Computer Networks

Lecture 9: RSA cryptography

Marcin Bieńkowski
Translation: *Krystian Bacławski*

9.1 RSA algorithm

Firstly, we generate two keys (public and private) in following manner:

1. Choose $p \neq q$: two large prime numbers.
2. Calculate $n = p \cdot q$.
3. Find large number d co-prime to $(p-1) \cdot (q-1)$.
4. Find e such as $d \cdot e \mod (p-1) \cdot (q-1) = 1$ holds (with extended Euclid's algorithm).
5. Pair (e, n) is ours public key and (d, n) private key.

How to encrypt our message? We write it down in binary form and split it into fragments of length not exceeding $\log n$. Thus, each of the fragments is a number in the range $[0, n)$. Each of the numbers we encrypt separately. [1]

Let us assume, therefore, that we want to encrypt a number $m \in [0, n)$. Let's calculate the number:

$$E(m) = m^e \mod n \ ,$$

and send it as ciphertext s to the recipient. After he receives the ciphertext, he decrypts it calculating:

$$D(s) = s^d \mod n \ .$$

9.1.1 Why does it work?

We need to prove that for any $m \in [0, n)$ following $D(E(m)) = m$ holds. We're going to show it for m greater than zero and co-prime to n. Proving that for other values of m is left out to the reader.

$$\begin{aligned} D(E(m)) &= (m^e \mod n)^d \mod n \\ &= (m^e)^d \mod n && \text{(modulo property)} \\ &= m^{k \cdot (p-1) \cdot (q-1) + 1} \mod n && (k \in \mathbb{N} \cup \{0\}) \\ &= 1^k \cdot m^1 \mod n && \text{(Euler's totient theorem)} \\ &= m \end{aligned}$$

Twierdzenie 9.1 (Euler's totient theorem)**.** *For any positive natural number n, let $\mathbb{Z}_n^* = (\{a : 1 \leq a \leq n \wedge a \perp n\}, \cdot \mod n)$ be a group, whose elements are co-prime to n, and let the operation be multiplication modulo n. Let $\phi(n)$ (Euler's totient function) be number of elements in such a group. Then, for $m \in \mathbb{Z}_n^*$ equation $m^{\phi(n)} \equiv 1 \mod n$ holds.*

[1] This naive approach leads to the fact that the same fragments would be encrypted in the same way. In practice, different approaches are used to work around this problem (e.g. append a random string).