CURS 11
SEMNĂTURI DIGITALE

Semnătură digitală = un șir binar de lungime fixă (de obicei, 128, 256, 512 sau 1024 de biți) atașat unui document electronic care depinde de document și de cheia privată a semnatarului

Caracteristicile unei semnături digitale:
- să fie unică pentru un anumit document și un anumit semnatar
- să fie inimitabilă sau, cel puțin, foarte greu imitabilă
- să fie ușor de generat
- să fie ușor de verificat

Semnătură olografă = NU depinde de document, ci trebuie să fie aceeași indiferent de documentul semnat

Un algoritm de semnătură digitală constă în:
- *algoritmul de generare a cheilor* (privată și publică)
- *algoritmul de semnare*, care utilizează documentul și cheia privată a semnatarului și generează o semnătură (sub forma unui șir binar)
- *algoritmul de verificare*, care utilizează documentul, semnătura documentului și cheia publică a semnatarului, după care generează unul dintre mesajele DA/true sau NU/false.

Observație: NU se utilizează aceeași pereche de chei pentru criptare și semnare!!!

- Presupunem că utilizatorul B folosește aceeași pereche de chei $(K_{priv}^B, K_{pub}^B, n)$ pentru criptare și semnare
- Un utilizator H interceptează (neautorizat!!!) un mesaj clar M criptat într-un mesaj C pe care un utilizator A i-l transmite utilizatorului B => $C = M^{K_{pub}^B} (\mathrm{mod}\ n)$
- Utilizatorul H îi cere utilizatorului B să semneze mesajul criptat C, iar dacă utilizatorul B acceptă, atunci acesta îi va transmite lui H perechea (C, S), unde $S = C^{K_{priv}^B} (\mathrm{mod}\ n) = \left(\underbrace{M^{K_{pub}^B}}_{C} \right)^{K_{priv}^B} (\mathrm{mod}\ n) = M$, deci utilizatorul H va primi mesajul clar M (pe care H nu avea dreptul să-l acceseze)!!!

Criptare+semnare vs. Semnare+criptare (utilizatorul A -> utilizatorul B)

- **Criptare + semnare:**
  - utilizatorul A criptează un mesaj clar M și obține mesajul criptat $C = M^{K_{pub}^{B}} (\mathrm{mod}\ n_B)$
  - utilizatorul A semnează mesajul criptat C și obține semnătura $S = C^{K_{priv}^{A}} (\mathrm{mod}\ n_A)$
  - utilizatorul A transmite utilizatorului B perechea (C, S)
  - un utilizator H interceptează (neautorizat!!!) perechea (C, S) de la utilizatorul A și oprește transmiterea sa către utilizatorul B
  - utilizatorul H elimină semnătura utilizatorului A și o înlocuiește cu semnătura sa S', după care îi trimite utilizatorului B perechea (C, S') din partea sa => *furt de identitate* (de exemplu, o idee genială a utilizatorului A va fi transmisă șefului B de către "hoțul" H ca fiind a sa)!!!

- **Semnare + criptare:**
  - utilizatorul A semnează mesajul clar M și obține semnătura $S = M^{K_{priv}^{A}} (\mathrm{mod}\ n_A)$
  - utilizatorul A criptează mesajul M și semnătura S, după care obține mesajul criptat $C = (M, S)^{K_{pub}^{B}} (\mathrm{mod}\ n_B)$
  - utilizatorul A transmite utilizatorului B mesajul C
  - utilizatorul B decriptează mesajul C de la utilizatorul A și obține perechea utilizatorul (M, S)
  - utilizatorul B poate să trimită mesajul semnat de utilizatorul A oricărei alte persoane (criptându-l cu cheia publică a persoanei respective) => *repeated forwarding* (de exemplu, un mesaj malițios)!!!

- **Rezolvare simplă (pentru un text):** în orice mesaj se adaugă numele expeditorului și numele destinatarului!

- **Rezolvare generală:**
  - utilizatorul A semnează mesajul clar M și apoi îl criptează în mesajul C
  - utilizatorul A adaugă la mesajul C cheia publică K_{pub}^{B} a utilizatorului B
  - utilizatorul A semnează mesajul $C \,||\, K_{pub}^{B}$ și îi transmite lui B perechea $(C \,||\, K_{pub}^{B}, S)$

Modele de atac asupra unei semnături digitale:

Au fost elaborate de Goldwasser, Micali și Rivest în 1988:

1) *Key-only attack* – atacatorul cunoaște doar cheia publică a semnatarului;
2) *Known message attack* – atacatorul cunoaște perechi (M, S), dar nu poate controla mesajul M (i.e., mesajele M sunt aleatorii);
3) *Adaptive chosen message attack* – atacatorul cunoaște perechi (M, S) cu mesajul M ales de el (i.e., atacatorul controlează mesajele M).

Rezultatele unui atac asupra unei semnături digitale:

1) *Total break* – atacatorul determină cheia privată a semnatarului;
2) *Universal forgery* – atacatorul poate semna orice mesaj în numele semnatarului (i.e., falsifica semnătura semnatarului), fără a-i cunoaște cheia privată;
3) *Selective forgery* – atacatorul poate semna doar anumite mesaje în numele semnatarului (i.e., atacatorul poate controla parțial mesajul M pentru care dorește să falsifice semnătura);
4) *Existential forgery* – atacatorul poate crea perechi (M, S) valide în numele semnatarului (i.e., atacatorul nu poate controla mesajul M).

Exemple RSA:

a) *Key-only attack => Existential forgery*

- atacatorul H alege un mesaj M;
- atacatorul H criptează mesajul M cu cheia publică a utilizatorului A, a cărui semnătură vrea să o falsifice, și obține $M' = M^{K_{pub}^{A}} (\mathrm{mod}\ n)$;
- perechea (M', M) este validă, adică mesajul M este o semnătură validă pentru mesajul M'!
- https://8gwifi.org/rsasignverifyfunctions.jsp:
  pentru M = "Ana are mere" se poate obține M' = "ipUJPx5QDj6b+Rd+m0l2zdtOefHo5nsxX0SPzDbqFb5Kivea997tSE2Pk6qk1sya6pP8K0CBnn2nIQrhSw8/IC+JTV1dBSAK1xv5mkXtEI1hKS6ur42MS2l9VDmBkT9tfDBJhPdNjjSik2ZQZoPyHdpEWscyZ/hwwhbJiRd+y5w=", deci M = "Ana are mere" este semnătura mesajului M' = "ipUJP..." 😊

b) *Known message attack => Existential forgery*

- atacatorul H interceptează două perechi valide (M_1, S_1) și (M_2, S_2);
- atacatorul H va calcula perechea validă $(M_1 \cdot M_2, S_1 \cdot S_2)$, respectiv produsul celor două semnături este o semnătură validă pentru produsul celor două mesaje!
- *Demonstrație*: $S_1 = M_1^{K_{priv}^A} (\mathrm{mod}\, n)$ și $S_2 = M_2^{K_{priv}^A} (\mathrm{mod}\, n)$ => $S_1 \cdot S_2 = (M_1 \cdot M_2)^{K_{priv}^A}$ => în momentul verificării semnăturii $S_1 \cdot S_2$ pentru mesajul $M_1 \cdot M_2$ vom obține mesajul $M' = (S_1 \cdot S_2)^{K_{pub}^A} = {S_1}^{K_{pub}^A} \cdot {S_2}^{K_{pub}^A} = \left(M_1^{K_{priv}^A}\right)^{K_{pub}^A} \cdot \left(M_2^{K_{priv}^A}\right)^{K_{pub}^A} = M_1 \cdot M_2$ => semnătura $S_1 \cdot S_2$ este validă pentru mesajul $M_1 \cdot M_2$!
- M_1 = "Ana are mere!" = 518399133322533717749569961296 1
- M_2 = "Ne vedem la ora 20, da?" = 75087517920304625636744938219 98827837806935142970974527
- $M_1 \cdot M_2$ = 38925304213226137337401457495636276889238684276287481041267515517265431551690890044447 = " zp>fSi&1ŹŮ^ éműëˆbxŽvIÇ»n1’§í[đ\d"

c) *Adaptive chosen message attack => Selective forgery*

- atacatorul H descompune un mesaj M ales de el în $M = M_1 \cdot M_2$;
- atacatorul H obține semnături valide pentru mesajele M_1 și M_2, adică două perechi valide (M_1, S_1) și (M_2, S_2);
- atacatorul H calculează perechea validă $(M_1 \cdot M_2, S_1 \cdot S_2) = (M, S_1 \cdot S_2)$, respectiv produsul celor două semnături este o semnătură validă pentru mesajul M ales inițial!

Semnătura RSA blind

- A fost introdusă de către David Chaum în 1982.
- Permite semnarea unui mesaj fără ca semnatarul să vizualizeze mesajul!!!

- Un utilizator B vrea ca autoritatea A să-i semneze blind mesajul M!

1. utilizatorul B generează un număr aleatoriu $r \in \{1, \dots, n_A - 1\}$
2. utilizatorul B calculează mesajul mascat (blind message):

$$M' = r^{K_{pub}^A} \cdot M \ (\mathrm{mod}\, n)$$

3. utilizatorul B trimite autorității A mesajul mascat M' și informații care să-i autentifice identitatea
4. autoritatea A verifică identitatea lui B și, dacă aceasta este validă, semnează mesajul mascat M':

$$S' = (M')^{K^A_{priv}} \pmod n = \left(r^{K^A_{pub}} \cdot M\right)^{K^A_{priv}} \pmod n =$$

$$= r^{K^A_{pub} \cdot K^A_{priv}} \cdot M^{K^A_{priv}} \pmod n = r \cdot M^{K^A_{priv}} \pmod n$$

5. autoritatea A îi trimite lui B semnătura S'
6. utilizatorul B elimină valoarea aleatorie r, astfel:

$$S = S' \cdot r^{-1} \pmod n = r \cdot M^{K^A_{priv}} \cdot r^{-1} \pmod n =$$

$$= M^{K^A_{priv}} \pmod n$$

7. utilizatorul B obține în acest mod o semnătura validă S a autorității A pentru mesajul său inițial M!!!

Aplicație:

Protocolul de vot electronic Fujioka, Okamoto și Ohta (1992) https://people.csail.mit.edu/rivest/voting/papers/FujiokaOkamotoOhta-APracticalSecretVotingSchemeForLargeScaleElections.pdf

FUNCȚII CRIPTOGRAFICE DE HASH

Funcție criptografică de hash = o funcție care se aplică unui șir binar de lungime variabilă și furnizează un șir binar de lungime fixă (de obicei, 128, 256, 512 sau 1024 de biți)

Utilitate:

- asigurarea integrității informației
- semnături digitale (de obicei, se semnează hash-ul unui document/fișier)
- stocarea și verificarea parolelor
- generarea unor chei secrete din texte ("parole")
- regăsirea informației (fișiere mari)

Proprietăți:

- sunt deterministice
- sunt rapide
- verifică *criteriul strict de avalanșă*: *modificarea unui singur bit din mesajul binar trebuie să conducă la modificarea a cel puțin jumătate din biții valorii de hash*

  Exemplu:
  M_1 = "Ana are mere!" =>
  SHA512(M_1) = 42EBE9E1DBEF46B68F4D62F0125822FBB471FC15AD2C9870EB966C8C1C332BEB3E43753E1C489A49A57F8CBEF073D81308C6D8907D372B554FBF3A78E7969079

  M_2 = "Ana are mese!" =>
  SHA512(M_2) = FEBD7F672E1022B1B8AF2DAE65D1085F6388E6AD6239C701C3427EB138ECA5E0ED195115EBE0C76CCE46C6CEA7A150A995896CCC615E33EB0327908F7ADEFD8A

  Cele două mesaje diferă printr-un singur bit, iar cele două valori de hash diferă prin 269 de biți, adică mai mult de jumătate dintre cei 512!

- sunt rezistente la *atacul în pre-imagine*: *pentru o valoare de hash H este foarte greu de determinat un mesaj M astfel încât $H = hash(M)$*

- nu prezintă *coliziuni slabe*: *pentru un mesaj M_1 este foarte greu de determinat un mesaj diferit M_2 astfel încât $hash(M_1) = hash(M_2)$*

  Exemplu: funcția SHA512 furnizează valori de hash pe 512 biți => funcția SHA512 poate furniza maxim 2^{512} valori distincte => după ce funcția SHA512 va fi aplicată asupra a $2^{512} + 1$ șiruri binare (indiferent de lungimea lor!) sigur vor apărea coliziuni!

  Totuși, $2^{512} \approx 1.34 \times 10^{154}$, deci, dacă funcția SHA512 s-ar utiliza în fiecare secundă, atunci coliziunile vor apărea sigur după aproximativ 3.171×10^{146} ani 😊

  În realitate, din cauza *paradoxului zilei de naștere* (Birthday problem - Wikipedia, Birthday attack - Wikipedia), coliziunile unei funcții de hash care generează n biți pot să apară după aplicarea sa asupra a $\sqrt{2^n} = 2^{n/2}$ șiruri binare!

- nu prezintă *coliziuni tari*: *este foarte greu de determinat perechi arbitrare* (M_1, M_2) *astfel încât* $hash(M_1) = hash(M_2)$

  Exemple:
  - Peter Selinger: MD5 Collision Demo (dal.ca)
  - cryptanalysis - Are there two known strings which have the same MD5 hash value? - Cryptography Stack Exchange

Construcția funcțiilor de hash criptografic:

- construcția Merkle – Damgård din 1979 a fost utilizată pentru crearea funcțiilor de hash MD5, SHA-1, SHA-2 etc.: Merkle–Damgård construction - Wikipedia
- construcția Sponge (Joan Daemen) din 2007 a fost utilizată pentru crearea standardului actual SHA-3: Sponge function - Wikipedia