Kryptografia

Tomasz Perła

17.04.2025

Spis Treści

1 Algorytm Euklidesa

Algorytm Euklidesa jest oparty na zależnościach:

$$NWD(a,b) = NWD(b, a\ mod\ b) \text{ dla } a \geq b \tag{1}$$

$$NWD(a,0) = a \tag{2}$$

W ogólnym przypadku działanie algorytmu Euklidesa można przestawić następująco:

1. $r_0 = a$; $r_1 = b$
2. $r_{i-1} = r_i q_i + r_{i+1}$
3. IF $r_{i+1} > 0$ THEN i++; goto 2
4. IF $r_{i+1} = 0$ THEN $NWD(a,b) = r_i$

W trakcie działania algorytmu Euklidesa można wyznaczyć takie liczby $x, y \in \mathbb{Z}$, że $ax - by = 1$.

$$P_n = \begin{cases} 0, & \text{jeśli } \; n = -2 \\ 1, & \text{jeśli } \; n = -1 \\ q_n P_{n-1} + P_{n-2}, & \text{jeśli } \; n \in \mathbb{N}_0 \end{cases} \tag{3}$$

$$Q_n = \begin{cases} 1, & \text{jeśli } \; n = -2 \\ 0, & \text{jeśli } \; n = -1 \\ q_n Q_{n-1} + Q_{n-2}, & \text{jeśli } \; n \in \mathbb{N}_0 \end{cases} \tag{4}$$

Zatem:

$$\begin{cases} x = (-1)^{k-1} Q_{k-1} \\ y = (-1)^{k-1} P_{k-1} \\ P_k = a \\ Q_k = b \end{cases} \tag{5}$$

Wtedy:

$$P_k(-1)^{k-1} Q_{k-1} - Q_k(-1)^{k-1} P_{k-1} = 1 \tag{6}$$

Wygodnym jest przeprowadzić te obliczenia za pomocą tabeli.

n	-2	-1	0	1	2	$\dots$	$k-1$	k
q_n			q_0	q_1	q_2	$\dots$	q_{k-1}	q_k
P_n	0	1	P_0	P_1	P_2	$\dots$	P_{k-1}	P_k
Q_n	1	0	Q_0	Q_1	Q_2	$\dots$	Q_{k-1}	Q_k

1.1 Przykład

Wyznaczyć takie liczby $x, y \in \mathbb{Z}$, że $17x - 13y = 1$.

$$\begin{aligned} 17 &= 13 \cdot 1 + 4 & q_0 &= 1 \\ 13 &= 4 \cdot 3 + 1 \text{ (NWD)} & q_1 &= 3 \\ 4 &= 1 \cdot 4 + 0 & q_2 &= 4 \end{aligned}$$

n	-2	-1	0	1	$k=2$
q_n			1	3	4
P_n	0	1	1	4	17
Q_n	1	0	1	3	13

Zatem:

$$\begin{cases} x = (-1)^{k-1} Q_{k-1} = -3 \\ y = (-1)^{k-1} P_{k-1} = -4 \\ 17(-3) - 13(-4) = 1 \end{cases} \tag{7}$$

1.2 Przykład

Znajdowanie elementów odwrotnych w $\mathbb{Z}_n$. Obliczyć $8^{-1} \equiv 1 \pmod{35}$ (tzn. rozwiązać równanie $8x \equiv 1 \pmod{35}$).

$$\begin{aligned} 8 &= 35 \cdot 0 + 8 & q_0 &= 0 \\ 35 &= 8 \cdot 4 + 3 & q_1 &= 4 \\ 8 &= 3 \cdot 2 + 2 & q_2 &= 2 \\ 3 &= 2 \cdot 1 + \mathbf{1} \text{ (NWD)} & q_2 &= 1 \\ 2 &= \mathbf{1} \cdot 2 + 0 & q_2 &= 2 \end{aligned}$$

n	-2	-1	0	1	2	3	$k=4$
q_n			0	4	2	1	2
P_n	0	1	0	1	2	3	8
Q_n	1	0	1	4	9	13	35

Zatem:

$$x = (-1)^{k-1} Q_{k-1} = -Q_3 = -13 \pmod{35} = 22 \pmod{35} \tag{8}$$
$$22 \cdot 8 = 176 = 35 * 5 + 1 \equiv 1 \pmod{35}$$

1.3 Implementacja w języku Java

Implementacja pozostaje jako ćwiczenia dla czytelnika.

2 Algorytm RSA

Algorytm RSA szyfruje jednostki tekstu jawnego m takie, że $m \in [0; N) \cap \mathbb{N}_0$. W praktyce liczba N ma około 200 do 600 cyfr dziesiętnych. Bezpieczeństwo szyfrowania opiera się na trudności faktoryzacji dużych liczb złożonych.

1. Losujemy dwie "duże" (zgodnie z zaleceniami minimum 2048 bitów) liczby pierwsze $p, q \in \mathbb{P}$ spełniające dodatkowo zależność:

$$n = pq > N$$

2. Losowo wybieramy liczbę $e \in \mathbb{P}$ o takiej samej długości bitowej co n, spełniającą dodatkowo zależności:

$$\begin{cases} e < \phi(n) = (p-1)(q-1) \\ NWD(e, \phi(n)) = 1 \end{cases} \iff e \in \mathbb{Z}^*_{\phi(n)}$$

Gdzie:
$\mathbb{Z}_n = \{0, 1, \ldots, n-1\}$ dla $n \in \mathbb{N}_1 \setminus \{1\}$
$x = y^{-1} \mod n$ - x jest elementem odwrotnym do $y \mod n$. $xy \equiv 1 \pmod n$
$\mathbb{Z}^*_n$ - zbiór elementów, dla który istnieją elementy odwrotne w $\mathbb{Z}_n$

$$\mathbb{Z}^*_n = \{x \in \mathbb{Z}_n : \exists\, y \in \mathbb{Z}_n, xy \equiv 1 \pmod n\} \iff$$
$$\iff \mathbb{Z}^*_n = \{x \in \mathbb{Z}_n \setminus \{0\} : NWD(x, n) = 1\}$$

W szczególności dla $\mathbb{Z}_p$ gdzie $p \in \mathbb{P}$:

$$\mathbb{Z}^*_p = \mathbb{Z}_p \setminus \{0\}$$

3. Za pomocą algorytmu Euklidesa oblicza sie d:

$$ed \equiv 1 \pmod{\phi(n)}$$

$$d = e^{-1} \mod \phi(n)$$

$$\begin{cases} \text{Para } (e, n) \text{ jest kluczem publicznym} \\ \text{Para } (d, n) \text{ jest kluczem prywatnym} \end{cases}$$

2.1 Szyfrowanie

Szyfrowane mogą być liczby $m \in [0; n) \cap \mathbb{N}_0 = \mathbb{Z}_n$:

$$E : \mathbb{Z}_n \to \mathbb{Z}_n, \quad E(m) = m^e \mod n \tag{9}$$

Liczbę $c = E(m)$ nazywamy kryptogramem.

2.2 Deszyfrowanie

$$D : \mathbb{Z}_n \to \mathbb{Z}_n, \quad D(c) = c^d \mod n \tag{10}$$

W wyniku czego otrzymujemy tekst jawny $m = D(c)$.
Zasadniczo zachodzi zależność:

$$D(E(m)) = m \wedge E(D(c)) = c$$

$$D \circ E = \mathrm{id}_m \wedge E \circ D = \mathrm{id}_c$$

$$E = D^{-1} \wedge D^{-1} = E$$

3 Podpis cyfrowy

Podpisy tworzone za pomocą RSA:

1. Dla dokumentu M jest obliczana wartość $H(M)$, gdzie H jest funkcją haszującą
2. Tomasz szyfruje skrót $H(M)$ za pomocą klucza prywatnego używając RSA. Kryptogram $c = E(H(M))$ jest nazywany podpisem cyfrowym.
3. Sprawdzenie podpisu Tomasza polega na deszyfrowaniu podpisu kluczem publicznym Tomasza i sprawdzeniu czy $D(c) = H(M)$

4 Ślepy podpis cyfrowy

Sytuacja, w której Tomasz chce uzyskać ślepy podpis Notariusza pod wiadomością. Notariusz wykorzystuje klucz publiczny (e, n) i swój klucz prywatny (d, n) dla szyfrowania RSA.

1. Obliczamy dla wiadomości m' funkcje skrótu $H(m')$ i przypisujemy $m = H(m')$
2. Wybieramy losowo liczbę $k < n \wedge NWD(k, n) = 1$.
$$t = m \cdot k^e \mod n$$
3. Notariusz szyfruje (podpisuje) t za pomocą swojego klucza prywatnego.
$$s = t^d \mod n$$
4. Zatem Tomasz oblicza $m^d = s \cdot k^{-1} \mod n$. Wartość $m^d \mod n$ jest podpisem wiadomości m.
5. W celu weryfikacji są wykonywane te same operacje i otrzymana wartość jest porównywana z zadanym podpisem.

5 Podziękowania

Dziękuję tylko i wyłącznie sobie.