Rezolvare examen
Criptografie & Securitate

Gabriel Majeri

13 mai 2021

Exercițiul 1

(a) Într-un sistem de criptare perfect sigur lungimea cheii trebuie să fie cel puțin la fel de mare cu lungimea mesajului care trebuie criptat, conform teoremei lui Shannon.

Pentru un cifru OTP, lungimea cheii este egală cu lungimea mesajului original care este egală cu lungimea mesajului criptat. Având în vedere că mesajul criptat are lungime 10 bytes, la fel de mare trebuie să fi fost și cheia.

(b) Având în vedere modul în care funcționează OTP și proprietățile operatorului XOR, avem că $C' = K \oplus M = M \oplus M = 0$. Deci noul mesaj criptat primit de la Alice este șirul cu toți biții 0.

Având în vedere că noul mesaj criptat s-a obținut cu o altă „cheie”, nu a fost încălcat principiul de funcționare al OTP. Dacă Alice nu ne transmite noul K într-un mod sigur (adică de fapt M), nu avem cum altcumva să obținem mesajul original.

Exercițiul 2

(a) Mesajul criptat este $C = (c_1|c_2|\dots|c_n)$, unde $c_i = F_k(m_i \oplus (ctr + i))$ pentru un mesaj $M = (m_1|m_2|\dots|m_n)$.

(b) Mesajul decriptat este se poate obține cu formula $m_i = F_k^{-1}(c_i) \oplus (ctr + i)$.

(c) Sistemul nu este corect deoarece apare o ambiguitate în cazul mesajelor care se termină cu zerouri. Nu putem distinge între un mesaj inițial care nu are nevoie de padding, dar care are niște bytes de zero la final, și unul care are nevoie de padding, și care se termină cu zero pentru că i-am adăugat noi.

De exemplu, dacă dimensiunea unui bloc este 2 bytes, mesajul `FF 00` este deja egal cu lungimea blocului, deci nu are nevoie de padding. Dacă luăm mesajul `FF`, trebuie să-i adăugăm padding, și obținem `FF 00`, care este la fel cu mesajul de mai sus.

Aceeași problemă apare și dacă se realizează padding indiferent de situație. Dacă reluăm exemplul de mai sus, nu putem distinge între mesajul `FF 00` care primește padding ca `FF 00 00 00` și `FF 00 00` care devine tot `FF 00 00 00`.

(d) Acest sistem nu este CCA-sigur (de fapt, nici CPA-sigur) deoarece alege să facă XOR între mesaj și contor înainte de aplicarea algoritmului de criptare bloc.

Ne imaginăm că suntem în experimentul CPA. Alegem ca mesaje $M_1 = (1, 2)$ și $M_2 = (0, 0)$, adică $(01, 10)$ și $(00, 00)$ în binar.

Dacă challenger-ul alege mesajul M_1 și îl criptează, e suficient să ne uităm la ce se întâmplă cu ultimii doi biți ai contorului:

$$\begin{aligned} \text{ctr} + 1: &\quad 01 \oplus 01 = 00 \\ \text{ctr} + 2: &\quad 10 \oplus 10 = 00 \end{aligned}$$

Dacă alege M_2:

$$\begin{aligned} \text{ctr} + 1: &\quad 01 \oplus 00 = 01 \\ \text{ctr} + 2: &\quad 10 \oplus 00 = 10 \end{aligned}$$

(am presupus că *ctr* se termină în 00, dar se obțin rezultate similare în orice caz)

Dacă a ales M_1, vom obține cu siguranță două blocuri criptate care au aceeași valoare. Altfel, vom obține două blocuri criptate diferite. Putem determina cu probabilitate 100% ce mesaj a fost criptat.

Sistemul nu este CPA-sigur deci nici CCA-sigur.

(e) Paradoxul zilei de naștere ne spune că după ce am strâns radical din numărul total de blocuri care pot să existe, șansa ca cel puțin două dintre ele să se repete este mai mare de 50%. Deci, pentru un cifru cu lungimea blocului de 48 de biți, avem 2^{48} blocuri posibile, iar paradoxul ne spune că ne așteptăm să se repete un bloc după aproximativ $\sqrt{2^{48}} = 2^{24}$ blocuri.

În practică, asta înseamnă că pentru acest cifru nu e recomandat să criptăm mai mult de $2^{24} \times 48$ biți = 768MB de date cu aceeași cheie.

Exercițiul 3

(a) Nu, deoarece funcția de hashing MD5 nu mai este considerată sigură de mult timp.

(b) G nu este un PRG sigur, deoarece $G(x) = 0$ pentru orice x (se vede ușor că x^2 este multiplu de x, deci dă rest 0 la împărțirea cu x). Un algoritm determinist care să distingă pe G ar putea fi o funcție care returnează 1 de fiecare dată când input-ul este egal cu 0.

(c) Ni se spune că *CryptStream* este un sistem de criptare fluid, deci $c = G(x) \oplus m$.

(d) Funcția f este de forma $\frac{1}{\text{polinom}(n)}$, deci nu este neglijabilă. Protocolul în acest caz este nesigur.

(e) Dacă se rezolvă problema de la ultimul subpunct, în continuare rămâne problematic faptul că protocolul utilizat de companie este *neautentificat*. Asta îl face susceptibil al atacuri de tip *man-in-the-middle*.

Autentificarea se poate realiza în diferite moduri (de exemplu cu acest protocol sau prin sistemul PKI), dar în principiu cei doi participanți la conversație vor folosi semnături digitale oferite de criptografia asimetrică.

(f) *AuthMAC* nu este un sistem de autentificare sigur. Este suficient să știm o pereche m și $Mac(k, m)$. Calculând $h(m) \oplus Mac(k, m)$ obținem $h(k||len(m))$. În acest moment putem falsifica orice mesaj m' vrem noi de lungime egală cu cea a lui m. Tot ce trebuie să facem este să calculăm $m' \oplus h(k||len(m))$.

(g) Un principiu care se încalcă este, de exemplu, principiul lui Kerckhoff, care susține că securitatea unui sistem criptografic nu ar trebui să fie obținută prin ascunderea informațiilor despre metoda lui de funcționare.

Din enunț aflăm că *AuthMAC* este un sistem de autentificare proprietar, știut doar de persoane care au semnat un NDA. Asta demonstrează că firma respectivă a încercat pe cât posibil să ascundă modul de funcționare al algoritmului.

(h) Un obiectiv al criptografiei este **integritatea**: mesajele transmise nu ar trebui să poată fi modificate de un atacator. Din enunț reiese că se folosește exclusiv *CryptStream* pentru comunicarea cu clienții, care asigură confidențialitatea dar nu și integritatea datelor.

Un alt obiectiv al criptografiei este **autentificarea** părților care participă la o comunicare. Din enunț, reiese că firma folosește protocolul Diffie-Hellman neautentificat pentru comunicarea internă.

Exercițiul 4

(a) Numărul dat are 2048 de biți. Nu este practic să-l factorizăm. Ca e să fie valid, ar trebui să fie $\leq \phi(N)$ și prim față de $\phi(N)$.

Având în vedere că N este produsul a două numere prime p și q de 2048 de biți, și că $\phi(N) = (p-1)(q-1)$, este foarte clar că se respectă prima condiție.

De asemenea, 65537 este număr prim, deci respectă și a doua condiție.

(b) Inversul unui punct (x, y) de pe o curbă eliptică este punctul $(x, -y)$. Deci inversul lui $(8, 10)$ este $(8, -10) = (8, 19) \mod 29$. Inversul lui $(8, 11)$ este $(8, 18)$.

(c) Este destul de ușor de spart deoarece numărul de puncte pe curbă este relativ mic. Lucrând în $\mathbb{Z}_{29}$, putem avea maxim 29^2 perechi (x, y) distincte. Un calculator obișnuit poate să verifice destul de rapid toate perechile și să obțină lista tuturor punctelor de pe curbă.

(d) Am ales curba **P-192** recomandată de NIST în acest document. Ecuația ei este

$$y^2 = x^3 - 3x + b \mod p$$

unde

$p = 6277101735386680763835789423207666416083908700390324961279$

și

$$b = 64210519e59c80e70fa7e9ab72243049feb8deecc146b9b1$$

(în hexa)