Tema 9 - Criptografie

1. Implementarea criptosistemului Rabin

Criptosistemul Rabin este un criptosistem asimetric bazat pe problema factorizării numerelor întregi mari. Să implementăm criptosistemul Rabin în C++:

- Alegem două numere prime mari: p, q
- $n = p \times q$
- cheia publică: $K_e = n$
- cheia privată: $K_d = (p, q)$

Criptare:
- $m \in \mathbb{Z}$ e $\{0, 1, \dots n-1\}$
- $c = m^2 \bmod n$
- scriem c

Decriptarea:
- "citim" c
- det. rădăcinile pătrate ale lui c mod p și q
- folosim Teorema Chineză a Resturilor pentru a calcula cele patru rădăcini pătrate modulo n.
- verificăm enunțurile pentru a alege rădăcina pătrată corectă.

Rabin.cpp

2. Implementarea Massey-Omura

! Este un algoritm de criptare bazat pe exponențierea modulară.

Generarea cheilor:

alegem un numar prim: p

utilizatorul alege un exponent "secret" e a.î. e mmdc de $(e, p-1) = 1$

fiecare utilizator calculează inversul modular $d = e^{-1} \bmod (p-1)$.

Criptarea & Decriptarea:

1. Bob criptează mesajul m trimițând $C_1 = m^{eB} \bmod p$ lui Alice
2. Alice criptează C_1 trimițând $C_2 = C_1^{eA} \bmod p$ inapoi la Bob
3. El va decripta C_2, apoi trimite $C_3 = C_2^{dB} \bmod p$ lui Alice
4. Aceasta decriptează C_3 și obține mesajul $m = C_3^{dA} \bmod p$

Massey Omura.cpp

3. Criptasistemul Merkle-Hellman

! Acesta este bazat pe problema rucsacului, întâlnită și în liceu, dar și la facultate.

Generarea cheilor:

1. Alegem un șir supercrescător $\{v_0, v_1, \dots v_{k-1}\}$
2. Alegem un modul m a.î. $m > \sum v_i$
3. α a.î. c.m.m.d.c $(\alpha, m) = 1$
4. inversul: $b = \alpha^{-1} \bmod m$
5. șirul $\{w_0, w_1, \dots w_{k-1}\}$ unde $w_i = \alpha \cdot v_i \bmod m$

Criptare: vom reprezenta mesajul m într-un vector binar $\{e_0, e_1, \dots e_{k-1}\}$

calculăm: $c = \sum e_i \cdot w_i$

returnăm c

Decriptarea:

primim c

calc. $V = b \cdot c \bmod m$

reprezentăm pb. rucsacului pentru V folosind șirul $\{v_0, v_1, \dots v_{k-1}\}$

MerKle Hellman.cpp

4. Problema rucsacului în general.

Problema aceasta, Knapsack este de optimizare combinatorială. În varianta clasică, 0-1, avem o

mulțime de obiecte, fiecare cu o valoare și o greutate. Rucsacul are o capacitate limitată. Trebuie maximizată valoarea totală a obiectelor din rucsac fără a depăși capacitatea.

rucsac.cpp

20.

V = 473

Pentru a determina un șir supracrescător minim și a rezolva pb rucsacului pt V = 473

Plan "de atac": generăm un șir ~~poate~~ supracrescător minim, apoi vom rezolva problema rucsacului pt valoarea V = 473

tema9_20rucsac.cpp