

-Examenem -

Subiecte Nefractante: 3R; 3d

Resolvare:

1. a) • i) HASH:

Există mai multe avantaje atunci când folosim funcțiile hash (dar și dezavantaje, în funcție de context):

- când folosim hash, textul devine nedescifrabil și nu mai poate fi întors la forma lui originală;^(nu e)
- toate mesajele originale, indiferent de lungimea lor, folosind același algoritm de hash, se vor transforma în texte hash-uite de lungime fixă

Datorită acestor aspecte, funcțiile hash sunt des folosite la stocarea parolelor în baze de date. Spre ex., avem un site web care ne cere un cont de utilizator (Facebook, Enigam etc.). Nu există niciun motiv pt. care parola stocată să fie ~~ștește~~ "citibilă". Aici ea va fi hash-uită și reținută într-o bază de date, iar atunci când încercăm să ne logăm din nou, se vor compara versiunea din baza de date cu forma hash-uită a parolei introduse de noi.

• ii) CRIPTARE:

Spre deosebire de hash, criptarea este reversibilă și ~~lungimea~~ lungimea mesajului criptat este egală cu lungimea originală.

lungimea mesajului sursă. Dacă folosim, spre ex., la o aplicare de stocat paralele, doar ce noi vrem să putem să primim paralel, în forma originală, dar ele să fie criptate, pt. oricare altcineva; dacă am folosi hash, nu am niciun putut accesa forma originală a paralei.

b) Sistemul de criptare e reversibil și folosește o cheie de criptare, dar un mesaj fără cheie încă poate fi aplicat o funcție hash nu mai poate fi întors la forma originală (nu avem o cheie). De asemenea, mesajul criptat are lungimea mesajului sursă; dar la mesajul hash-uit aveam o lungime fixă, în funcție de ce algoritm folosim.

c) Nu putem vorbi de securitate perfectă în cazul funcțiilor hash pt. că există (sau pot exista) coliziuni.

Să nu gândim în situația fel: aveam un număr infinit de texte, de lungimi variate, care doar nu să fie hash-uite și vom obține același rezultat de același lungime. Deci există probabilitatea ca 2 texte diferite să dea același hash. (În 2016, spre ex., Google a denunțat că SHA-1 este vulnerabil și astfel s-a trezit la SHA-2: 256 / 512 biti). Însă computațional este sigur, deoarece ne-ar lua mult timp să spargem un mesaj hash-uit folosind brute-force.

$$2. a) C_i = P_i \oplus E_K(\text{Counter } 0 + i)$$

Ciphertext Plaintext

Functie criptare cu cheia "K"

Auth Tag = $E_K(\text{Counter } 0) \oplus r_{u_0}$, unde:

$$\left\{ \begin{array}{l} r_{u_0} = \text{mult}_H(\text{Auth Data } 1) \\ r_{u_i} = \text{mult}_H(r_{u_{i-1}} \oplus C_i) \end{array} \right.$$

$$\vdots$$

$$r_{u_{q-1}} = \text{mult}_H(r_{u_{q-2}} \oplus (\text{len}(A) // \text{len}(C)))$$

↳ multiplication cu cheia hash "H" în corpul (2^{128})

b) $\text{Tag}' = E_K(\text{Counter } 0) \oplus r_{u_0}$

$$P_i = E_K(C_i) \oplus (\text{Counter } 0 + i)$$

c) Mesaj criptat = (C_1, C_2, \dots, C_m) (concatenarea cipertextelor)

$$320 : 128 = 2 \text{ r } 64 \Rightarrow$$

| -urile

\Rightarrow deci urmărt în 3 plaintext-uri și la al III-lea adaug padding ca să ajungă la 128. Deci lungimea mesajului criptat = $3 \cdot 128 = 384$

d) 128 (are deasă operatia de xorare între texte de 128 biti)

e) La CTR, atacatorul poate să modifice textul criptat și Bob să nu își dea seama de acest lucru. GCM aduce un layer în plus de protecție, deoarece are și "autentificarea": Alice trimite acel tag, obținut din funcție hash, și astfel, Bob poate să verifice autenticitatea mesajului criptat.

f) Fals

Lungimea nu ne garantează securitate perfectă, lungimea face ca atacările brute-force să nu fie fezabile, asigurând securitatea computatională (la momentul actual).

3. a) LINK: <http://apo.it/arsusjs/>

$$x = 65537$$

Nr biti N = 2048

b) (k, s) sunt prime între ele.

$$5) a) s = k \oplus a$$

$$u = s \oplus b = k \oplus a \oplus b$$

$$w = u \oplus a = k \oplus a \oplus b \oplus a = k \oplus b$$

Alice $\rightarrow k$

$$\left. \begin{array}{l} \\ \\ \end{array} \right\} \rightarrow \text{Sunt egale}$$

$$Bob \rightarrow w \oplus b = k \oplus b \oplus b = k$$

b) Nu. (Un atacator pasiv nu modifică informația transmisă, doar observă). Oscar/Eve pot să vadă doar s, u și w .

$$s \oplus u \oplus w = k \oplus a \oplus k \oplus a \oplus b \oplus k \oplus b = k$$

Aici atacatorul are acces la cheia de criptare.

c) (Atacator activ modifică/afectează informația transmisă).

Diffie-Hellman însumată fit cu autentificare (semnături digitale). D.-H. nu e sigur în cazul unui atac activ pt că atacatorul poate împersona pe Alice/Bob; de aceea avem nevoie de autentificare în prealabil.

4. a) Se părădește de ce informații se doresc să fie stocate.

Faceți avea o aplicație de comunicare externă, care să protejeze nume, parole, date de naștere, iar, în acest caz, SHA-256 nu e suficient de sigur datorită lungimii minime (pe lită) recomandată este 32. \Rightarrow Coliziuni. Ar fi mai sigur să folosiți SHA-256 cu salt sau SHA-512.

= Examene =

4. b) Rezultent la a 2-a preimagine: "X" dat; e dificil de gasit $X' \neq X$ a.i. $H(X) = H(X')$.

Presupun $a = X \parallel Y$. Din cerinta, $H(a) = f(X \oplus Y)$.

Fie $b = Y \parallel X$. Din cerinta, $H(b) = f(Y \oplus X) \leftarrow$
~~($X \neq Y$, pt. a obtine $a \neq b$)~~

c) Conform punctului "b", am demonstrat ca functia " H " nu produce rezultate diferite pt input-uri diferite, deci nu o putem folosi pe post de semnatura digitala, care trebuie sa fie unica.

d) Trebuie sa folosim o operatie care sa nu fie comutativa, spre exemplu scaderea sau impartirea.

e) Nu se pare o metoda sigura. Chiar daca arun se repeta, PRG este sigur si glucreaza nevoie nr. aleatoare. Totusi sistemul "Fluid" este o abordare moderna si sigura.

f) Principiul lui Kerckhoffs: sistemul nu trebuie sa fie secret, deoarece devine vulnerabil.