Securitatea Rețelor și Serviciilor de Comunicații

Algoritmi criptografici: Autentificarea datelor

Integritatea datelor

Funcții hash criptografice

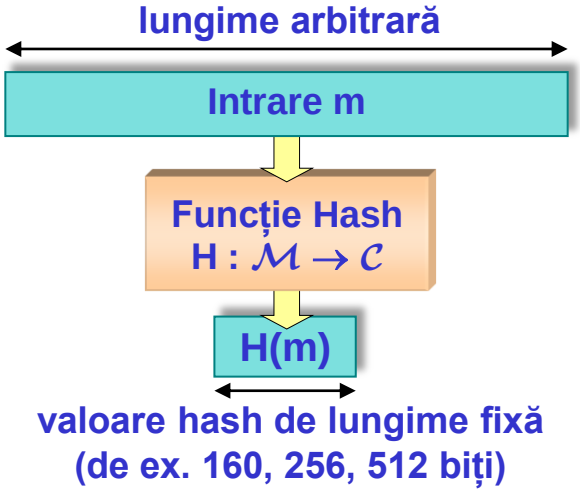
Funcții hash

Definiție: Funcție hash

- O funcție hash este o aplicație H : $\mathcal{M} \to \mathcal{C}$, unde:
  $\mathcal{M} = \{0,1\}^*$ este mulțimea intrărilor, șiruri binare cu lungime arbitrară,
  $\mathcal{C} = \{0,1\}^n$ este mulțimea ieșirilor, șiruri binare cu lungime fixă, n biți.

Proprietăți generale

- Compresie: Asociază fiecărui șir de intrare $m \in \{0,1\}^*$, de lungime arbitrară, un șir de ieșire $H(m)$ de lungime mai mică, fixă (ex: 256 biți).
- Calcul eficient: Există un algoritm eficient care calculează $H(m)$ pentru oricare $m \in \mathcal{M}$.



- Funcțiile hash au numeroase aplicații în informatică (ex: hash table). Se folosesc diverși algoritmi, cu proprietăți adaptate aplicațiilor.
- Funcțiile hash criptografice formează o categorie aparte: sunt proiectate pentru a îndeplini proprietăți specifice de securitate.

Funcții hash în criptografie

Funcții hash criptografice

- Primitive folosite pentru a construi algoritmi criptografici pentru diverse aplicații: integritatea datelor (digital fingerprint), autentificarea datelor cu cheie secretă/publică, funcții pseudoaleatoare (generatoare de șiruri pseudoaleatoare), etc.

Proprietăți de securitate

- Algoritmii folosiți pentru aceste funcții sunt special proiectați pentru a îndeplini proprietățile de securitate necesare în criptografie:
  - Rezistență la preimagine (preimage resistance).
  - Rezistență la a doua preimagine (2nd preimage resistance).
  - Rezistența la coliziuni (collision resistance).

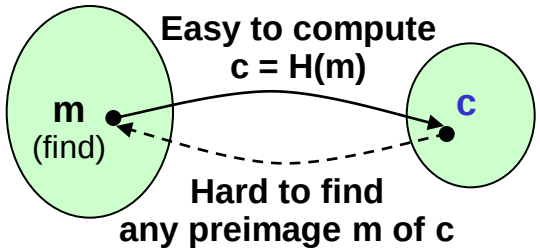
O coliziune a unei funcții hash $\mathrm{H}: \mathcal{M} \to \mathcal{C}$ este o pereche de intrări distincte, $\mathrm{m}_1 \neq \mathrm{m}_2$, cu proprietatea că $\mathrm{H}(\mathrm{m}_1) = \mathrm{H}(\mathrm{m}_2)$. Funcțiile hash asigură compresie a intrărilor, deci există numeroase coliziuni.

Proprietăți de securitate

Rezistența la preimagine

- Pentru oricare $c \in \mathcal{C}$, este dificil de calculat $m \in \mathcal{M}$ astfel încât $H(m) = c$ (orice preimagine m a lui c).
- Căutarea prin forță brută necesită în medie 2^n evaluări ale funcției H.

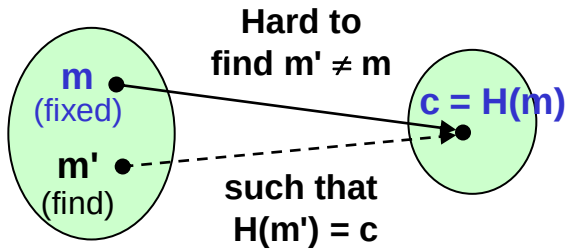
Preimage resistance



Rezistența la a doua preimagine

- Pentru oricare $m \in \mathcal{M}$ (fixat) este dificil de calculat $m' \in \mathcal{M}$ astfel încât $H(m) = H(m')$.
- Căutarea prin forță brută necesită în medie 2^n evaluări ale funcției H.

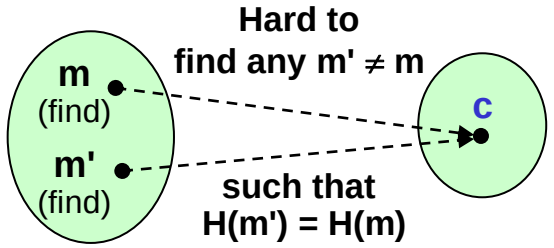
2nd-preimage resistance



Rezistența la coliziuni

- Este dificil de determinat orice pereche $m \neq m'$ astfel încât $H(m) = H(m')$.
- Căutarea prin forță brută necesită în medie $2^{n/2}$ evaluări ale funcției H (birthday paradox).

Collision resistance



Securitatea funcțiilor hash

Relații între proprietăți

- Rezistența la coliziune implică rezistența la a doua preimagine și rezistența la preimagine, deci este cerința cea mai puternică.

  Dacă am putea calcula eficient preimagini, am putea găsi eficient și coliziuni: alegem m arbitrar, calculăm $c = H(m)$ și apoi calculăm o preimagine m' a lui c.

Atacuri asupra funcțiilor hash

- Atacuri prin forță brută: Căutare prin încercări repetate. Contracarată alegând parametrul n suficient de mare. Recomandat: $n \geq 256$ biți.
- Atacuri criptanalitice: Identifică vulnerabilități în algoritmul funcției hash care să permită atacuri mai eficiente decât cele prin forță brută.

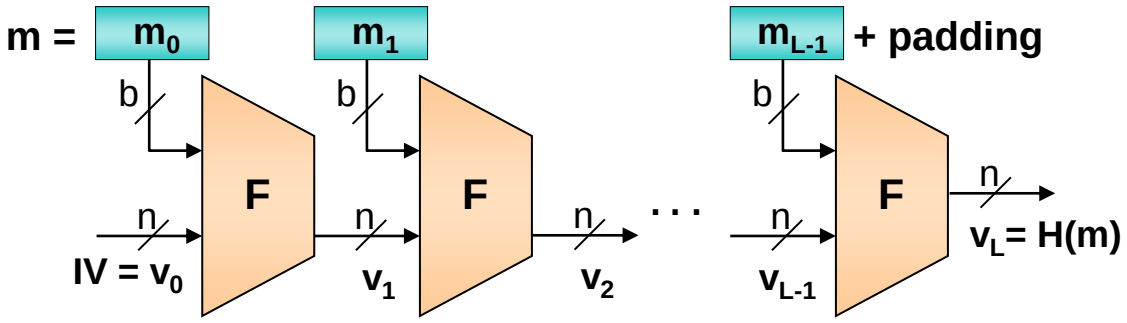
Cerințe de securitate

- Orice adversar (algoritm de atac) trebuie să necesite:
  - 2^n evaluări ale funcției H pentru preimagine sau a doua preimagine
  - $2^{n/2}$ evaluări ale funcției H pentru coliziune.
- Deci să nu existe un atac criptanalitic mai eficient decât prin forță brută.

Funcții hash iterative

Principiul construcției iterative

- O funcție hash cu intrări de lungime arbitrară este construită iterând o funcție de compresie cu intrări de lungime fixă, convenabilă.
- Funcția de compresie, $F : \{0,1\}^b \times \{0,1\}^n \rightarrow \{0,1\}^n$, are ca intrări un bloc de date de b biți și o variabilă de înlănțuire de n biți și o ieșire de n biți.



Preprocesare:
Completează mesajul:
a.î. $|m| \equiv 0 \bmod b$
Partiționează m în blocuri:
$m_0 \,||\, m_1 \,||\, \ldots \, m_{L-1}$, a. î. $|m_i| = b$

Calculul valorii hash:
v_0 = IV (fixat);
for $0 \le i \le t-1$
$v_{i+1} = F(v_i, m_i)$;
$H(m) = v_L$

Notații:
F = Funcție de compresie.
IV = Val. de inițializare fixă.
m_i = blocul de date i.
v_i = Variabilă de înlănțuire.

Prin înlănțuirea valorilor intermediare, H(m) depinde de toți biții mesajului m.

IV trebuie fixat pentru a evita coliziuni banale: H(m) = H(m'), eliminând m_0 din m și alegînd IV = v_1

Securitatea construcției iterative

Teorema Merkle-Damgård

- Considerăm o funcție de completare injectivă, $m \rightarrow P(m) = m \,||\, ext(m)$, care extinde mesajul m cu $ext(m)$ biți, astfel încât $|P(m)| \equiv 0 \bmod b$, și sunt îndeplinite proprietățile următoare:
  - Dacă $|m_1| = |m_2|$ atunci $ext(m_1) = ext(m_2)$, deci intrări cu lungime egală sunt extinse identic.
  - Altfel, dacă $|m_1| \neq |m_2|$, atunci ultimul bloc din $P(m_1)$ și din $P(m_2)$ sunt diferite (soluția uzuală: se înscrie lungimea mesajului).
- Dacă funcția de completare (padding) are proprietățile de mai sus și funcția de compresie F este rezistentă la coliziuni, atunci construcția iterativă este o funcție hash rezistentă la coliziuni.

- Construcția iterativă reduce problema proiectării unei funcții hash cu intrări de lungime arbitrară la problema proiectării funcției de compresie.
- Funcția de compresie trebuie să îndeplinească cerințele de securitate ale unei funcții hash criptografice.

Funcții hash: MD4 și MD5

Familia MD (Message Digest): Scurt istoric

- MD4: Algoritm propus de Rivest în 1990, b = 512 bits, n = 128 bits. Construcție iterativă. Optimizat pentru implementare software.
- MD5: Versiune îmbunătățită a MD4 (elimină vulnerabilități), propusă de Rivest în 1991. Cea mai utilizată funcție hash în anii '90 (foarte rapidă).
- Toți algoritmii propuși în 1992-2002 sunt construiți pe același principiu.

Atacuri criptanalitice asupra algoritmilor MD4 și MD5

- MD4, 1996 (†): Atac asupra rezistenței la coliziuni (ameliorat ulterior) cu complexitate 2^{22} operații (în loc de 2^{64}), câteva secunde pe PC.
- MD5, 2004 (†): Atac asupra rezistenței la coliziuni (ameliorat ulterior) cu complexitate 2^{34} operații (în loc de 2^{64}), câteva ore pe PC.
- MD5 nu mai poate fi utilizat în aplicații criptografice.

  2008: A fost demonstrat un atac care permite falsificarea certificatelor pentru chei publice semnate folosind MD5 și RSA. 2012: Malware-ul Flame folosea deja un certificat Microsoft falsificat printr-un atac criptanalitic diferit asupra MD5.

Funcții hash: SHA

Scurt istoric: de la SHA-1 la SHA-3

- SHA-1: Standardizat de NIST în 1995. Variantă ameliorată a MD5.
- SHA-2: Familie de algoritmi (SHA-256/384/256) standardizată de NIST în 2002, pentru a oferi un nivel de securitate comparabil cu cel oferit de AES-128/192/256. Același principiu ca SHA-1 și MD5.
- SHA-3: Nouă familie de algoritmi, standardizată de NIST în 2015.

	SHA-1	SHA-256	SHA-384	SHA-512
Size of hash value (n)	**160**	**256**	**384**	**512**
Message block size (b)	**512**	**512**	**1024**	**1024**
Number of rounds/steps	**80**	**64**	**80**	**80**
Best collision attack	**2^{66} (attack) < 2^{80}**	**2^{128}**	**2^{192}**	**2^{256}**

Atacuri criptanalitice asupra algoritmului SHA-1

- SHA-1, 2005 (†): Atac asupra rezistenței la coliziuni (ameliorat ulterior), cu complexitate 2^{66} operații (în loc de 2^{80}). SHA-1 nu mai poate fi utilizat în aplicații care necesită rezistență la coliziuni.

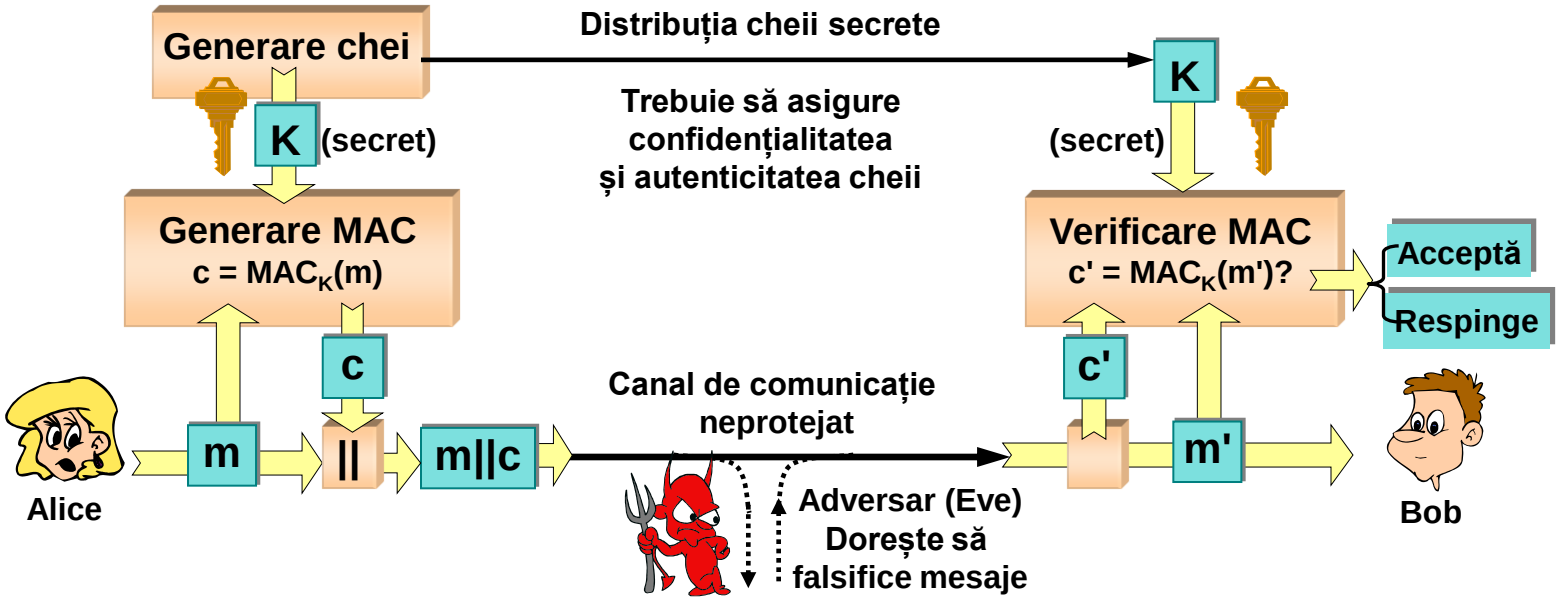
  Feb. 2017: A fost demonstrată o coliziune SHA-1 pentru 2 fișiere PDF.

Autentificarea datelor folosind criptografie cu cheie secretă

Coduri de autentificare a mesajelor

(MAC: Message Authentication Code)

Autentificarea datelor cu cheie secretă



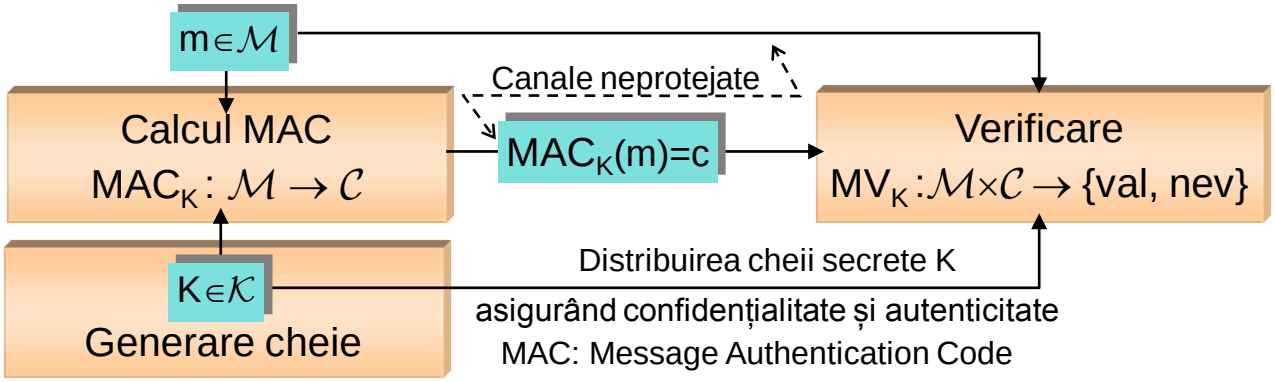
Cod de autentificare a mesajelor - Message Authentication Code

- Permite verificarea originii mesajului și (implicit) a integrității sale.
- Alice și Bob stabilesc o cheie secretă K asociată identităților lor.
- Alice generează codul de autentificare $\mathrm{c} = \mathrm{MAC}_{\mathrm{K}}(\mathrm{m})$ și transmite (m, c).
- Bob verifică mesajul și codul primit folosind propria copie a cheii K.
- Adversarul nu poate falsifica mesaje (fabrica, modifica) pentru că nu poate calcula codul de autentificare corect.

Definiții

Schemă de autentificare cu cheie secretă (MAC)

- O schemă MAC cu *mulțimea mesajelor* $\mathcal{M}=\{0,1\}^*$, *mulțimea codurilor* $\mathcal{C}=\{0,1\}^n$ și *mulțimea cheilor* $\mathcal{K}=\{0,1\}^k$, este alcătuită din:
- *Alg. de generare a cheilor*: Alege aleator uniform o cheie $K \in \mathcal{K}$.
- *Alg. de calcul al codului de autentificare*: Intrări: $m \in \mathcal{M}$ și $K \in \mathcal{K}$. Ieșire: codul de autentificare $c = MAC_K(m) \in \mathcal{C}$. Familie de funcții $MAC = \{ MAC_K \mid K \in \mathcal{K}, MAC_K : \mathcal{M} \to \mathcal{C} \}$.
- *Alg. de verificare*: Intrări: $m \in \mathcal{M}$, $c \in \mathcal{C}$ și $K \in \mathcal{K}$. Ieșire: rezultatul verificării, $MV_K(m, c) \in \{0,1\}$. Dacă $c = MAC_K(m)$ atunci $MV_K(m, c) = 1$ (valid), altfel $MV_K(m, c) = 0$ (nevalid).



Securitatea schemelor de autentificare

Cadrul general (similar celui pentru securitatea criptării)

- Model de securitate: Specifică obiectivele adversarului, modelele de atac și resursele adversarului.
- Cerință de securitate: Adversarul nu poate atinge un anumit obiectiv folosind un anumit model de atac și anumite resurse.
- Ipoteze generale despre adversar: Cunoaște algoritmii criptografici utilizați, nu cunoaște cheia secretă.

Obiectivele adversarului: falsificarea mesajelor

- Aplicațiile interpretează un mesaj în funcție de origine și conținut.
- Adversarul dorește să compromită o aplicație falsificând mesaje: să producă mesaje (creare sau modificare) și să le atribuie alte origini.

Modele de atac: capabilități suplimentare ale adversarului

- Cum poate interveni adversarul în funcționarea aplicației? Ce fel de informații poate să obțină privind protecția criptografică utilizată?

  Exemplu: poate obține perechi $(m, MAC_K(m))$, cu/fără a alege m.

Obiectivele adversarului

Falsificare selectivă

Adversarul poate genera o pereche $(m, MAC_K(m))$ validă pentru mesaje pe care le alege el însuși.

Falsificare existențială

Adversarul poate genera o pereche $(m, MAC_K(m))$ validă, dar nu poate controla conținutul mesajelor.

Falsificare universală

Adversarul poate genera codul de autentificare pentru oricare mesaj.

Recuperarea cheii

Permite falsificarea universală pe durata de viață a cheii.

falsificare universală > selectivă > existențială

Falsificare verificabilă / neverificabilă

O falsificare este verificabilă dacă adversarul poate să determine dacă perechea $(m, MAC_K(m))$ pe care a generat-o este validă.

falsificare verificabilă > neverificabilă

Modele de atac

Atac cu mesaje cunoscute (Known Message Attack (KMA))

- Adversarul poate să obțină perechi $(m_i, MAC_K(m_i))$, dar nu poate controla conținutul mesajelor obținute.
- Multe aplicații permit unui *adversar pasiv*, capabil doar să captureze trafic în rețea, să obțină un mare număr de perechi $(m_i, MAC_K(m_i))$.

Atac cu mesaje alese (Chosen Message Attack (CMA))

- Adversarul poate să obțină perechi $(m_i, MAC_K(m_i))$ pentru mesaje pe care le alege el însuși, adaptiv sau neadaptiv.
- Atacul poate fi realizat de un *adversar activ* care are acces la procesul care generează codul MAC printr-un protocol sau interfață.
- Posibilitatea de a alege mesaje permite adversarului să se concentreze asupra unor vulnerabilități cunoscute ale algoritmului MAC.

atac cu mesaje cunoscute < atac cu mesaje alese (< adaptiv)

Cerințe de securitate

Formularea cerințelor de securitate

- O cerință de securitate afirmă că adversarul nu poate atinge un anumit obiectiv folosind un anumit model de atac și anumite resurse.
- O cerință este mai puternică (strictă) dacă afirmă că adversarul nu poate atinge un obiectiv mai slab folosind un model de securitate mai puternic.

Cerința de securitate pentru autentificarea datelor folosind MAC

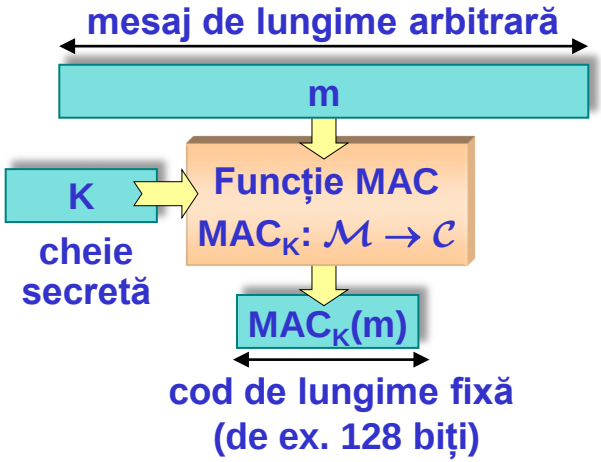
- UF-CMA ((Existential) Unforgeability under Chosen-Message Attack): Nefalsificarea existențială a mesajelor în atacuri cu mesaj ales adaptive.
- Este cerința standard de securitate și cea mai strictă: obiectivul minim (falsificare existențială) nu poate fi atins prin cel mai puternic atac.

Altă perspectivă: Algoritmul MAC este o funcție pseudoaleatoare

- O schemă de autentificare $\mathrm{MAC} : \{0,1\}^k \times \{0,1\}^* \to \{0,1\}^n$ trebuie să se comporte ca o *funcție pseudoaleatoare*: pentru o alegere uniform aleatoare a cheii $K \in \{0,1\}^k$, valorile sale de ieșire, $\mathrm{MAC}_K(m)$, nu pot fi distinse de cele ale funcției aleatoare $F : \{0,1\}^* \to \{0,1\}^n$.

Securitatea schemelor MAC (1)

- Considerăm o schemă de autentificare a mesajelor MAC : $\mathcal{K} \times \mathcal{M} \to \mathcal{C}$, unde $\mathcal{K} = \{0,1\}^k$, $\mathcal{M} = \{0,1\}^*$ și $\mathcal{C} = \{0,1\}^n$, care se comportă ca o *funcție pseudoaleatoare*.
- Parametrii de securitate sunt k și n (biți).
- Complexitatea atacurilor prin forță brută indică nivelul maxim de securitate care poate fi atins.



Atacuri aleatoare asupra mesajelor

- Pentru a falsifica un mesaj m, adversarul poate încerca să ghicească:
  - codul $c = \mathrm{MAC}_K(m) \in \{0,1\}^n$, cu probabilitate 2^{-n}, sau
  - cheia $K \in \{0,1\}^k$, cu probabilitate 2^{-k}, calculând apoi $c = \mathrm{MAC}_K(m)$.
- Pentru un cod $c = \mathrm{MAC}_K(m) \in \{0,1\}^n$, adversarul poate încerca să ghicească m cu probabilitate 2^{-n}.
- Pentru a verifica astfel de falsuri adversarul trebuie să interacționeze cu victima și va reuși un fals după $\min(2^n, 2^k)$ evaluări MAC.

Securitatea schemelor MAC (2)

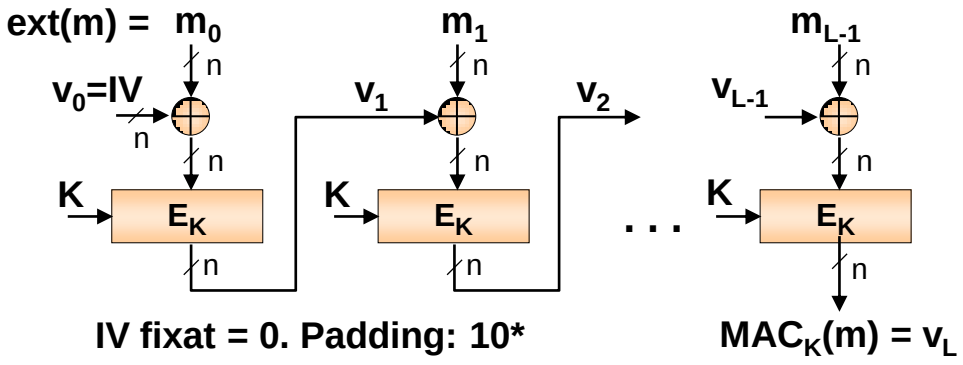
Căutarea exhaustivă a cheii (exhaustive key search, total break)

- Presupunem că am obținut mai multe perechi $(m_i, MAC_X(m_i))$ arbitrare.
- Algoritmul de căutare se execută *offline* și se pot aloca resurse importante (calcul paralel, hardware dedicat).
- Inițial, selectăm cheile $K \in \mathcal{K}$ cu proprietatea că $MAC_K(m_1) = MAC_X(m_1)$. Obținem aprox. $1 + 2^{k-n}$ chei posibile.
- Repetăm pentru setul de chei rămase folosind $(m_i, MAC_X(m_i))$, $i > 1$, până când setul de chei posibile se reduce la o singură cheie.
- În total: $O(2^k)$ evaluări MAC și aprox. $1 + k/n$ perechi $(m_i, MAC_X(m_i))$.

Cerințe de securitate cantitative

- O schemă de autentificare trebuie proiectată astfel încât nici un atac criptanalitic să nu fie mai eficient decât atacurile prin forță brută.
- Pentru o schemă MAC cu cheie de k biți și cod de n biți, complexitatea celui mai eficient atac ar trebui să fie $O(\min(2^k, 2^n))$.

MAC cu cifru bloc: CBC-MAC



CBC-MAC:
Extinde m a.î. |ext(m)| mod n = 0;
Împarte ext(m) în blocuri:
ext(m) = m_0, m_1 ... m_{L-1};
v_0 = IV (de obicei IV = 0);
For $0 \le i \le L-1$ $v_{i+1} = E_K(m_i \oplus v_i)$;
$MAC_K(m) = v_L$

- CBC (Cipher-Block Chaining) MAC: Algoritm similar cu criptarea CBC, dar reținem doar ultimul bloc, care reprezintă codul MAC. Prin înlănțuire, codul MAC obținut depinde de toți biții mesajului.
- Construcție iterativă: Un algoritm MAC cu intrări de lungime arbitrară este obținut iterând o *funcție de compresie* cu intrări de lungime fixă. Similar cu o funcție hash iterativă, dar adăugăm o cheie.
- În algoritmul CBC-MAC, funcția de compresie este realizată cu un cifru bloc: $F : \{0,1\}^k \times \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$, $F_K(v, m) = E_K(v \oplus m)$.

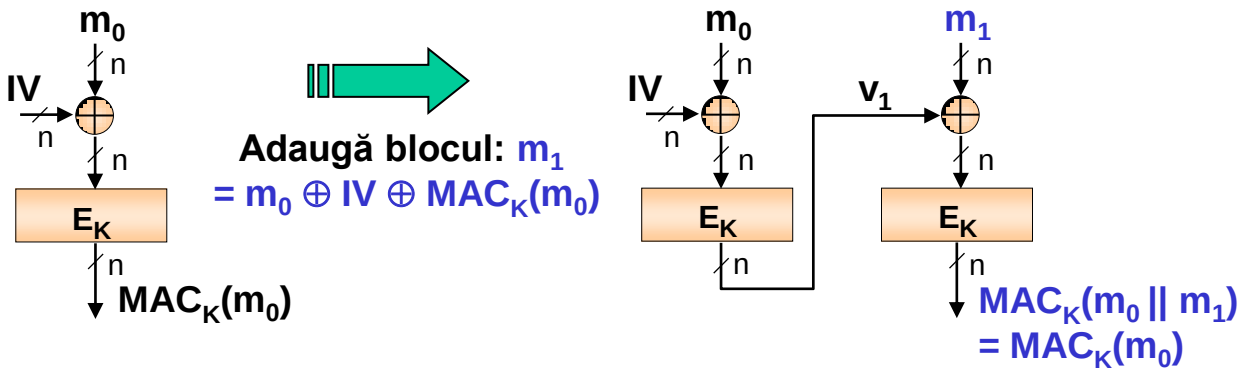
- Valoarea de inițializare (IV) trebuie fixată pentru a evita falsificări banale. Altfel, am putea înlocui m_0 cu m'_0 înlocuind IV cu $IV' = IV \oplus m_0 \oplus m'_0$.

Securitatea schemei CBC-MAC

- **Teoremă:** CBC-MAC îndeplinește cerința UF-CMA pentru *mesaje de lungime fixă* dacă cifrul bloc este o *permutare pseudoaleatoare*.

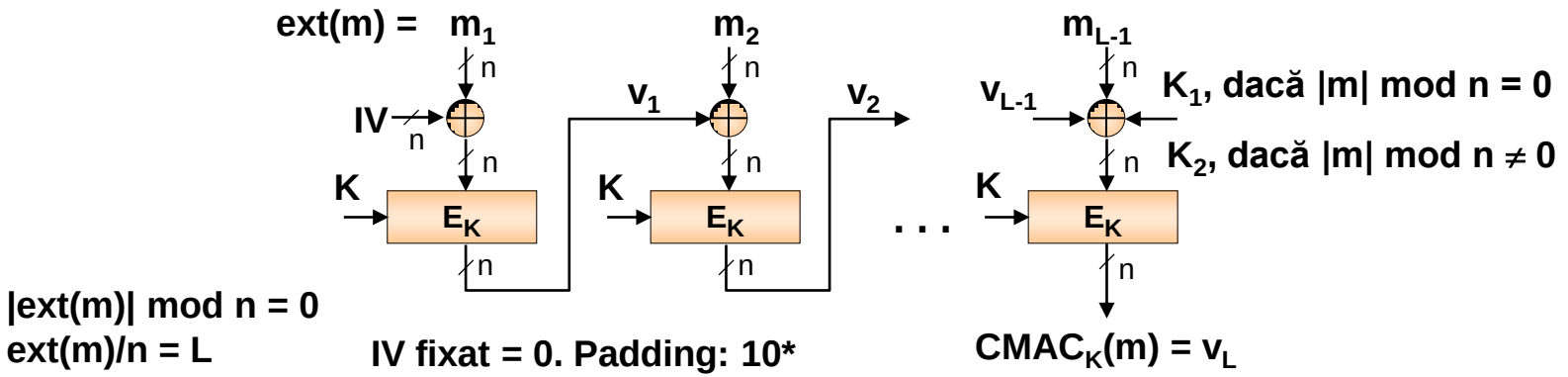
- CBC-MAC nu este o schemă MAC sigură pentru mesaje de lungime arbitrară. Este vulnerabilă la falsificări prin extinderea mesajului.

CBC-MAC: Exemplu de falsificare a mesajelor de lungime variabilă (XOR forgery)



- Putem obține o schmeă MAC sigură pentru mesaje de lungime variabilă adăugând o transformare de ieșire care să blocheze înlănțuirea.
- Exemplu: Criptăm codul CBC-MAC folosind cifrul bloc cu altă cheie, K', derivată din K: $\mathrm{MAC}_K(m) = E_{K'}(\mathrm{CBCMAC}_K(m))$. Există soluții mai eficiente.

CMAC (Cipher-based MAC)

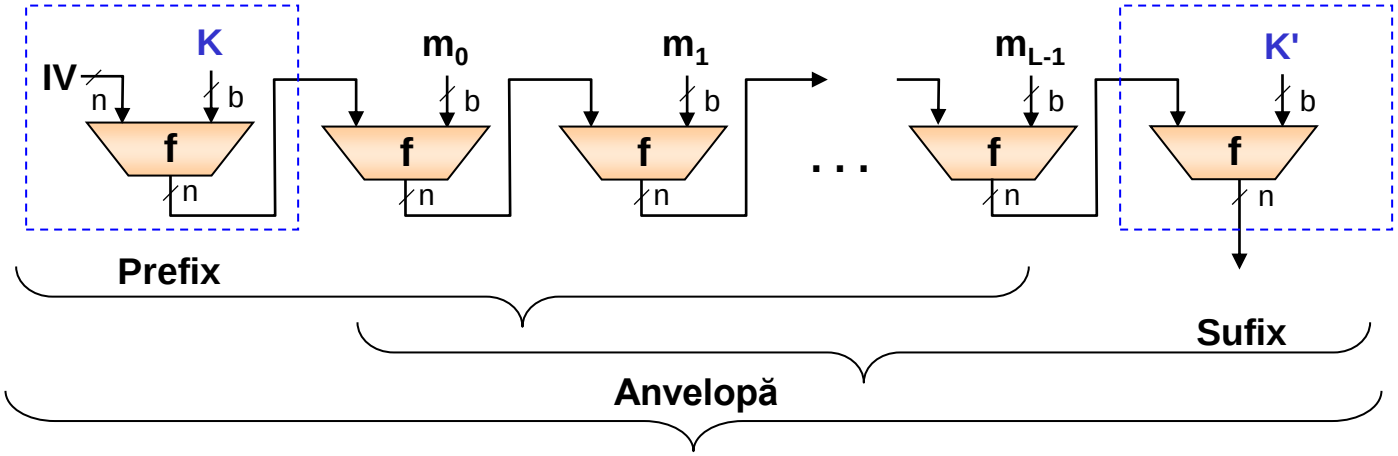


- Standard NIST (SP 800-38B), etc. Variantă CBC-MAC care îndeplinește în mod *eficient* cerința UF-CMA pentru *mesaje de lungime variabilă*.

- Dacă $|m| \bmod n \neq 0$, atunci m este extins cu un minim de biți, folosind o funcție injectivă ext, astfel încât $|ext(m)| \bmod n = 0$ (padding).
- Transformarea de ieșire este integrată în ultima iterație (blocul m_{L-1}) și folosește cheile K_1 și K_2 derivate din K. Sunt necesare 2 chei pentru a evita falsuri banale: m cu/fără extindere să aibă coduri MAC diferite.
  - Dacă $|m| \bmod n = 0$ atunci $v_L = E_K(v_{L-1} \oplus m_{L-1} \oplus K_1)$.
  - Altfel $v_L = E_K(v_{L-1} \oplus m_{L-1} \oplus K_2)$.

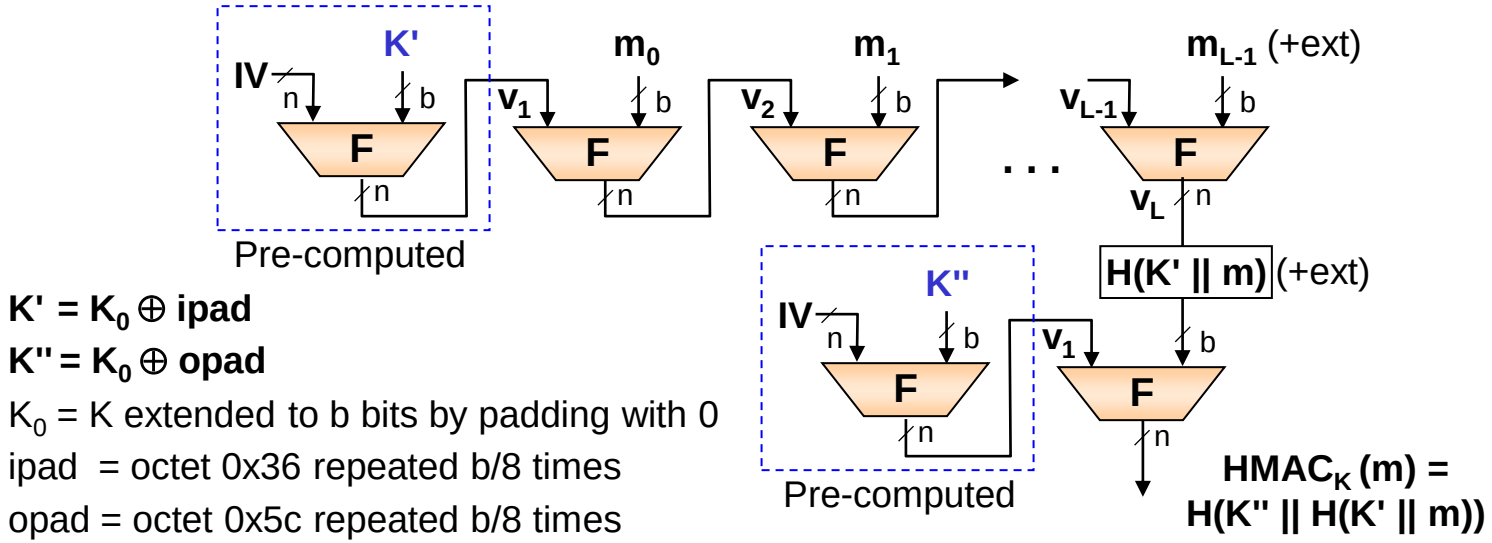
Construcții MAC cu funcții hash

- Putem construi un algoritm MAC integrând o cheie secretă în algoritmul unei funcții hash criptografice? Cum adăugăm cheia secretă?



- **Prefix secret**: Cheia secretă înaintea mesajului: $\mathrm{MAC_K(m) = H(K \,||\, m)}$. Algoritm vulnerabil la atacuri banale prin extinderea mesajului.
- **Sufix secret**: Cheia secretă după mesaj: $\mathrm{MAC_K(m) = H(m \,||\, K)}$. Algoritm expus atacurilor asupra rezistenței la coliziuni a funcției hash: $O(2^{n/2})$.
- **Anvelopă**: Chei secrete înainte și după mesaj: $\mathrm{MAC_K(m) = H(K \,||\, m \,||\, K')}$. Evită deficiențele construcțiilor precedente. Există soluții mai eficiente.

HMAC (Hash-based MAC)



K' = $K_0 \oplus$ ipad

K'' = $K_0 \oplus$ opad

K_0 = K extended to b bits by padding with 0

ipad = octet 0x36 repeated b/8 times

opad = octet 0x5c repeated b/8 times

- Standard NIST (FIPS 198), ISO 9797-2, ANSI X9.71, RFC 2104.
- Variantă optimizată a metodei anvelopei, cu securitate demonstrată.

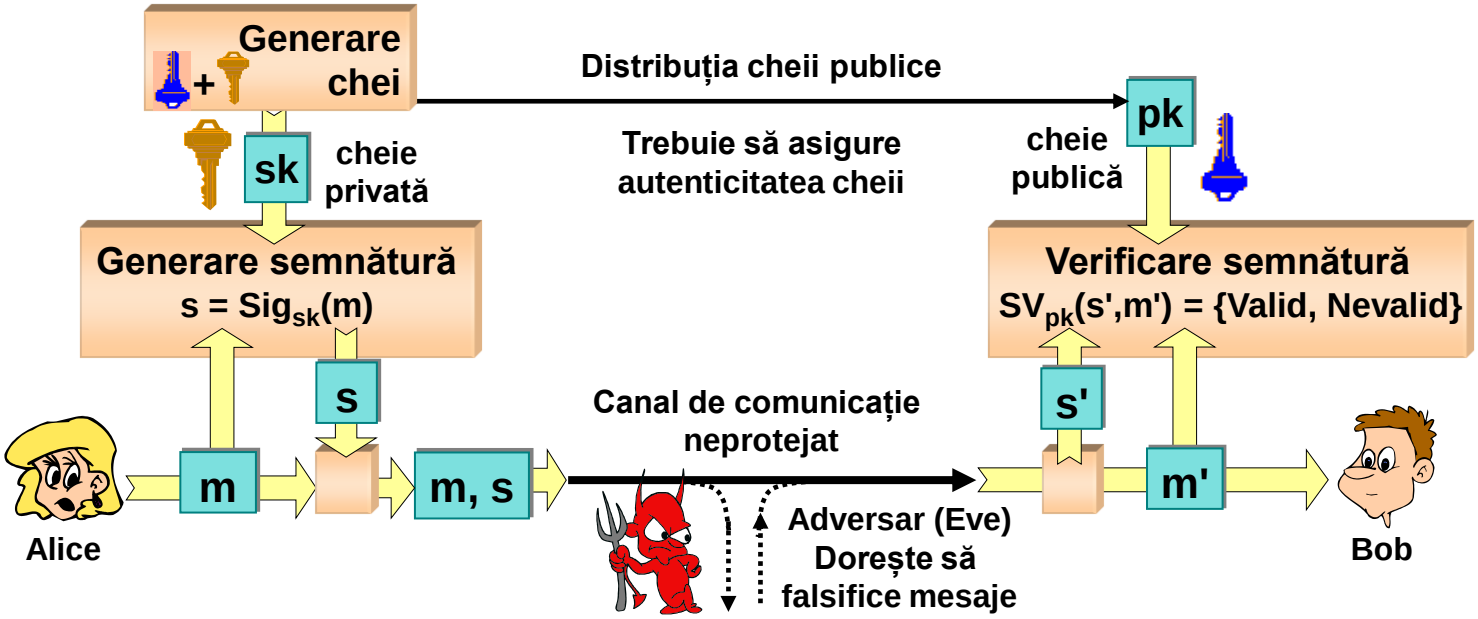
- Teoremă: HMAC îndeplinește UF-CMA dacă funcția H, cu o valoare aleatoare și secretă pe intrarea IV, este o funcție hash rezistentă la coliziuni, iar funcția F, cu valoare aleatoare și secretă ca intrare de înlănțuire, este un algoritm MAC UF-CMA cu mesaje de lungime fixă.
- Eficiență: Valorile $F(IV, K')$ și $F(IV, K'')$ pot fi precalculate.

Autentificarea datelor folosind criptografie cu cheie publică

Semnătura digitală

(digital signature)

Autentificarea datelor cu cheie publică

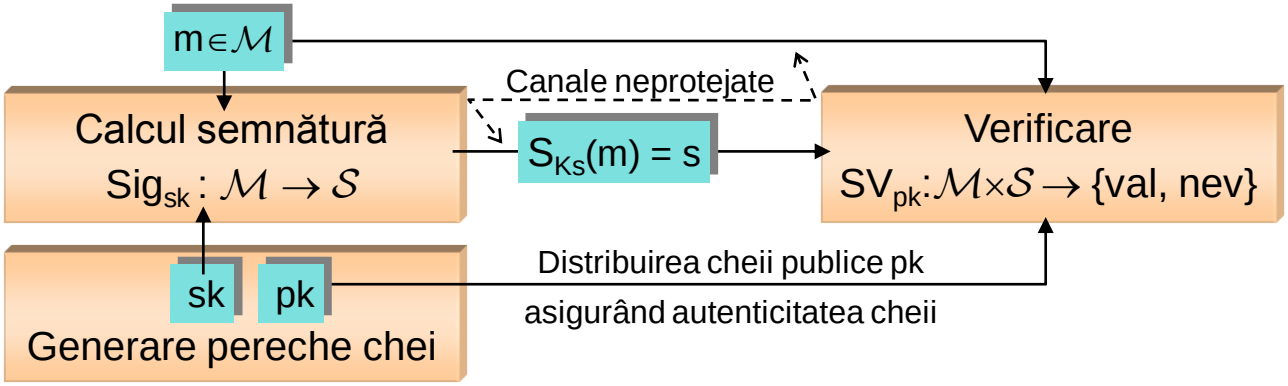


- Permite verificarea originii și integrității mesajului. În plus, *nerepudiere*.
- Alice generează o *pereche de chei*, *cheia privată*, sk, și *cheia publică*, pk, și distribuie pk partenerilor de comunicație.
- Alice generează *semnătura digitală* $\mathrm{s} = \mathrm{Sig}_{\mathrm{sk}}(\mathrm{m})$ și transmite (m, s).
- Bob verifică mesajul și semnătura cu cheia publică a lui Alice.
- Adversarul nu poate falsifica mesaje (fabrica, modifica) pentru că nu poate calcula semnătura corectă (doar Alice poate calcula semnătura).

Definiții

Schemă de autentificare cu cheie publică (semnătură digitală)

- O schemă de semnătură digitală cu *mulțimea mesajelor* $\mathcal{M} = \{0,1\}^*$, *mulțimea semnăturilor* $\mathcal{S}$ și *mulțimea cheilor* $\mathcal{K}$, este alcătuită din:
- *Alg. de generare a cheilor*: Intrare: parametru de securitate k. Ieșire: pereche $(sk, pk) \in \mathcal{K}$ aleasă aleator (cheia privată sk, cheia publică pk).
- *Alg. de semnare*: Intrări: $m \in \mathcal{M}$ și sk. Ieșire: semnătura digitală $s = Sig_{sk}(m) \in \mathcal{S}$. Familie de funcții $\{ Sig_{sk} \mid (sk, pk) \in \mathcal{K}, Sig_{sk} : \mathcal{M} \to \mathcal{S} \}$.
- *Alg. de verificare*: Intrări: $m \in \mathcal{M}$, $s \in \mathcal{S}$ și pk. Ieșire: rezultatul verificării, $SV_{pk}(m, s) \in \{0,1\}$. Dacă $s = Sig_{sk}(m)$ atunci $SV_{pk}(m, s) = 1$ (valid), altfel $SV_{pk}(m, s) = 0$ (nevalid). Familie $\{SV_{pk} \mid (sk, pk) \in \mathcal{K}, SV_{pk} : \mathcal{M} \times \mathcal{S} \to \{0,1\}\}$.



Securitatea schemelor de semnătură (1)

Model de securitate pentru autentificare cu cheie publică

- Adaptăm modelul general pentru scheme de autentificare.
- Aspect specific: Semnare cu cheie secretă, verificare cu cheie publică.

Obiectivele adversarului

- Adversarul dorește să compromită o aplicație falsificând mesaje.

Falsificare selectivă

Adversarul poate genera o pereche $(s, \mathrm{Sig}_{sk}(m))$ validă pentru mesaje pe care le alege el însuși.

Falsificare existențială

Adversarul poate genera o pereche $(s, \mathrm{Sig}_{sk}(m))$ validă, dar nu poate controla conținutul mesajelor.

Falsificare universală

Adversarul poate genera semnătura pentru oricare mesaj.

Recuperarea cheii private

Permite falsificarea universală pe durata de viață a cheii.

falsificare universală > selectivă > existențială

Securitatea schemelor de semnătură (2)

Modele generale de atac

- Modele aplicabile oricărei scheme de autentificare (similar MAC): atac cu mesaje cunoscute (KMA), atac cu mesaje alese (CMA), side-channel.
- Aspect specific: Cunoscând cheia publică, adversarul poate verifica eficient, offline, orice încercare de falsificare.

atac cu mesaje cunoscute < atac cu mesaje alese (< adaptiv)

Atacuri specifice pentru scheme cu cheie publică

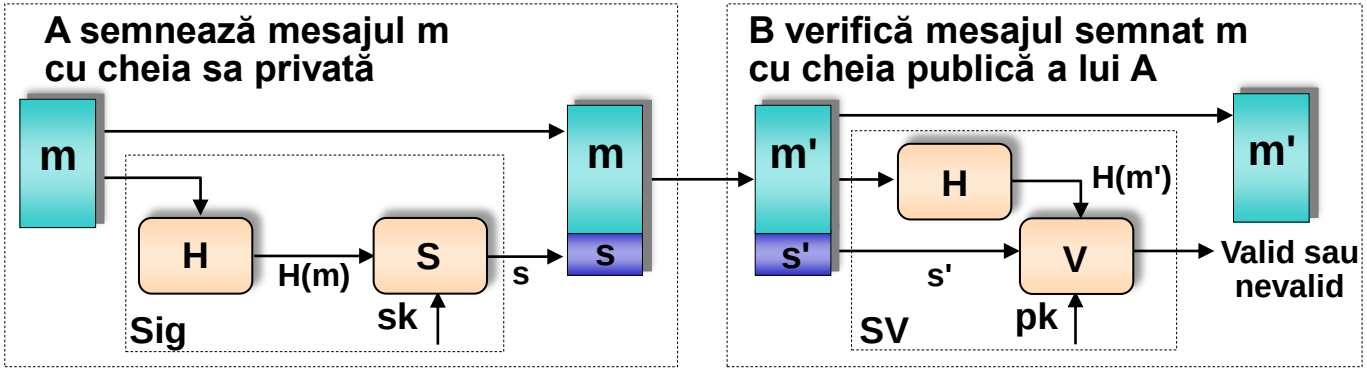
- Atacuri bazate pe cunoașterea cheii publice: recuperarea cheii private sau falsificare individuală a mesajelor.

Cerința de securitate pentru autentificarea datelor cu semnătură

- UF-CMA ((Existential) Unforgeability under Chosen-Message Attack): Nefalsificarea existențială a mesajelor în atacuri cu mesaj ales adaptive.
- Este cerința standard de securitate și cea mai strictă: obiectivul minim (falsificare existențială) nu poate fi atins prin cel mai puternic atac.

Construcția "hash and sign"

- Avem nevoie (de obicei) de o schemă de semnătură digitală pentru mesaje de lungime arbitrară. Construcția uzuală (scheme standard):
  - Comprimă mesajul folosind o funcție hash criptografică H: Pentru oricare $m \in \{0,1\}^*$ se obține un șir de lungime fixă $H(m) \in \{0,1\}^n$.
  - Calculează semnătura folosind o permutare pseudoaleatoare S: $s = S_{sk}(H(m))$. Adversarul nu poate calcula s pentru că nu cunoaște S_{sk}.



- Exemplu: S_{sk} ar putea fi inversa unei permutări cu sens unic și trapă V_{pk}.
  - Alg. de calcul al semnăturii: $h = H(m)$; $s = S_{sk}(H(m)) = V^{-1}_{pk}(H(m))$.
  - Alg. de verificare: $h' = V_{pk}(s')$. Dacă $h' = H(m')$ atunci valid, altfel nevalid.

Securitatea construcției "hash and sign"

Analiza construcției hash and sign

- Dacă S este o permutare pseudoaleatoare și S_{sk} este secretă, adversarul nu poate determina $s = S_{sk}(H(m))$ (decât cu probabilitate neglijabilă).
- Construcția este însă vulnerabilă la atacuri asupra funcției hash: Mesaje cu aceeași valoare hash au aceeași semnătură: dacă $H(m) = H(m')$ atunci $S_{sk}(H(m)) = S_{sk}(H(m'))$.

Falsificare prin atac asupra rezistenței la a doua preimagine

- Fiind dat $(m, s = S_{sk}(H(m))$ găsește m' a.î. $H(m') = H(m)$. Fals: (m', s).
- Permite falsificarea unui mesaj m ***după semnare***. Pentru H cu ieșire de n biți, complexitatea atacului (forță brută) este $O(2^n)$.

Falsificare prin atac asupra rezistenței la coliziune

- Găsește m, m' astfel încât $H(m) = H(m')$. Află $s = S_{sk}(H(m)$. Fals: (m', s).
- Permite falsificarea unui mesaj m ***înainte de semnare***. Pentru H cu ieșire de n biți, complexitatea atacului (forță brută) este $O(2^{n/2})$.

Falsificări prin atac asupra funcției hash

- Un document *deja semnat* poate fi falsificat doar căutând *a doua preimagine*, ceea ce necesită un efort $O(2^n)$.
- Falsificarea prin *căutarea coliziunilor* este mult mai eficientă, fiind suficient efort $O(2^{n/2})$, dar poate fi aplicată doar *înainte de semnare* și necesită *acces la funcția de semnare*.
- În prezent, pentru un nivel de securitate de 128 biți, algoritmul de semnare trebuie să folosească o funcție hash cu n = 256 biți.
- Atacurile criptanalitice vizează rezistența la coliziune.

Exemple

- Semnatar neonest: Determină m, m' a.î. $H(m') = H(m)$, calculează $s = S_{sk}(H(m))$ și trimite (m, s). Ulterior, pretinde că a trimis (m', s).
- Verificator neonest: Determină m, m' a.î. $H(m') = H(m)$. Trimite m pentru semnare și obține $s = S_{sk}(H(m))$. Ulterior, pretinde că a trimis (m', s).
- În practică: Au fost demonstrate falsificări selective pentru certificate semnate folosind MD5 și documente PDF semnate folosind SHA-1.

Autentificarea datelor folosind criptografie cu cheie publică

Semnături digitale folosind RSA

Semnătura RSA-FDH

- Alg. de generare a perechii de chei: Parametru de securitate $k = 2\ell$.
  - Alege aleator și independent numerele prime $p, q \in [2^{\ell-1}, 2^{\ell}-1]$ astfel încât $p \neq q$ și $|pq| = k$ (biți).
  - Calculează $n = pq$ și $\varphi(n) = (p-1)(q-1)$. Alege $e \in [2, \varphi(n)-1]$ aleator, a.î. $\gcd(e, \varphi(n)) = 1$. Calculează $d = e^{-1} \bmod \varphi(n)$.
  - Cheia privată (semnare): $sk = \langle n, d \rangle$ sau $sk = \langle p, q, d \rangle$.
  - Cheia publică (verificare): $pk = \langle n, e \rangle$.
- Presupunem o funcție hash: $H : \{0,1\}^* \to Z_n$ (full-domain hash: FDH).
- Alg. de semnare: $Sig_{sk} : \{0,1\}^* \to Z_n$. $Sig_{sk}(m) = H(m)^d \bmod n$.
- Alg. de verificare: $SV_{pk} : Z_n \to \{0,1\}$. Verificare (m,s): Dacă $s^e \bmod n = H(m)$ atunci $SV_{pk}(m, s) = 1$ (valid), altfel $SV_{pk}(m, s) = 0$ (nevalid).

- Condiția de corectitudine: $s = H(m)^d \bmod n = RSA_{n,d}(H(m))$ și $s^e \bmod n = RSA_{n,e}(s) = RSA_{n,e}(RSA_{n,d}(H(m))) = H(m)$. Prin urmare, pentru oricare m și pentru $s = Sig_{sk}(m)$ verificarea reușește, $SV_{pk}(m, s) = 1$.

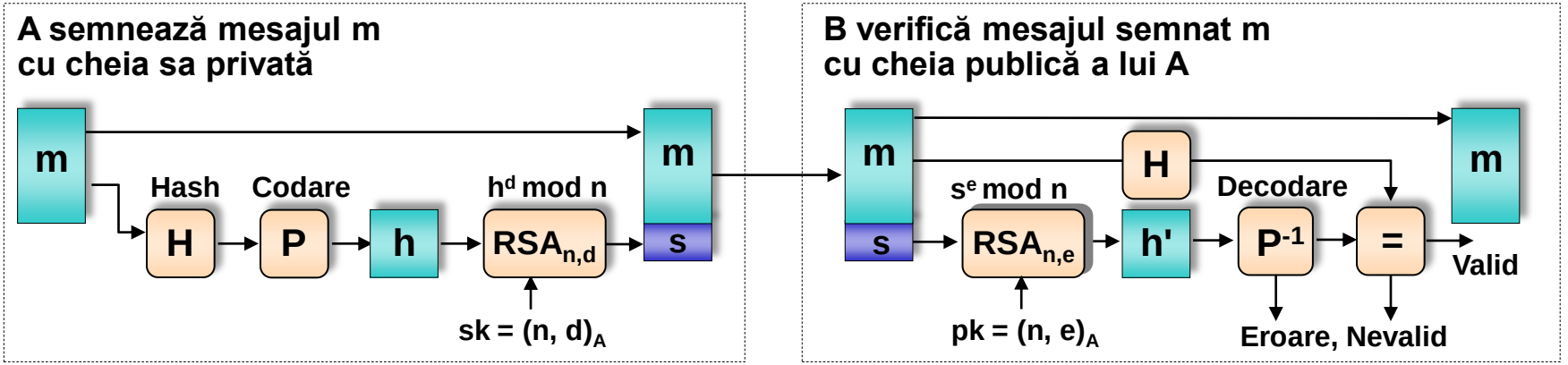
Securitatea semnăturii RSA

Securitatea schemei RSA-FDH

- RSA-FDH îndeplinește UF-CMA dacă $H : \{0,1\}^* \to Z_n$ este o funcție hash rezistentă la coliziuni și ipoteza RSA este validă ($RSA_{n,e}$ nu poate fi inversată într-un punct aleator din domeniul său).

Scheme de semnătură RSA cu codare

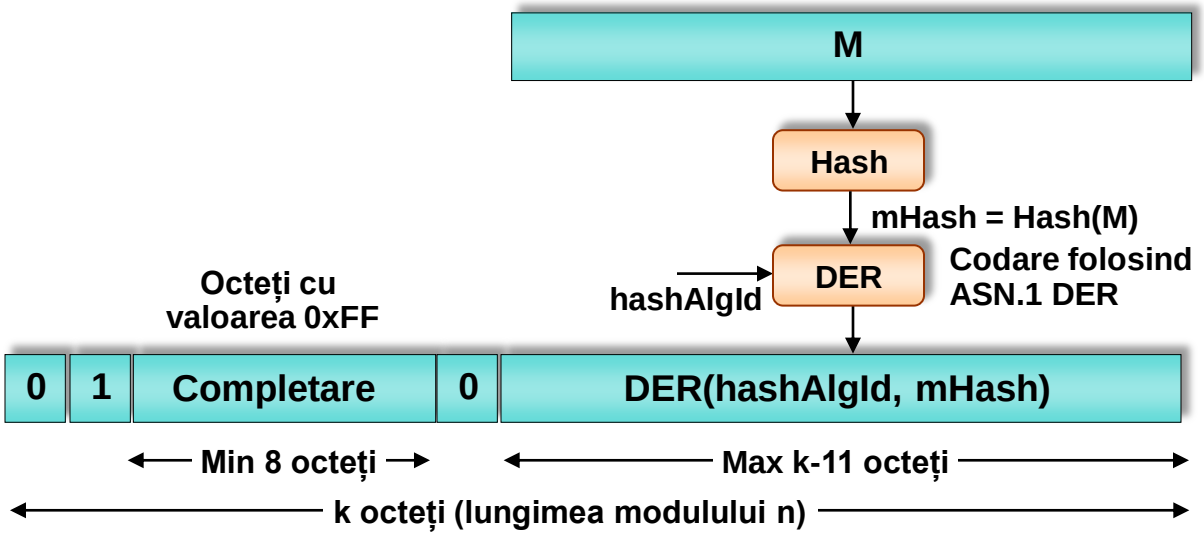
- RSA-FDH nu poate fi implementată cu algoritmii uzuali pentru funcții hash, deoarece au ieșiri de lungime fixă, predeterminată (ex: 256 biți).
- Avem nevoie de o funcție de codare inversabilă care să transforme $H(m)$ într-un element (pseudo)aleator din Z_n.



Codare PKCS #1 v1.5

O primă soluție: Semnătură RSA cu codare PKCS #1 v1.5 (1993)

- Schema de semnătură RSA descrisă în PKCS #1 v1.5.
- Blocul de date este completat până la k octeți cu un șir de octeți cu valorea 0xFF terminat cu un octet egal cu zero.
- Algoritmul de semnare este determinist.

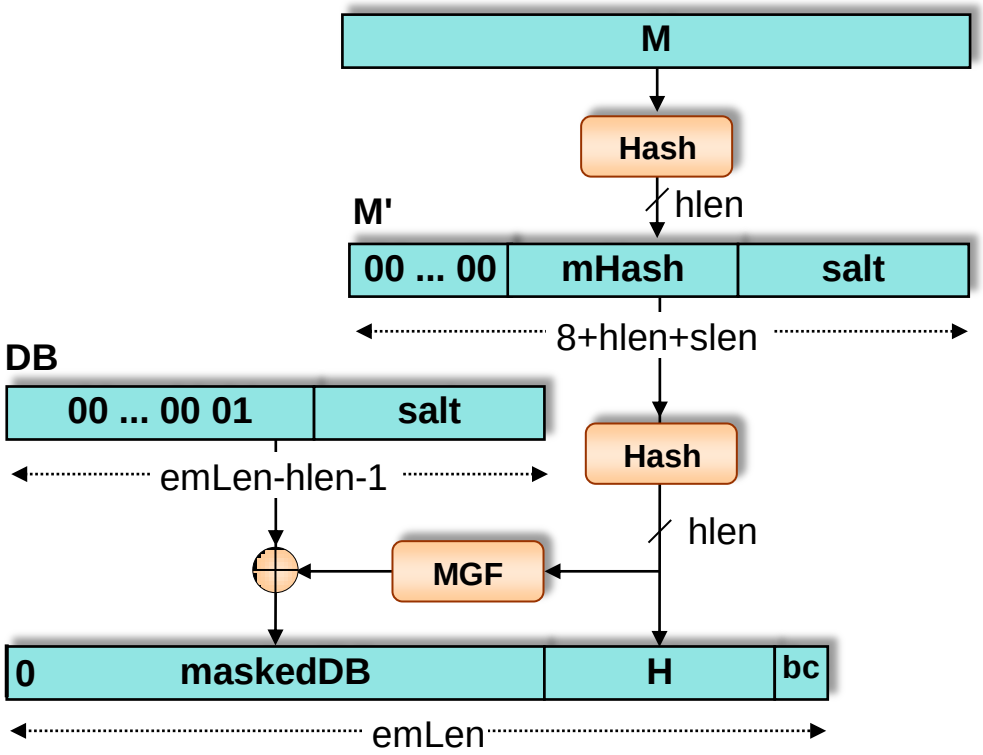


- Nu există o demonstrație de securitate pentru această schemă.

Codare PKCS #1 v2 (PSS)

Semnătură RSA-PSS (PKCS #1 v2, 2000)

- Variantă a schemei nedeterministe PSS (Probabilistic Signature Scheme) propusă de Bellare și Rogaway.
- Se poate demonstra că RSA-PSS îndeplinește UF-CMA, presupunând că ipoteza RSA este validă și funcția hash este rezistentă la coliziuni.



Hash = Funcție hash criptografică cu ieșire de *hlen* octeți. *mHash* = Hash(M).

***salt* = Șir de octeți pseudoaleator de lungime *slen* (hlen sau 0 octeți). Dacă slen = 0 schema devine deterministă.**

MGF = Mask Generation Function. Funcție psudoaleatoare construită folosind funcția hash.

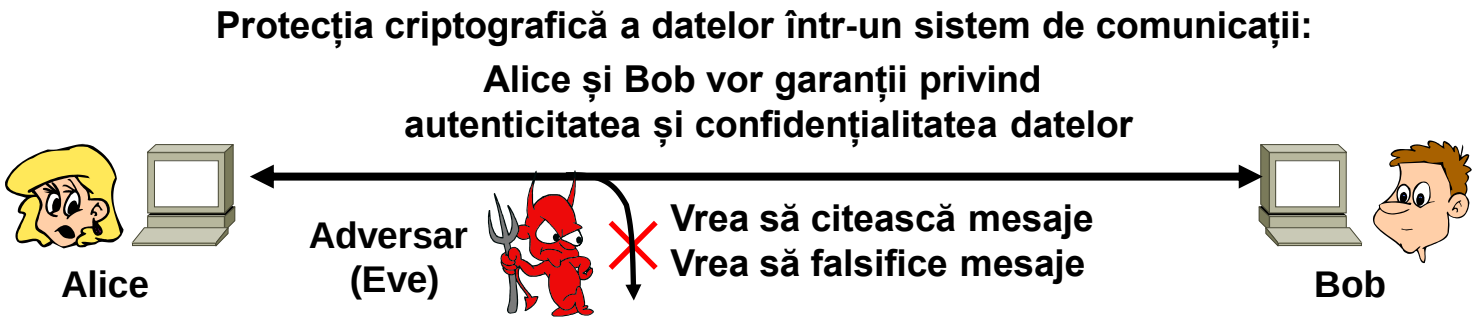
Primul octat are valoarea 0, ultimul octet are valoarea 0xbc.

Decodare: Se recuperează șirul *salt* din semnătură, se recalculează H și se verifică dacă valorile sunt egale.

Autentificarea și criptarea datelor folosind criptografie cu cheie secretă

Criptare autentificată

Criptare autentificată



Autentificarea datelor este necesară tuturor aplicațiilor

- Criptarea nu poate asigura autentificarea datelor. Avem nevoie de scheme care oferă atât confidențialitate, cât și autentificare.
- Schemele de criptare simetrice prezentate oferă doar securitate IND-CPA. Adăugând autentificarea, putem obține criptare IND-CCA.

Soluții pentru criptare autentificată

- Compunerea unei scheme de criptare IND-CPA cu o schemă de autentificare UF-CMA.
- Scheme dedicate, special proiectate și optimizate pentru criptare autentificată. Ex: CCM, GCM.

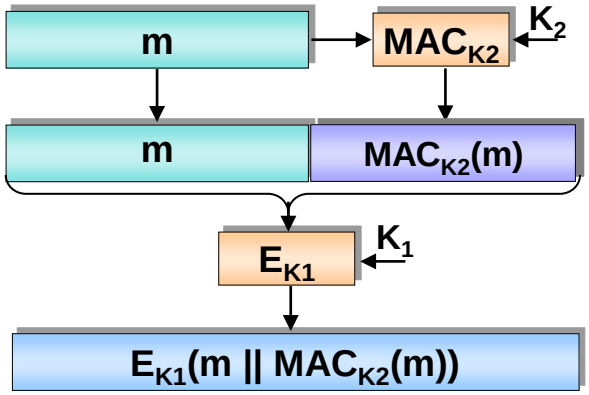
Compunere generică (1)

Obiectiv

- Schemă de criptare autentificată care oferă UF-CMA și IND-CCA prin compunerea oricărei scheme de criptare IND-CPA cu oricare schemă de autentificare UF-CMA. Cum combinăm aceste scheme?

Autentificare urmată de criptare (MAC then Encrypt (MtE))

- Metodă utilizată în protocolul SSL (>1996) și apoi în protocolul TLS.
- În prezent: înlocuită treptat în TLS de scheme de criptare autentificată dedicate (ex: GCM).

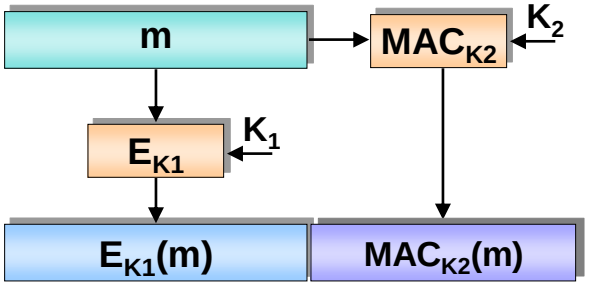


- Securitate: MtE îndeplinește cerințele de securitate pentru anumite scheme de criptare și autentificare, dar *nu în general*.
- *Nu este recomandată* ca metodă de compunere generică.

Compunere generică (2)

Criptare și autentificare (Encrypt and MAC (EaM))

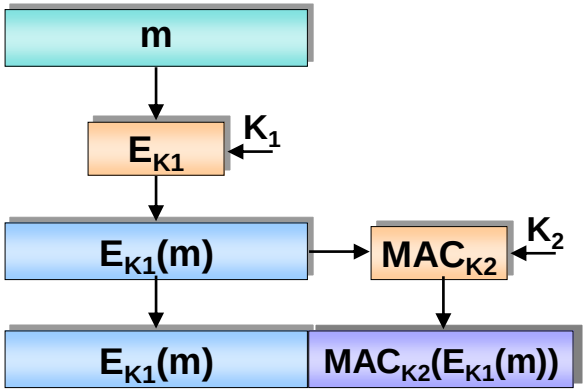
- Metoda cea mai evidentă, utilizată în protocolul SSH (Secure Shell).



- Securitate: EaM îndeplinește cerințele de securitate doar pentru anumite scheme de criptare și autentificare. *Nu este recomandată*.

Criptare urmată de autentificare (Encrypt then MAC (EtM))

- Metodă utilizată în suita IPsec (în particular, în Encapsulated Security Payload (ESP)).



- Securitate: EtM îndeplinește cerințele de securitate pentru orice scheme de criptare și de autentificare. *Este metoda de compunere recomandată*.