

2.2 Computing the Key and Factorization

Question: How to compute the private RSA exponent d , given the public exponent e and the module n ?

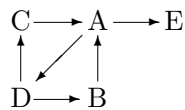
Answer: Each of the following tasks (A) – (D) is efficiently reducible to each of the other ones:

- (A) Computing the private key d .
- (B) Computing $\lambda(n)$ (CARMICHAEL function).
- (C) Computing $\varphi(n)$ (EULER function).
- (D) Factoring n .

Breaking RSA is the (possibly properly) easier task:

- (E) Computing e -th roots in $\mathbb{Z}/n\mathbb{Z}$.

The “proof” (not an exact proof in the mathematical sense) follows the roadmap:



We always assume that n and the public exponent e are known, and $n = p_1 \cdots p_r$ with different primes p_1, \dots, p_r .

Clearly “A \rightarrow E”: Taking an e -th root means raising to the d -th power. So if d is known, computing e -th roots is easy.

Note that the converse implication is unknown: *Breaking RSA could be easier than factoring*.

“D \rightarrow C”: $\varphi(n) = (p_1 - 1) \cdots (p_r - 1)$.

“D \rightarrow B”: $\lambda(n) = \text{kgV}(p_1 - 1, \dots, p_r - 1)$.

“B \rightarrow A”: Compute d by congruence division from $de \equiv 1 \pmod{\lambda(n)}$.

“C \rightarrow A”: Since $\varphi(n)$ has exactly the same prime factors as $\lambda(n)$, also $\varphi(n)$ is coprime with e . From $de \equiv 1 \pmod{\varphi(n)}$ we get a solution for d by congruence division. This might not be the “true” exponent, but works in the same way as private key since a fortiori $de \equiv 1 \pmod{\lambda(n)}$.

“A \rightarrow D” is significantly more involved. Moreover we only construct a probabilistic algorithm.

Preliminary Remarks

1. *It suffices to decompose n into two proper factors.*

(a) Let $n = n_1 n_2$ be a proper decomposition, and assume for simplicity that $n_1 = p_1 \cdots p_s$ with $1 < s < r$. Then

$$\lambda(n_1) = \text{kgV}(p_1 - 1, \dots, p_s - 1) \mid \text{kgV}(p_1 - 1, \dots, p_r - 1) = \lambda(n),$$

thus also $de \equiv 1 \pmod{\lambda(n_1)}$. This reduces the problem to the analogous ones for n_1 and n_2 .

(b) Since the number of prime factors of n is at most $\log_2(n)$ the recursive reduction suggested by (a) is efficient.

2. *How can a residue class $w \in \mathbb{Z}/n\mathbb{Z}$ help with factoring n ?*

(a) Finding a $w \in [1 \dots n - 1]$ with $\text{gcd}(w, n) > 1$ decomposes n since $\text{gcd}(w, n)$ is a proper divisor of n .

(b) Finding a $w \in [2 \dots n - 2]$ with $w^2 \equiv 1 \pmod{n}$ (that is a non-trivial square root of 1 in $\mathbb{Z}/n\mathbb{Z}$) likewise decomposes n :

Since $n \mid w^2 - 1 = (w + 1)(w - 1)$ and $n \nmid w \pm 1$ we have $\text{gcd}(n, w + 1) > 1$, and this decomposes n by (a).

Now let (d, e) be a pair of RSA exponents. Then also $u := ed - 1 = k \cdot \lambda(n)$ is known (with unknown k and $\lambda(n)$). Since $\lambda(n)$ is even we may write

$$u = r \cdot 2^s \quad \text{with } s \geq 1 \text{ and } r \text{ odd.}$$

If we choose a random $w \in [1 \dots n - 1]$, then we have to deal with two possibilities:

- $\text{gcd}(w, n) > 1$ —then n is decomposed.
- $\text{gcd}(w, n) = 1$ —then $w^{r \cdot 2^s} \equiv 1 \pmod{n}$.

In the second case we efficiently find the minimal $t \geq 0$ with

$$w^{r \cdot 2^t} \equiv 1 \pmod{n}.$$

Again we distinguish two cases:

- $t = 0$ —bad luck, choose another w .
- $t > 0$ —then $w^{r \cdot 2^{t-1}}$ is a square root $\neq 1$ of 1 in $\mathbb{Z}/n\mathbb{Z}$.

In the second case we distinguish:

- $w^{r \cdot 2^{t-1}} \equiv -1 \pmod{n}$ —bad luck, choose another w .

- $w^{r2^{t-1}} \not\equiv -1 \pmod{n}$ —then n is decomposed by preliminary remark 2.

Thus every choice of $w \in [1 \dots n - 1]$ has one of four possible outcomes, two of them decompose n , and the other two flop. Denote the last two events by

$$\begin{aligned} (\mathbf{E}_{n,u}(w)/\mathbf{I}) \quad w^r &\equiv 1 \pmod{n} \\ (\mathbf{E}_{n,u}(w)/\mathbf{II}) \quad w^{r2^{t-1}} &\equiv -1 \pmod{n} \quad \text{for a } t \text{ with } 1 \leq t \leq s. \end{aligned}$$

Altogether this yields a tree-like structure:

$$\begin{aligned} w \in [1 \dots n - 1] &\longrightarrow \\ \gcd(w, n) > 1 &\longrightarrow n \text{ decomposed SUCCESS} \\ w \in \mathbb{M}_n &\longrightarrow \\ w^r &\equiv 1 \pmod{n} \longrightarrow (\mathbf{E}_{n,u}(w)/\mathbf{I}) \text{ FLOP} \\ w^r &\not\equiv 1 \pmod{n} \longrightarrow \\ w^{r2^{t-1}} &\equiv -1 \pmod{n} \longrightarrow (\mathbf{E}_{n,u}(w)/\mathbf{II}) \text{ FLOP} \\ w^{r2^{t-1}} &\not\equiv -1 \pmod{n} \longrightarrow n \text{ decomposed SUCCESS} \end{aligned}$$

Thus our procedure decomposes n “with high probability” if there are only “few” “bad” integers w with $(\mathbf{E}_{n,u}(w)/\mathbf{I,II})$. The next section will provide an upper bound for their number.