m such that $\lambda(n) \mid m$, $m = t\,2^h$ for t odd;
Output: d, where $d \mid n$, $1 < d < n$, or a message "failure"

- $a \stackrel{\not c}{\leftarrow} \mathbb{Z}_n^+$
- $d \leftarrow gcd(a, n)$
- if $d > 1$ then output d and halt endif
- $b \leftarrow a^t$ in $\mathbb{Z}_n$ (now $a \in \mathbb{Z}_n^*$, so $a^m = 1$ in $\mathbb{Z}_n$)
- for $j \leftarrow 0$ to $h - 1$ do
  - $d \leftarrow \gcd(b - 1, n)$
  - it $1 < d < n$ then output d and halt endif
  - $b \leftarrow b^2$ in $\mathbb{Z}_n$ enddo
- output "failure"

Factorising n by $\varphi(n)$

Proposition

The probability that the algorithm finds a factor of n is at least $\frac{1}{2}$.

Choosing $a \in \mathbb{Z}_n^+ \setminus \mathbb{Z}_n^*$ leads to factorization in part 1, choosing $a \in \mathbb{Z}_n^*$ leads to some square root of 1 in the part 2.
The algorithm can only report failure if $a \in L$ is chosen, where $L = \{a \in \mathbb{Z}_n^*, \text{ when } a^{t\,2^j} = 1, \text{then } a^{t\,2^{j-1}} = \pm 1, \text{for } 1 \leq j \leq h\}$.

Similary to the Miller-Rabin test, it can be shown that for $n = \prod_{i=1}^r p_i^{e_i}$, where $r \geq 2$ and p_i are odd primes:

$$|L| \leq \frac{2}{2^r} |Ker\,\rho_{t2^g}| \leq \frac{1}{2} |\mathbb{Z}_n^*|,$$

where $\rho_{t2^g} : x \mapsto x^{t2^g}$, $g = \min\{h, h_1, \ldots, h_r\}$, $m = t2^h$, $\varphi(p_i^{e_i}) = t_i 2^{h_i}$ and t, t_i are odd.

Factorising n by φ(n)

Time complexity

- If $m \in O(n)$ (which is true for $\varphi(n)$), then the algorithm needs time $O(\mathrm{len}(n)^3)$. The expected number of iterations before a success is two.
- If $n = d_1 d_2$, then $\lambda(d_i) \mid \lambda(n) \mid m$ and the algorithm can be used recursively. There will be at most $O(\mathrm{len}(n))$ recursive calls of the algorithm.
- Verifying primality or perfect powers takes roughly $O(\mathrm{len}(n)^3)$ (see below).
- We obtain the full Factorising n from the knowledge of a multiple of $\lambda(n)$ in time about $O(\mathrm{len}(n)^4)$.

Factorising n by φ(n)

Time complexity

Our algorithm, which with finds the non-trivial factor of n from knowledge of m, where $\lambda(n) \mid m$, works only for odd $n \neq p^e$, p is prime. But this is sufficient:

- For even $n = 2^i \tilde{n}$ we find $\tilde{n}$ in time $O(\mathrm{len}(n))$. Then we factorize $\tilde{n}$ with our algorithm, since $\lambda(\tilde{n}) \mid m$.
- We find the perfect power $n = \tilde{n}^e$ in time $O(\mathrm{len}(n)^3 \,\mathrm{len}(\mathrm{len}(n)))$ and factorize $\tilde{n}$ since $\lambda(\tilde{n}) \mid m$.
- We can check primality of $n = p$ by the Miller-Rabin test $MR(\cdot, k)$ in time $O(k \,\mathrm{len}(n)^3)$ and no longer factorize it.

Algorithm for recognising a perfect power

Calculating the integer square root

Input: $n \in \mathbb{N}$
Output: $m = \lfloor \sqrt{n} \rfloor$
Note: If $2^{l-1} \leq n < 2^l$, then $2^{\frac{l-1}{2}} \leq m < 2^{\frac{l}{2}}$.
We will calculate the square root of n by bits.

- $k \leftarrow \lfloor \frac{\mathrm{len}(n)-1}{2} \rfloor$
- $m \leftarrow 0$
- for $i \leftarrow k$ down to 0 do
    - if $(m + 2^i)^2 \leq n$ then $m \leftarrow m + 2^i$ endif enddo
- output m

The time complexity is $O(\frac{\mathrm{len}(n)}{2} \mathrm{len}(n)^2) = O(\mathrm{len}(n)^3)$.

Algorithm for recognising a perfect power

Calculating the integer e−th square root

Input: $n \in \mathbb{N}$
Output: $m = \lfloor \sqrt[e]{n} \rfloor$
Note: If $2^{l-1} \leq n < 2^l$, then $2^{\frac{l-1}{e}} \leq m < 2^{\frac{l}{e}}$.

- $k \leftarrow \lfloor \frac{len(n)-1}{e} \rfloor$
- $m \leftarrow 0$
- for $i \leftarrow k$ down to 0 do
  - if $(m + 2^i)^e \leq n$ then $m \leftarrow m + 2^i$ endif enddo
- output m

The time complexity is $O(\frac{1}{e}\,\mathrm{len}(n)^3)$.

Algorithm for recognising a perfect power

Algorithm for recognising a perfect power

Input: $n \in \mathbb{N}$
Output: answer to the question if $n = m^e$ for some $m, e \in \mathbb{N}$.
Note: $m \geq 2$, $2 \leq e \leq \operatorname{len}(n) + 1$

- for $e \leftarrow 2$ to $\operatorname{len}(n) + 1$ do
  - $m \leftarrow \lfloor \sqrt[e]{n} \rfloor$
  - if $m^e = n$ then output m, e and return *true* endif enddo
- return *false*

The time complexity is $O(\sum_{e=2}^{\operatorname{len}(n)+1} \frac{1}{e} \operatorname{len}(n)^3)$,
replacing the sum by the integral, we get $O(\operatorname{len}(n)^3 \operatorname{len}(\operatorname{len}(n)))$.

Factorising n by φ(n)

Literature

- Shoup: A Computational Introduction to Number Theory and Algebra. Chapter 10.
  http://shoup.net/ntb/