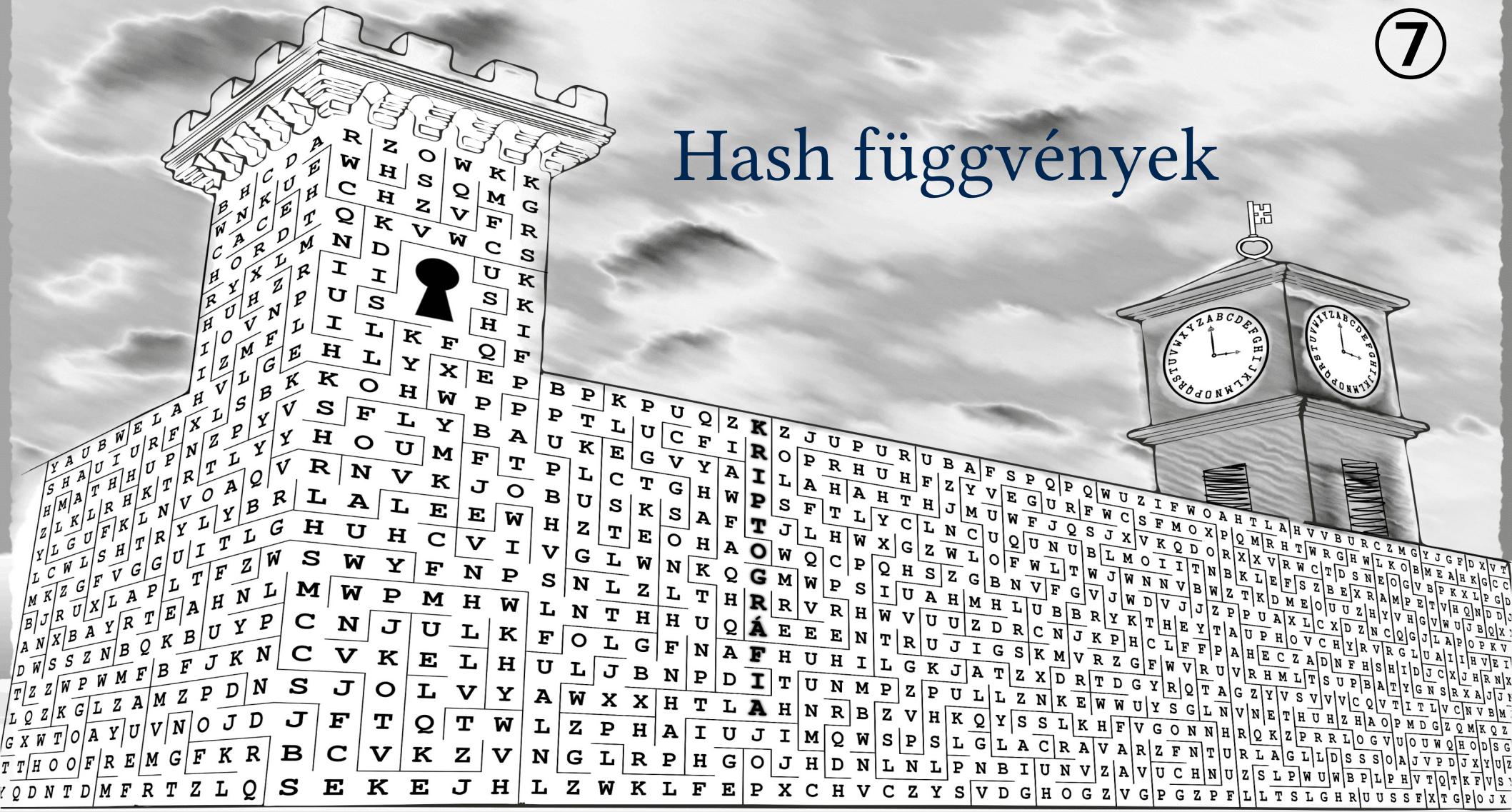


Hash függvények



Hash függvények

A hash-függvények (lenyomatkészítő függvények):

- tetszőleges méretű adatot fix hosszúságúra sűrítenek
- az adatot blokkonként dolgozzák fel
- valamilyen tömörítő függvényen vagy tömbtitkosító eljáráson alapulnak, amit iteráltan alkalmaznak

Hash függvények

Jelölések:

- \mathbb{M} – tetszőleges hosszú bitsorozatok (**üzenetek**) halmaza
- \mathbb{M}_r – r hosszú bitsorozatok (**üzenetek**) halmaza
- ha $M \in \mathbb{M}$ egy üzenet, akkor $L(M)$ jelöli a bitekben mért hosszát
- $h : \mathbb{M} \rightarrow \mathbb{M}_r$ **hash függvény** egy nyilvános függvény, ami hatékonyan (polinomiális időben) kiszámolható
- ha $M \in \mathbb{M}$ egy üzenet, akkor a $h(M) \in \mathbb{M}_r$ bitsorozat a **lenyomata** (message digest)

Hash függvények

Legyenek $M, M' \in \mathbb{M}$ és $H \in \mathbb{M}_r$ sorozatok. A $h: \mathbb{M} \rightarrow \mathbb{M}_r$ hash függvény:

- **egyirányú** (vagy **őskép-ellenálló**), ha adott H -hoz nehéz (gyakorlatilag kivitelezhetetlen számításigényű) feladat találni olyan M sorozatot, amire $h(M) = H$.
- **gyengén ütközésmentes** (vagy **második őskép-ellenálló**), ha adott M -hez nehéz egy másik M' -et találni, amire $h(M') = h(M)$.
- **ütközésmentes**, ha gyengén ütközésmentes és nehéz olyan különböző elemekből álló (M, M') párokat találni, amire $h(M') = h(M)$. Egy ilyen párt **ütközésnek** nevezünk.

Hash függvények

- A gyakorlatban használt hash függvényeknek rendelkezniük kell az ún. **lavina-hatással** (kismértékű változás az üzenetben nagymértékű változást idéz elő a lenyomatban).
- A hash függvények bizonyos biztonságos, „hűséges” és irreverzibilis módon tömörítenek össze nagy méretű adatokat.
- Alkalmazások: hitelesítés, digitális aláírás, üzenetek épségének ellenőrzése.

Hash függvények szerkesztése

- Szerkesztési módszerek: Merkle–Dåmgard-séma, Davies–Meyer-séma
- A gyakorlatban használt hash függvények majdnem mindegyike a Merkle–Dåmgard-sémát követi.
 - A módszer lényege: a bemeneti adatot egyenlő méretű blokkokra osztjuk, majd mindenblokkot iteratív módon egy tömörítő függvény segítségével redukáljuk.

A Merkle–Dåmgard-séma

Jelölések:

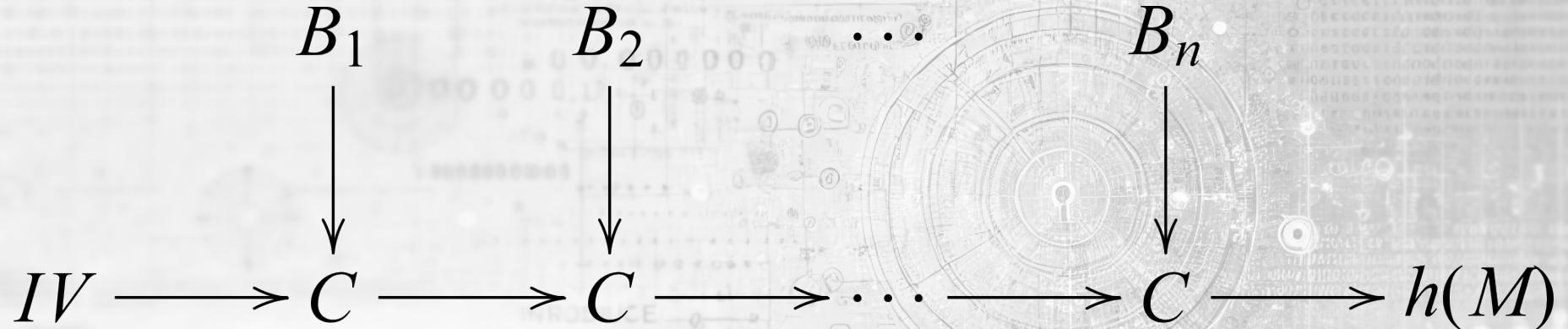
- $M \in \mathbb{M}$ – üzenet
- $C: \mathbb{M}_r \times \mathbb{M}_t \rightarrow \mathbb{M}_r$ ($t \geq r$) – tömörítő függvény
- $L(M)_s$ – az M üzenet bithossza s biten ábrázolva
(általában $s = 64$, tehát az üzenet hossza kisebb kell legyen, mint 2^{64} bit)

A Merkle–Dåmgard-séma

A Merkle–Dåmgard-konstrukció lépései:

1. Az M üzenetet megtoldjuk egy 1 értékű bittel, majd egy sorozat 0 értékű bittel és végül $L(M)_s$ -sel úgy, hogy a kiegészített üzenet teljes hossza t -nek legkisebb többszöröse legyen. Jelöljük \tilde{M} -mel a kiegészített üzenetet. Ennek alakja tehát $\tilde{M} = (M \parallel 1 \parallel 0 \dots 0 \parallel L(M)_s)$, ahol \parallel bitsorozatok összeillesztését (konkatenálását) jelöli.
2. \tilde{M} -met felosztjuk t bites blokkokra, legyenek ezek: B_1, \dots, B_n .
3. Választunk egy r hosszúságú IV kezdőértéket, és legyen $H_0 := IV$.
4. minden $i = 1, \dots, n$ -re, legyen $H_i := C(H_{i-1}, B_i)$.
5. Legyen $h(M) = H_n$.

A Merkle–Dåmgard-séma



Tétel. Ha a C tömörítő függvény ütközésmentes, akkor a h hash függvény is ütközésmentes.

A Davies–Meyer-séma

- Legyen $E_k: \mathbb{M}_r \rightarrow \mathbb{M}_r$ egy tömbtitkosító kódoló eljárása, ahol k a titkosításhoz használt kulcs.
- Az előbbi jelölésekkel használva a C tömörítő függvényt a köv. módon adjuk meg:

$$C(H_{i-1}, B_i) := E_{B_i}(H_{i-1}) \oplus H_{i-1}$$

Az MD5 (Message Digest 5)

- az egyik legelterjedtebb hash függvény
- széles körben használják internetes alkalmazásokban (pl. adatok épségének ellenőrzése)
- Ronald Rivest fejlesztette ki 1991-ben
- 1992-ben szabványosították internetes alkalmazásokban való használatra (RFC 1321)

Az MD5 (Message Digest 5)

- brute force módon (és kriptoanalízissel is) ütközés található, ezért hitelesítésre, digitális tanúsítványok kiállítására, már nem használják
- a Merkle–Dåmgard-konstrukciót követi
- a tömörítő függvénye $128 \times 512 \rightarrow 128$ típusú, tehát $r = 128$, $t = 512$ és az üzenet hosszát $s = 64$ biten tárolja

Az MD5 (Message Digest 5)

Az MD5 algoritmus lépései:

1. Az M üzenetet megtoldjuk egy 1 értékű bittel, egy sorozat 0 értékű bittel és végül $L(M)_{64}$ -el úgy, hogy a kiegészített üzenet teljes hossza 512-nek legkisebb többszöröse legyen. Jelöljük \tilde{M} -mel a kiegészített üzenetet. Ennek alakja tehát $\tilde{M} = (M \parallel 1 \parallel 0 \dots 0 \parallel L(M)_{64})$, ahol \parallel bitsorozatok összeillesztését jelöli.
2. \tilde{M} -et felosztjuk 512 bites blokkokra, legyenek ezek: M_1, \dots, M_n . Továbbá, minden blokkot felosztunk 16 szóra (egy szó 32 bitet tartalmaz). Jelöljük $M_{i,j}$ az i -edik blokk j -edik szavát, ahol $j \in \{1, \dots, 16\}$ és $i \in \{1, \dots, n\}$.

Az MD5 (Message Digest 5)

3. Választunk egy négy szóból álló (128 bites) $IV = (X_1, X_2, X_3, X_4)$ kezdeti értéket, ahol a négy szó a következő értékeket kapja (hexadecimális számrendszerben):

$$X_1 = 67452301, X_2 = efcdab89, X_3 = 98badcfe, X_4 = 10325476.$$

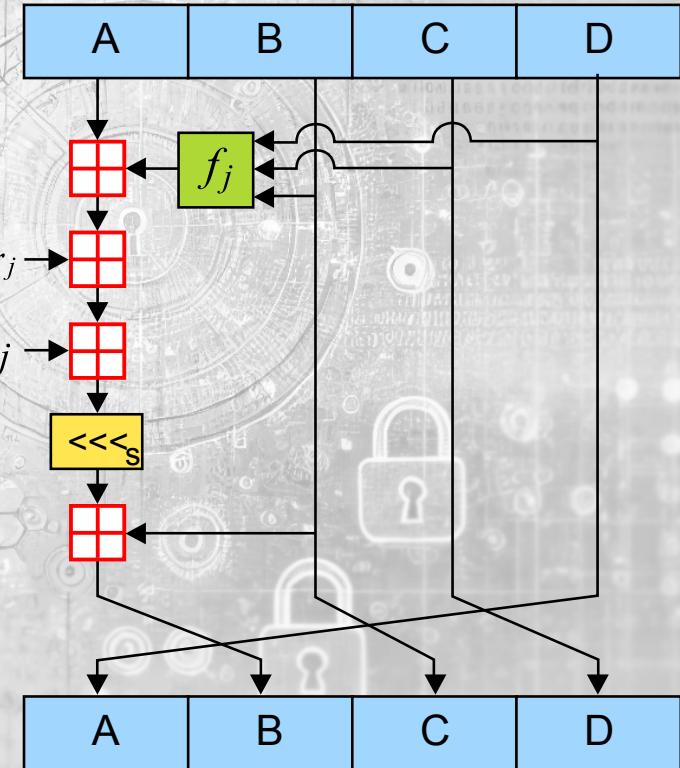
4. Végrehajtjuk a következőket:

```

 $A := X_1, B := X_2, C := X_3, D := X_4$ 
for  $i := 1, \dots, n$  do
    for  $j := 1, \dots, 64$  do
         $T := add(A, f_j(B, C, D), M_{i,1+r_j}, t_j)$ 
         $A := D$ 
         $D := C$ 
         $C := B$ 
         $B := add(B, rol(T, s_j))$ 
    end for
     $X_1 := add(X_1, A)$ 
     $X_2 := add(X_2, B)$ 
     $X_3 := add(X_3, C)$ 
     $X_4 := add(X_4, D)$ 
end for

```

Az MD5 (Message Digest 5)



5. Legyen $h(M) = (X_1, X_2, X_3, X_4)$.

Az MD5 (Message Digest 5)

Az algoritmus leírásában szereplő függvények és egyéb jelölések:

- f_i egy bitenkénti Boole-függvény (\oplus jelöli a XOR, \vee az OR, \wedge az AND és \neg a NOT bitműveletet):

$$f_i(X, Y, Z) = \begin{cases} (X \wedge Y) \vee (\neg(X) \wedge Z), & \text{ha } i = 1, \dots, 16 \\ (X \wedge Y) \vee (\neg(Z) \wedge Y), & \text{ha } i = 17, \dots, 32 \\ X \oplus Y \oplus Z, & \text{ha } i = 33, \dots, 48 \\ (\neg(Z) \vee X) \oplus Y, & \text{ha } i = 49, \dots, 64 \end{cases}$$

- $t_i = [2^{32}|\sin(i)|]$, ahol $i = \overline{1, 64}$ radiánban van megadva

Az MD5 (Message Digest 5)

$$\bullet \quad r_i = \begin{cases} i - 1, & \text{ha } i = 1, \dots, 16 \\ (1 + 5(i - 1)) \bmod 16, & \text{ha } i = 17, \dots, 32 \\ (5 + 3(i - 1)) \bmod 16, & \text{ha } i = 33, \dots, 48 \\ 7(i - 1) \bmod 16, & \text{ha } i = 49, \dots, 64 \end{cases}$$

- $\text{add}(X, Y, \dots) = (X + Y + \dots) \bmod 2^{32}$
- $\text{rol}(X, s) = X$ bitjeit s pozícióval ciklikusan balra toljuk

Az MD5 (Message Digest 5)

$$\bullet \ s_i = \begin{cases} 4, & \text{ha } i = 33, 37, 41, 45 \\ 5, & \text{ha } i = 17, 21, 25, 29 \\ 6, & \text{ha } i = 49, 53, 57, 61 \\ 7, & \text{ha } i = 1, 5, 9, 13 \\ 9, & \text{ha } i = 18, 22, 26, 30 \\ 10, & \text{ha } i = 50, 54, 58, 62 \\ 11, & \text{ha } i = 34, 38, 42, 46 \\ 12, & \text{ha } i = 2, 6, 10, 14 \end{cases} \quad \begin{cases} 14, & \text{ha } i = 19, 23, 27, 31 \\ 15, & \text{ha } i = 51, 55, 59, 63 \\ 16, & \text{ha } i = 35, 39, 43, 47 \\ 17, & \text{ha } i = 3, 7, 11, 15 \\ 20, & \text{ha } i = 20, 24, 28, 32 \\ 21, & \text{ha } i = 52, 56, 60, 64 \\ 22, & \text{ha } i = 4, 8, 12, 16 \\ 23, & \text{ha } i = 36, 40, 44, 48 \end{cases}$$

Az MD5 (Message Digest 5)

Lavina-effektus:

- $\text{MD5}(\text{Hello})=8b1a9953c4611296a827abf8c47804d7$
- $\text{MD5}(\text{Hello!})=952d2c56d0485958336747bcdd98590d$
- $\text{MD5}(\text{Message Digest 5 is one of the most popular hash functions.})=2ccf2ef3b66683e2d97a21a42da5b8e4$
- $\text{MD5}(\text{Message Digest 5 is one of the most popular hash functions})=a39981f6e0d67364dc194b0577a71c1f$

Az SHA-1 (Secure Hash Algorithm 1)

- Az SHA-1 szintén széles körben használt hash függvény
- 1995-től NIST szabvány
- a Merkle–Dåmgard-konstrukciót követi
- a tömörítő függvénye $160 \times 512 \rightarrow 160$ típusú, tehát $r = 160$, $t = 512$ és az üzenet hosszát $s = 64$ biten tárolja

Az SHA-1 (Secure Hash Algorithm 1)

Az SHA-1 algoritmus lépései:

1. Az M üzenetet megtoldjuk egy 1 értékű bittel, egy sorozat 0 értékű bittel és végül $L(M)_{64}$ -gyel úgy, hogy a kiegészített üzenet teljes hossza 512-nek legkisebb többszöröse legyen. Jelöljük \tilde{M} -mel a kiegészített üzenetet. Ennek alakja tehát $\tilde{M} = (M \parallel 1 \parallel 0 \dots 0 \parallel L(M)_{64})$, ahol \parallel bitsorozatok összeillesztését jelöli.
2. \tilde{M} -met felosztjuk 512 bites blokkokra, legyenek ezek: B_1, \dots, B_n . Továbbá, minden blokkot felosztunk 16 szóra (egy szó 32 bitet tartalmaz). Jelöljük $B_{i,j}$ az i -edik blokk j -edik szavát, ahol $j \in \{1, \dots, 16\}$ és $i \in \{1, \dots, n\}$. Induktívan értelmezzük:

Az SHA-1 (Secure Hash Algorithm 1)

$$B_j(i) = \begin{cases} B_{i,j}, & \text{ha } j \in \{1, \dots, 16\} \\ \text{rol}(B_{j-3}(i) \oplus B_{j-8}(i) \oplus B_{j-14}(i) \oplus B_{j-16}(i), 1), & \text{ha } j \in \{17, \dots, 80\} \end{cases}$$

3. Választunk egy öt szóból álló (160 bites) $IV = (X_1, X_2, X_3, X_4, X_5)$ kezdeti értéket, ahol az öt szó a következő értékeket kapja (hexadecimális számrendszerben):

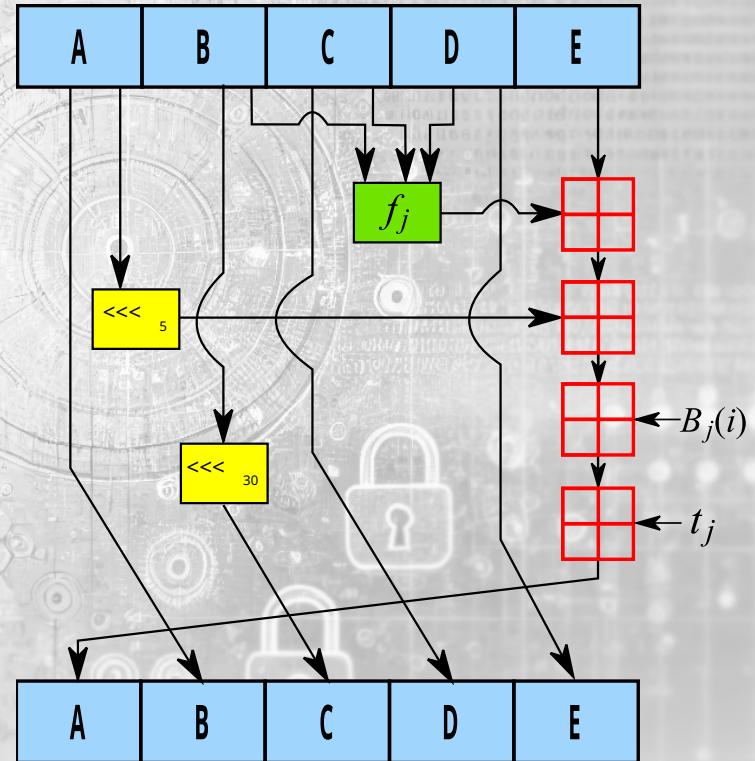
$$X_1 = 67452301, \quad X_2 = efcdab89, \quad X_3 = 98badcfe, \\ X_4 = 10325476, \quad X_5 = c3d2e1f0.$$

4. Végrehajtjuk a következőket:

```
A := X1, B := X2, C := X3, D := X4, E := X5
for i := 1, ..., n do
    for j := 1, ..., 80 do
        T := add(rol(A, 5), fj(B, C, D), E, Bj(i), tj)
        E := D
        D := C
        C := rol(B, 30)
        B := A
        A := T
    end for
    X1 := add(X1, A)
    X2 := add(X2, B)
    X3 := add(X3, C)
    X4 := add(X4, D)
    X5 := add(X5, E)
end for
```

5. Legyen $h(M) = (X_1, X_2, X_3, X_4, X_5)$.

Az SHA-1 (Secure Hash Algorithm 1)



Az SHA-1 (Secure Hash Algorithm 1)

Az algoritmus leírásában szereplő függvények és egyéb jelölések:

- f_i egy bitenkénti Boole-függvény, a bemeneti és kimeneti értékek bitek (\oplus jelöli az XOR, \vee az OR, \wedge az AND és \neg a NOT bitműveletet):

$$f_i(X, Y, Z) = \begin{cases} (X \wedge Y) \vee (\neg(X) \wedge Z), & \text{ha } i = 1, \dots, 20 \\ X \oplus Y \oplus Z, & \text{ha } i = 21, \dots, 40 \\ (X \wedge Y) \vee (X \wedge Z) \vee (Y \wedge Z), & \text{ha } i = 41, \dots, 60 \\ X \oplus Y \oplus Z, & \text{ha } i = 61, \dots, 80 \end{cases}$$

Az SHA-1 (Secure Hash Algorithm 1)

$$\bullet \quad t_i = \begin{cases} 5a827999, & \text{ha } i = 1, \dots, 20 \\ 6ed9eba1, & \text{ha } i = 21, \dots, 40 \\ 8f1bbcdcc, & \text{ha } i = 41, \dots, 60 \\ ca62c1d6, & \text{ha } i = 61, \dots, 80 \end{cases}$$

- $\bullet \quad add(X, Y, \dots) = (X + Y + \dots) \bmod 2^{32}$
- $\bullet \quad rol(X, s) = X$ bitjeit s pozícióval ciklikusan balra toljuk

Az SHA-1 (Secure Hash Algorithm 1)

Lavina-effektus:

- $\text{SHA-1(Hello)} = \text{f7ff9e8b7bb2e09b70935a5d785e0cc5d9d0abf0}$
- $\text{SHA-1(Hello!)} = \text{69342c5c39e5ae5f0077aecc32c0f81811fb8193}$
- $\text{SHA-1(Secure Hash Algorithm 1 is one of the most popular hash functions.)} = \text{e72d8eb5282c3717319972db275946244f63ded8}$
- $\text{SHA-1(Secure Hash Algorithm 1 is one of the most popular hash functions)} = \text{49283b6dc3bd9b60b51bb94b640a4ba7b9781254}$

A hash függvények alkalmazásai

- üzenetek sértetlenségének ellenőrzése
- jelszavas azonosítás
- kötelezettségvállalási sémák
- biztonságos kulcscsere

Üzenetek sértetlenségének ellenőrzése

Ha fennáll az üzenet szándékos (rosszindulatú) módosításának veszélye, akkor a következő módon járhatunk el:

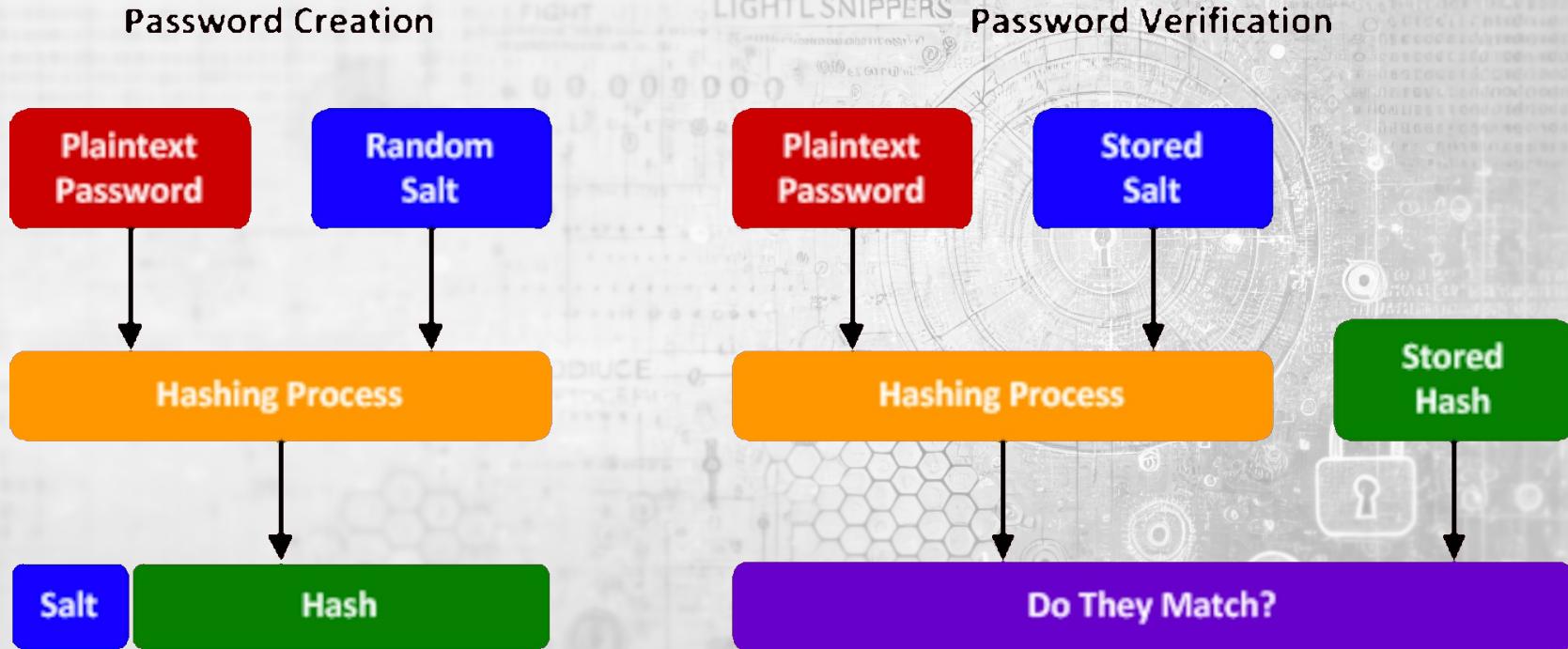
- Az M üzenetet a nyilvános (nem biztonságos) csatornán, a $H = h(M)$ lenyomatot pedig egy biztonságos csatornán továbbítjuk (ebben az esetben H az üzenethitelesítő kód, vagy angolul Message Authentication Code, MAC).
- Feltételezzük, hogy a címzett megkapja a (valószínűleg módosított) M' üzenetet és a H lenyomatot.
- A címzett leellenőrzi a $H = h(M')$ egyenlőséget. Ha az egyenlőség fennáll, akkor majdnem bizonyosan $M = M'$, tehát a kapott üzenet nem volt módosítva. Ha $H \neq h(M')$, akkor az üzenetet módosította valaki.

Üzenetek sértetlenségének ellenőrzése

Ha csak a csatornán fellépő zajok miatt sérülhet az üzenet (adatátviteli hiba miatt), akkor nincs szükségünk biztonságos csatornára:

- Az M üzenetet és a $H = h(M)$ lenyomatot ugyanazon a csatornán küldjük.
- Feltételezzük, hogy a címzett megkapja a (valószínűleg sérült) M' üzenetet és a H lenyomatot.
- A címzett leellenőrzi a $H = h(M')$ egyenlőséget. Ha az egyenlőség fennáll, akkor majdnem bizonyosan $M = M'$, tehát a kapott üzenet nem sérült. Ha $H \neq h(M')$, akkor az üzenet adatátviteli hiba miatt sérült.

Jelszavas azonosítás



Kötelezettségvállalási sémák

- Elsőség bizonyítása
 - Alice és Bob versenyeznek, az a kihívás, hogy melyikük old meg hamarabb egy nehéz feladatot.
 - Alice megoldott egy fontos részfeladatot (amihez egy eredeti ötlet kellett), a többi rutinmunka, de időbe telik.
 - Alice szeretné, ha mindenki tudná, ő volt az első, aki rájött a megoldás „kulcsára”.
 - Alice nem akarja publikálni a részeredményét, mert ezzel Bob munkáját segítené.

Kötelezettségvállalási sémák

- Alice egy h hash függvény segítségével kiszámolja a részeredményt tartalmazó M dokumentum $H = h(M)$ lenyomatát
- Alice küld egy üzenetet Bobnak és a szakértőknek, amiben értesíti őket, hogy áttörést ért el, az üzenethez csatolja a H lenyomatot
- Ha Bob is megoldja a fontos részfeladatot és publikálja, mielőtt Alice az egész feladatot megoldaná, akkor Alice is közzéteszi az M dokumentumot és az üzenetben már régebb elküldött lenyomat segítségével tudja bizonyítani elsőségét

Kötelezettségvállalási sémák

- Ki végezzen el egy „kellemetlen” feladatot?
 - Alice és Bob telefonon keresztül, „telefonos pénzfeldobással” akarják eldönteni, hogy kinek kell elvégeznie egy kellemetlen feladatot.
 - Alice és Bob nem bíznak egymásban, de ki tudnak egyezni egy h hash függvény használatában.
 - Alice választ egy M természetes számot, kiszámolja a $H = h(M)$ lenyomatot és ezt megmondja Bobnak.
 - Alice arra kéri Bobot, hogy tippelje meg, M páros-e vagy páratlan: ha eltalálja, akkor Alice-é a feladat, ellenkező esetben a Bobé.

Kötelezettségvállalási sémák

- Bob megjegyzi a H lenyomatot és közli Alice-szel a tippjét.
- Alice megmondja Bobnak, hogy talált-e vagy sem.
- Bob nem bízik Alice-ben, ezért megkérdezi, hogy mi volt az eredeti M szám.
- Alice közli Bobbal az M értékét, Bob kiszámolja $h(M)$ -et és összehasonlítja H -val.

Megj: Alice mindenkor nyerni tud, ha olyan ütközéseket tudna előállítani, ahol $h(M) = h(M')$ és M illetve M' közül egyik páros, a másik páratlan.

Biztonságos kulcscsere

- Alice és Bob meg akarnak egyezni egy r -bites kulcsban.
- Alice és Bob megegyeznek egy r bit hosszúságú lenyomatokat készítő hash függvény használatában.
- Alice véletlenszerűen választ $2^{r/2}$ kulcsot, kiszámolja a kulcsok lenyomatait és ezeket nyilvánossá teszi.
- Bob hasonlóan jár el.

Biztonságos kulcscsere

- A születésnap-paradoxon eredményeként, nagyon valószínű, hogy a nyilvánossá tett lenyomatlistákban van közös elem.
- Az lesz a közös kulcs, aminek a lenyomata mindkét listában benne van.
- Mivel megtörténhet, hogy két különböző kulcsnak ugyanaz legyen a lenyomata, ajánlott egy második hash függvényt használni, és az ezzel készített lenyomatokat is összehasonlítani.