# SHA-1 algoritmus

Smidla József

Rendszer- és Számítástudományi Tanszék
Pannon Egyetem

2012. 4. 3.

# Tartalom

# Digitális aláírás

Cél: Hitelesíteni kell az elküldött üzenetet, azaz:

- A küldő fél ne tagadhassa le, hogy ő küldte az üzenetet
- Ha a támadó az üzenetet megváltoztatja, azt észrevegyük
- Nem minden esetben kell titkosítani az üzenetet, előfordulhat, hogy csak hitelesítésre van szükségünk

Persze még hitelesíteni kell azt a személyt, akivel beszélgetünk, hogy tényleg az-e, akinek mondja magát. Most viszont feltesszük, hogy a másik félben megbízhatunk, és csak az üzenet hitelesítésére koncentrálunk. Vizsgáljuk meg, hogy a szimmetrikus kódolók alkalmasak-e a feladatra.

# Üzenet hitelesítés, példa

1. Tegyük fel, hogy Bob egy megrendelést szeretne küldeni Alice-nak, például ezt:
   "Kérnék jövő hétre 300 darab lila körömlakkot, valamint 100 doboz csillámport."
2. Bob az üzenetet szimmetrikus algoritmussal titkosítja a közös $k$ kulccsal, majd elküldi Alice-nak
3. Bobot felvilágosítják a barátai, hogy egy igazi férfi nem használ ilyen szereket
4. Bob rájön, hogy butaságot csinált, de Alice már megküldte neki a csomagot
5. Bob nem hajlandó kifizetni a szállítmányt, azt mondja, hogy nem is ő volt a megrendelő
6. Mindketten bírósághoz fordulnak...

# Üzenet hitelesítés, példa

7. Bob azt állítja, hogy a titkosított üzenetet Alice hozta létre a $k$ kulccsal
8. Alice viszont azt állítja, hogy az üzenetet Bob hozta létre a $k$ kulccsal
9. A bíró nem tud dönteni, hiszen a közös kulcs segítségével bármelyikük létrehozhatta az üzenetet

Szimmetrikus kódoló helyett alkalmazzunk aszimmetrikus kódolót, ugyanis:

- A saját privát kulcsát senki sem hozza nyilvánosságra, csak a publikust
- Bob kódolja az üzenetet a saját privát kulcsával
- Alice dekódolja az üzenetet Bob publikus kulcsával
- Bob nem tudja letagadni az üzenetet, hiszen a privát kulcsot csak ő ismeri, Alice képtelen üzenetet hamisítani

# Digitális aláírás RSA-val

Bob üzenetet szeretne küldeni Alice-nak, az üzenetet digitális aláírással látja el, az elküldendő üzenet: $x$

1. Bob privát kulcsa: $k_{pr} = d$, nyilvános kulcsa: $k_{pub} = (n, e)$
2. Bob eljuttatja $(n, e)$-t Alice-nak
3. Bob kiszámolja az $x$-re a digitális aláírást: $s \equiv x^d \mod n$
4. Bob elküldi Alice-nak $(x, s)$-t
5. Alice ellenőrzi a kapott aláírást: $x' \equiv s^e \mod n$
6. Ha $x = x'$, akkor az aláírás helyes, egyébként helytelen

Mivel csak Bob ismeri $d$ értékét, ezért senki más nem képes a megfelelő aláírást előállítani
Jegyezzük meg, hogy $(x^d)^e \equiv x^{de} \equiv x \mod n$, ez következik az RSA helyességének bizonyításából
Megjegyzés: Az $x$ nem lehet nagyobb, mint $n$, ezért nagyobb üzeneteknél bonyolultabb lesz a megoldás

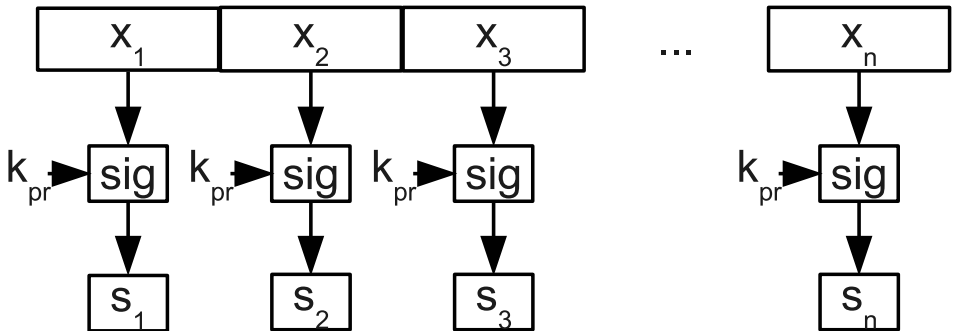
# Digitális aláírás RSA-val, példa

Legyen $x = 4$

1. Legyen $p = 3$ és $q = 11$
2. $n = pq = 33$
3. $\varphi(n) = (3-1)(11-1) = 20$
4. Legyen $e = 3$
5. $d \equiv e^{-1} \equiv 7 \mod 20$
6. Bob elküldi Alice-nak (33,3)-at
7. Bob kiszámolja a digitális aláírást az $x = 4$ üzenetre: $s = x^d \equiv 4^7 \equiv 16 \mod 33$
8. Bob elküldi $(x, s)$-t
9. Alice kiszámolja $x'$ értékét: $x' = s^e \equiv 16 \equiv 4 \mod 33$
10. Mivel $x \equiv x' \mod 33$, ezért az aláírás helyes

# Üzenet hitelesítés aszimmetrikus kódolóval

1. megközelítés: Osszuk fel blokkokra az üzenetet, majd minden blokkot kódoljunk a privát kulccsal, így megkapjuk az aláírást: minden blokkhoz tartozik egy kódolt blokk
Ezt az aláírást senki más nem képes reprodukálni

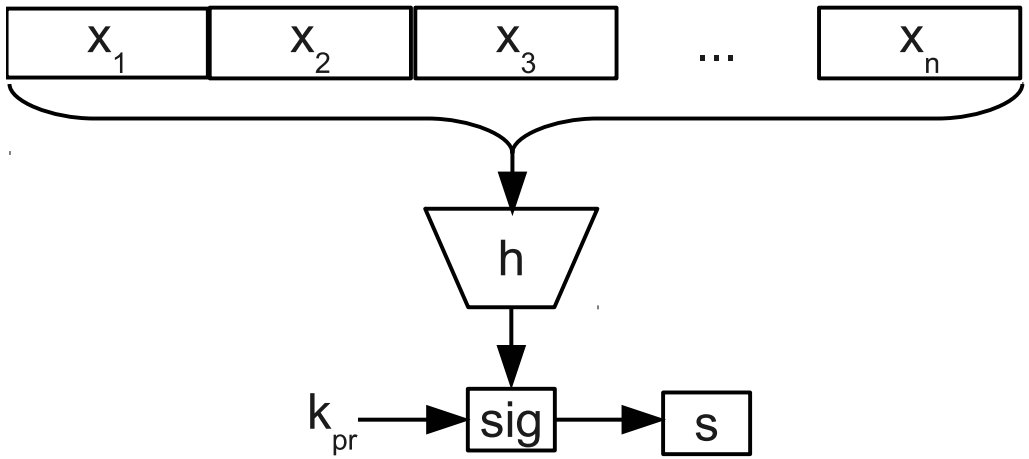



Hátrányok

- Az aszimmetrikus kódolók lassúak, nem hatékony rengeteg blokkra végrehajtani a kódolást
- A digitális aláírás mérete túl nagy
- A támadó átrendezheti az elküldött üzenet blokkjait, és hasonlóan az azokhoz tartozó aláírásokat anélkül, hogy ezt észrevennénk

# Üzenet hitelesítés aszimmetrikus kódolóval

2. megközelítés: Számoljunk egy rövid hash-t a teljes üzenetre, majd azt kódoljuk aszimmetrikus kódolóval, az eredmény lesz a digitális aláírás




Előny: A módszer mentes az előző megközelítés hátrányaitól: Mivel a hash mérete kicsi, ezért nem kell blokkokra bontani

# Üzenet hitelesítés aszimmetrikus kódolóval

Bob üzenetet szeretne küldeni Alice-nak, az üzenet: $x$

1. Bob nyílt kulcsa: $k_{pub}$, privát kulcsa: $k_{pr}$
2. Bob kiszámolja a hash-t az üzenetre: $z = h(x)$
3. Bob $z$-ből kiszámolja a digitális aláírást: $s = \text{sig}_{k_{pr}}(z)$
4. Bob elküldi $(x, s)$-t
5. Alice kiszámolja $z'$-t: $z' = h(x)$
6. Alice visszafejti $s$-ből $z$-t, és megnézi, hogy az egyezik-e $z'$-vel
7. Ha $z' = k_{pub}(s)$, akkor a digitális aláírás helyes, egyébként helytelen

Tegyük fel, hogy a támadó ki akar tolni Alice-al és Bobbal:

1. A támadó ismeri Bob nyilvános kulcsát
2. Visszafejti $s$-ből $z$-t
3. Megváltoztatja az $x$ üzenetet $x'$-re, majd kiszámolja $x'$ hash értékét: $z' = h(x')$
4. Viszont a támadónak szüksége van Bob privát kulcsára ahhoz, hogy $z'$-t helyesen kódolhassa, erre viszont nincs lehetősége

# Üzenetpecsétek

Az üzenetpecsét algoritmusok egy $x$ $n$ bites bemenetből egy $m$ bites kimenetet, $h(x)$-et állítanak elő
$m$ és $n$ nem feltétlen egyenlőek
Követelmények:

1. $h(x)$-ből ne lehessen visszafejteni $x$-et, azaz $h$ egyirányú függvény legyen
2. Adott $x_1$-hez ne lehessen könnyen $x_2$-t találni, ahol $h(x_1) = h(x_2)$
3. Nehezen lehessen tetszőleges $x_1$-et és $x_2$-t találni, ahol $h(x_1) = h(x_2)$

Mindig van több olyan üzenet, melyek pecsétje egyezik, ezért úgy kell megkonstruálni az ilyen algoritmusokat, hogy nehéz legyen ilyen párokat találni

# Születésnap támadás

Bob üzenetet ($x$) szeretne küldeni Alice-nak

1. Bob nyílt kulcsa: $k_{pub}$, privát kulcsa: $k_{pr}$
2. Bob kiszámolja a hash-t az üzenetre: $z = h(x)$
3. Bob $z$-ből kiszámolja a digitális aláírást: $s = \text{sig}_{k_{pr}}(z)$
4. Bob elküldi $(x, s)$-t
5. A támadó elfogja $(x, s)$-t, majd visszafejti $s$-ből $z$-t
6. A támadó keres olyan neki kedvező $x'$-et, amelyre $h(x') = h(x)$
7. A támadó Alice-nak $(x', s)$-t küldi tovább
8. Alice azt hiszi, hogy az üzenet sértetlen, mivel $x'$-re helyes az üzenetpecsét

A támadó azért járt sikerrel, mert egy $z$ pecséthez gyakorlatilag végtelen sok üzenet van, melyek pecsétje $z$, kérdés, hogy mennyi idő alatt lehet ilyen hamis üzenetet találni...

# Születésnap paradoxon

Hány ember legyen egy szobában ahhoz, hogy legalább 0.5 legyen a valószínűsége annak, hogy van két ember, kiknek azonos napon van a születésnapjuk? Tegyük fel, hogy nincs szökőév, és nincsenek ikrek a szobában.

# Születésnap paradoxon

Hány ember legyen egy szobában ahhoz, hogy legalább 0.5 legyen a valószínűsége annak, hogy van két ember, kiknek azonos napon van a születésnapjuk? Tegyük fel, hogy nincs szökőév, és nincsenek ikrek a szobában.
Első válasz az szokott lenni, hogy 365 / 2 ember kell, ám valójában elég 23

- Vegyük észre, hogy nem egy kiválasztott személyhez keresünk olyan embert, akinek a születésnapja vele egy napon van, hanem párokat keresünk
- $n = 23$ személy esetén 23*22 / 2 = 253 pár van, tehát jóval több lehetőségünk van, mint gondoltuk
- P(mindenki másik napon született) =

$$\frac{364}{365}\frac{363}{365}\cdots\frac{365-n+1}{365}$$

- P(legalább két embernek azonos napon van a születésnapja) =

$$1-\frac{364}{365}\frac{363}{365}\cdots\frac{365-n+1}{365}$$

# Születésnap paradoxon

- $n = 23$ esetén a keresett valószínűség $= 0.507$
- Ha az üzenetpecsét algoritmus 80 bites kimenetet ad, akkor azt gondolnánk, hogy a támadónak $2^{80}$ üzenet között kell keresnie olyant, amely pecsétje egyezik egy adott pecséttel
- Ám a születésnap paradoxon miatt elég $2^{40}$ üzenetet végigpróbálnia: Ezt mai közönséges laptopokkal is könnyen elvégezhetjük...
- Ezért az üzenetpecsét algoritmusok kimenetének legalább 128 bitesnek kell lennie, de a legtöbb ennél nagyobb kimenetet generál
- Az SHA-1 160 bites kimenetet ad

# Alapműveletek SHA-1-hez

## Bitenkénti XOR

Adottak A és B bitsorozatok

$A \oplus B = C$, $C_i$ akkor és csak akkor 1, ha $A_i \neq B_i$

$01101 \oplus 01010 = 00111$

## Bitenkénti AND

Adottak A és B bitsorozatok

$A \wedge B = C$, $C_i$ akkor és csak akkor 1, ha $A_i = B_i = 1$

$01101 \wedge 01010 = 01000$

## Bitenkénti OR

Adottak A és B bitsorozatok

$A \vee B = C$, $C_i$ akkor és csak akkor 1, ha $A_i = 1$, vagy $B_i = 1$

$01101 \vee 01010 = 01111$

# Alapműveletek SHA-1-hez

## balra_forgatás

Adott egy $x$ bitsorozat, ezt forgassuk el balra $n$-el, $n$ kisebb, mint $x$ hossza:

1. Az $x$ első $n$ bitjét $x$ végére fűzzük
2. $x$-et balra toljuk $n$-el, így az előzőleg módosított $x$ első $n$ bitje elveszik

Példa: balra_forgatás(110010, 2) = 001011

## Összeadás mod $2^{32}$-ben

Adottak A és B 32 bites előjel nélküli egész változók

$A + B \equiv C \bmod 2^{32}$

Példa: adjuk össze az alábbi hexadecimális formában adott értékeket:

CA62C1D6 + 8F1BBCDC $\equiv$ 597E7EB2 mod 100000000

Szerencsére C/C++ esetén ez a művelet egyszerű: Két unsigned int összeadása mod $2^{32}$-ben történik

# SHA-1

Számoljuk meg tetszőleges $l$ bites $m$ üzenet bitsorozat hash értékét SHA-1 algoritmussal.
Az algoritmus 512 bit hosszúságú blokkokra osztja a bemenetet.
Első lépésként ki kell bővíteni a bemenetet a hosszára vonatkozó 64 bites információval úgy, hogy a módosított bemenet 512 hosszú blokkokból álljon. A maradék helyet 0-val töltjük fel.

1. Az $m$ bitsorozat után egy 1-es bitet fűzünk
2. Számoljuk ki, hogy hány darab 0-val kell kibővíteni a sorozatot: $k \equiv 512 - 64 - 1 - l = 448 - (l + 1) \mod 512$
3. Az 1-es után fűzzünk $k$ darab 0-t $m$-hez
4. Végül fűzzünk $m$-hez egy 64 bites előjel nélküli egészt big endian kódolással, mely $l$ értékét tárolja

Megjegyzés: Az SHA-1 algoritmus legfeljebb $2^{64} - 1$ bit hosszúságú adathalmazt képes feldolgozni.

# SHA-1: Példa inicializálásra

Legyen a bemenet a következő string: m = "abc"

$$\underbrace{01100001}_{\text{a}}\ \underbrace{01100010}_{\text{b}}\ \underbrace{01100011}_{\text{c}}$$

1. Az üzenet után egy 1-es bitet fűzünk: 01100001 01100010 01100011 1
2. Mivel $l = 24$, ezért $k = 448 - (24 + 1) \mod 512 = 423$ darab 0-t is a bitsorozat végére fűzünk:
   01100001 01100010 01100011 1 $\underbrace{0000\ldots000}_{\text{423 db 0}}$
3. Végül az $l = 24$-et egy 64 bites számként a sorozat végéhez illesztjük:
   01100001 01100010 01100011 1 $\underbrace{0000\ldots000}_{\text{423 db 0}}$ $\underbrace{0000\ldots11000}_{\text{64 bites egész}}$

# w[i]-k kiszámolása egy blokkon belül

Osszuk fel a bemenetet 512 bites blokkokra, a következő diákon egy blokk feldolgozását mutatjuk be
A blokkot osszuk fel 16 darab 32 bites szóra: blokk $= x_0, x_1, \ldots, x_{15}$
Legyen w[] egy 80 elemű tömb, a tömb értékei:

- w[i] $= x_i$, ha $0 \leq i \leq 15$
- w[i] = balra_forgatás(w[i-3] $\oplus$ w[i-8] $\oplus$ w[i-14] $\oplus$ w[i-16], 1), egyébként

Megjegyzés: Az első 16 w érték megadásakor vegyük figyelembe, hogy Intel kompatibilis processzoroknál little-endian számábrázolást alkalmazunk, ám az SHA-1 esetében big-endian-t használunk, tehát az első 16 w bájtjainak sorrendjét fel kell cserélnünk:
w[i] $= \{w_0, w_1, w_2, w_3\} \rightarrow \{w_3, w_2, w_1, w_0\}$

# Iterációk: Egy blokk feldolgozása

Szükségünk lesz 5 db 32 bites változóra: A, B, C, D, E
Ezen változók kezdeti értékei az első blokk feldolgozása előtt:

- A = $H_0$ = 67452301
- B = $H_1$ = EFCDAB89
- C = $H_2$ = 98BADCFE
- D = $H_3$ = 10325476
- E = $H_4$ = C3D2E1F0

- 4 darab 20 iterációból álló iterációt hajtunk végre
- Az iterációk során A, B, C, D és E értékét módosítjuk
- Az iterációk végén elvégezzük az alábbi műveletet:
  1. $H_0 = \mathrm{A} + H_0 \pmod{2^{32}}$
  2. $H_1 = \mathrm{B} + H_1 \pmod{2^{32}}$
  3. $H_2 = \mathrm{C} + H_2 \pmod{2^{32}}$
  4. $H_3 = \mathrm{D} + H_3 \pmod{2^{32}}$
  5. $H_4 = \mathrm{E} + H_4 \pmod{2^{32}}$

A következő blokknál ezen $H_i$ értékekből indulunk ki.

# A, B, C, D, E számolása

4*20 iterációt hajtunk végre, az 5 változót az alábbi módon változtatjuk meg, ha $i$ az aktuális iteráció sorszámát tartalmazza ($0 \leq i \leq 79$):

1. TEMP = balra_forgatás(A, 5) + f(B, C, D) + E + K + w[i]
2. E = D
3. D = C
4. C = balra_forgatás(B, 30)
5. B = A
6. A = TEMP

Az alábbi táblázatban az f függvény és a K konstans értékei láthatóak $i$ függvényében:

| $i$ | f | K |
|---|---|---|
| $0 \leq i \leq 19$ | $f = D \oplus ( B \wedge ( C \oplus D ) )$ | 5A827999 |
| $20 \leq i \leq 39$ | $f = B \oplus C \oplus D$ | 6ED9EBA1 |
| $40 \leq i \leq 59$ | $f = (B \wedge C) \vee (B \wedge D) \vee (C \wedge D)$ | 8F1BBCDC |
| $60 \leq i \leq 79$ | $f = B \oplus C \oplus D$ | CA62C1D6 |

# 1. Példa

Határozzuk meg a hash értékét a következő stringre:

`"The quick brown fox jumps over the lazy dog"`

- A bemenet hossza: 344 bit
- 103 darab 0-s bittel kell kibővíteni
- A kibővített bemenet hexadecimálisan:

  ```
  54686520 71756963 6b206272 6f776e20 666f7820 6a756d70
  73206f76 65722074 6865206c 617a7920 646f6780 00000000
  00000000 00000000 00000000 00000158
  ```

- w tömb első 16 értéke ez alapján:

| | | | |
|---|---|---|---|
| w[0] | 54686520 | w[8] | 6865206c |
| w[1] | 71756963 | w[9] | 617a7920 |
| w[2] | 6b206272 | w[10] | 646f6780 |
| w[3] | 6f776e20 | w[11] | 00000000 |
| w[4] | 666f7820 | w[12] | 00000000 |
| w[5] | 6a756d70 | w[13] | 00000000 |
| w[6] | 73206f76 | w[14] | 00000000 |
| w[7] | 65722074 | w[15] | 00000158 |

# 1. Példa: 0-15. iterációk

| i | w[i] | temp | a | b | c | d | e |
|---|---|---|---|---|---|---|---|
| 0 | 54686520 | F41CFDD3 | F41CFDD3 | 67452301 | 7BF36AE2 | 98BADCFE | 10325476 |
| 1 | 71756963 | 5BC5F0EE | 5BC5F0EE | F41CFDD3 | 59D148C0 | 7BF36AE2 | 98BADCFE |
| 2 | 6B206272 | 330F21B4 | 330F21B4 | 5BC5F0EE | FD073F74 | 59D148C0 | 7BF36AE2 |
| 3 | 6F776E20 | 00E6C185 | 00E6C185 | 330F21B4 | 96F17C3B | FD073F74 | 59D148C0 |
| 4 | 666F7820 | 159CA989 | 159CA989 | 00E6C185 | 0CC3C86D | 96F17C3B | FD073F74 |
| 5 | 6A756D70 | 0C6853DE | 0C6853DE | 159CA989 | 4039B061 | 0CC3C86D | 96F17C3B |
| 6 | 73206F76 | F9FAC170 | F9FAC170 | 0C6853DE | 45672A62 | 4039B061 | 0CC3C86D |
| 7 | 65722074 | 508232FC | 508232FC | F9FAC170 | 831A14F7 | 45672A62 | 4039B061 |
| 8 | 6865206C | 9886D462 | 9886D462 | 508232FC | 3E7EB05C | 831A14F7 | 45672A62 |
| 9 | 617A7920 | A558DDCD | A558DDCD | 9886D462 | 14208CBF | 3E7EB05C | 831A14F7 |
| 10 | 646F6780 | 23A05402 | 23A05402 | A558DDCD | A621B518 | 14208CBF | 3E7EB05C |
| 11 | 00000000 | C12C3F73 | C12C3F73 | 23A05402 | 69563773 | A621B518 | 14208CBF |
| 12 | 00000000 | 392CA9EA | 392CA9EA | C12C3F73 | 88E81500 | 69563773 | A621B518 |
| 13 | 00000000 | CEB380F8 | CEB380F8 | 392CA9EA | F04B0FDC | 88E81500 | 69563773 |
| 14 | 00000000 | 4B10EDED | 4B10EDED | CEB380F8 | 8E4B2A7A | F04B0FDC | 88E81500 |
| 15 | 00000158 | 03D35D16 | 03D35D16 | 4B10EDED | 33ACE03E | 8E4B2A7A | F04B0FDC |

# 1. Példa: 16-31. iterációk

| i | w[i] | temp | a | b | c | d | e |
|---|---|---|---|---|---|---|---|
| 16 | AE5A4E7C | FADF5CEF | FADF5CEF | 03D35D16 | 52C43B7B | 33ACE03E | 8E4B2A7A |
| 17 | FEF0FCC6 | 7696F812 | 7696F812 | FADF5CEF | 80F4D745 | 52C43B7B | 33ACE03E |
| 18 | D240F914 | B423CC8E | B423CC8E | 7696F812 | FEB7D73B | 80F4D745 | 52C43B7B |
| 19 | 56B09A59 | 7F67B89A | 7F67B89A | B423CC8E | 9DA5BE04 | FEB7D73B | 80F4D745 |
| 20 | D77FD721 | 8B775307 | 8B775307 | 7F67B89A | AD08F323 | 9DA5BE04 | FEB7D73B |
| 21 | BA8F6821 | E6D681AB | E6D681AB | 8B775307 | 9FD9EE26 | AD08F323 | 9DA5BE04 |
| 22 | 9BEBAA86 | 3CE1D7A9 | 3CE1D7A9 | E6D681AB | E2DDD4C1 | 9FD9EE26 | AD08F323 |
| 23 | A6EF1E5B | FADFAD92 | FADFAD92 | 3CE1D7A9 | F9B5A06A | E2DDD4C1 | 9FD9EE26 |
| 24 | 31BEC362 | C3F1F28A | C3F1F28A | FADFAD92 | 4F3875EA | F9B5A06A | E2DDD4C1 |
| 25 | 08C25EC0 | 250AE88C | 250AE88C | C3F1F28A | BEB7EB64 | 4F3875EA | F9B5A06A |
| 26 | 2181019E | 5DEC0B31 | 5DEC0B31 | 250AE88C | B0FC7CA2 | BEB7EB64 | 4F3875EA |
| 27 | CE1CB276 | 74F1F976 | 74F1F976 | 5DEC0B31 | 0942BA23 | B0FC7CA2 | BEB7EB64 |
| 28 | BF7B13C3 | 6F9EE746 | 6F9EE746 | 74F1F976 | 577B02CC | 0942BA23 | B0FC7CA2 |
| 29 | 361CD1CF | 74986478 | 74986478 | 6F9EE746 | 9D3C7E5D | 577B02CC | 0942BA23 |
| 30 | F75AAD19 | A85D7DC2 | A85D7DC2 | 74986478 | 9BE7B9D1 | 9D3C7E5D | 577B02CC |
| 31 | CEC9E00D | 13122AC3 | 13122AC3 | A85D7DC2 | 1D26191E | 9BE7B9D1 | 9D3C7E5D |

# 1. Példa: 32-47. iterációk

| i | w[i] | temp | a | b | c | d | e |
|---|---|---|---|---|---|---|---|
| 32 | F7714B8A | 9469EAF7 | 9469EAF7 | 13122AC3 | AA175F70 | 1D26191E | 9BE7B9D1 |
| 33 | AFB12A8C | EBD39B9D | EBD39B9D | 9469EAF7 | C4C48AB0 | AA175F70 | 1D26191E |
| 34 | D4EF9F4D | D61D5700 | D61D5700 | EBD39B9D | E51A7ABD | C4C48AB0 | AA175F70 |
| 35 | AAA41709 | 514DADC4 | 514DADC4 | D61D5700 | 7AF4E6E7 | E51A7ABD | C4C48AB0 |
| 36 | B8BC89D0 | 60048405 | 60048405 | 514DADC4 | 358755C0 | 7AF4E6E7 | E51A7ABD |
| 37 | FD2671F1 | 6FE977DE | 6FE977DE | 60048405 | 14536B71 | 358755C0 | 7AF4E6E7 |
| 38 | EF57A7E9 | 182630F2 | 182630F2 | 6FE977DE | 58012101 | 14536B71 | 358755C0 |
| 39 | B0B0528D | 7D92EFDF | 7D92EFDF | 182630F2 | 9BFA5DF7 | 58012101 | 14536B71 |
| 40 | 34D1F10E | A2C1473D | A2C1473D | 7D92EFDF | 86098C3C | 9BFA5DF7 | 58012101 |
| 41 | 0C70C3A7 | EB515737 | EB515737 | A2C1473D | DF64BBF7 | 86098C3C | 9BFA5DF7 |
| 42 | F54BBF3B | 10CE5048 | 10CE5048 | EB515737 | 68B051CF | DF64BBF7 | 86098C3C |
| 43 | CCEB0B7C | E74AB18D | E74AB18D | 10CE5048 | FAD455CD | 68B051CF | DF64BBF7 |
| 44 | F9DBE95B | CA86E5B7 | CA86E5B7 | E74AB18D | 04339412 | FAD455CD | 68B051CF |
| 45 | E171FE11 | 106D5942 | 106D5942 | CA86E5B7 | 79D2AC63 | 04339412 | FAD455CD |
| 46 | 472E940C | 275C732A | 275C732A | 106D5942 | F2A1B96D | 79D2AC63 | 04339412 |
| 47 | 5026E2AE | 3FE65243 | 3FE65243 | 275C732A | 841B5650 | F2A1B96D | 79D2AC63 |

# 1. Példa: 48-63. iterációk

| i | w[i] | temp | a | b | c | d | e |
|---|---|---|---|---|---|---|---|
| 48 | EC7DB7B1 | 984FDCBF | 984FDCBF | 3FE65243 | 89D71CCA | 841B5650 | F2A1B96D |
| 49 | 9C96D45C | B62738DA | B62738DA | 984FDCBF | CFF99490 | 89D71CCA | 841B5650 |
| 50 | 927C9611 | F47A612D | F47A612D | B62738DA | E613F72F | CFF99490 | 89D71CCA |
| 51 | EE29B46A | 7C9C6868 | 7C9C6868 | F47A612D | AD89CE36 | E613F72F | CFF99490 |
| 52 | 654C267C | 3C0A6C26 | 3C0A6C26 | 7C9C6868 | 7D1E984B | AD89CE36 | E613F72F |
| 53 | 7D3696F8 | F1509834 | F1509834 | 3C0A6C26 | 1F271A1A | 7D1E984B | AD89CE36 |
| 54 | E502ED02 | 88C996BC | 88C996BC | F1509834 | 8F029B09 | 1F271A1A | 7D1E984B |
| 55 | 1354ABF1 | D7C472C1 | D7C472C1 | 88C996BC | 3C54260D | 8F029B09 | 1F271A1A |
| 56 | A1A2DEF8 | D4B4A435 | D4B4A435 | D7C472C1 | 223265AF | 3C54260D | 8F029B09 |
| 57 | 721FE30B | 5D272837 | 5D272837 | D4B4A435 | 75F11CB0 | 223265AF | 3C54260D |
| 58 | 1B70D701 | 0075E58A | 0075E58A | 5D272837 | 752D290D | 75F11CB0 | 223265AF |
| 59 | C4233FFE | F9533BFE | F9533BFE | 0075E58A | D749CA0D | 752D290D | 75F11CB0 |
| 60 | 534D7041 | 6019D530 | 6019D530 | F9533BFE | 801D7962 | D749CA0D | 752D290D |
| 61 | AE22BA8D | 9EF4D40D | 9EF4D40D | 6019D530 | BE54CEFF | 801D7962 | D749CA0D |
| 62 | 14E5E283 | F37D52C6 | F37D52C6 | 9EF4D40D | 1806754C | BE54CEFF | 801D7962 |
| 63 | 1953DA85 | 0C24DE59 | 0C24DE59 | F37D52C6 | 67BD3503 | 1806754C | BE54CEFF |

# 1. Példa: 64-79. iterációk

| i | w[i] | temp | a | b | c | d | e |
|---|---|---|---|---|---|---|---|
| 64 | E3028BAA | 7D1BFA29 | 7D1BFA29 | 0C24DE59 | BCDF54B1 | 67BD3503 | 1806754C |
| 65 | 288AC37C | 85B9FFB8 | 85B9FFB8 | 7D1BFA29 | 43093796 | BCDF54B1 | 67BD3503 |
| 66 | EA277BD3 | 565502CA | 565502CA | 85B9FFB8 | 5F46FE8A | 43093796 | BCDF54B1 |
| 67 | 687D2D8D | 5455D402 | 5455D402 | 565502CA | 216E7FEE | 5F46FE8A | 43093796 |
| 68 | F712F087 | B7B6EDEB | B7B6EDEB | 5455D402 | 959540B2 | 216E7FEE | 5F46FE8A |
| 69 | 54CFF8AE | 560661E2 | 560661E2 | B7B6EDEB | 95157500 | 959540B2 | 216E7FEE |
| 70 | 7071F9E8 | D446504F | D446504F | 560661E2 | EDEDBB7A | 95157500 | 959540B2 |
| 71 | 1E14C5F1 | 35D5820B | 35D5820B | D446504F | 95819878 | EDEDBB7A | 95157500 |
| 72 | 1A3EF5FA | E091E183 | E091E183 | 35D5820B | F5119413 | 95819878 | EDEDBB7A |
| 73 | DD8FCCC3 | FD6208EF | FD6208EF | E091E183 | CD756082 | F5119413 | 95819878 |
| 74 | 781C32C5 | 5D36C024 | 5D36C024 | FD6208EF | F8247860 | CD756082 | F5119413 |
| 75 | 3084BA08 | 5F042489 | 5F042489 | 5D36C024 | FF58823B | F8247860 | CD756082 |
| 76 | DA6B5D0C | AD124B0E | AD124B0E | 5F042489 | 174DB009 | FF58823B | F8247860 |
| 77 | 374554C7 | 5327078D | 5327078D | AD124B0E | 57C10922 | 174DB009 | FF58823B |
| 78 | 6E245593 | 8A5F7D73 | 8A5F7D73 | 5327078D | AB4492C3 | 57C10922 | 174DB009 |
| 79 | EB4D0209 | C88FBEC5 | C88FBEC5 | 8A5F7D73 | 54C9C1E3 | AB4492C3 | 57C10922 |

A keresett hash érték:

"2FD4E1C6 7A2D28FC ED849EE1 BB76E739 1B93EB12"

# 2. Példa

Határozzuk meg az üres string hash értékét!

- A bemenet hossza: 0 bit
- 447 darab 0-s bittel kell kibővíteni, majd utána fűzzük a 0 hosszt reprezentáló 64 bitet
- A kibővített bemenet hexadecimálisan:
  80000000 00000000 00000000 00000000 00000000 00000000
  00000000 00000000 00000000 00000000 00000000 00000000
  00000000 00000000 00000000 00000000
- w tömb első 16 értéke ez alapján:

| | | | |
|---|---|---|---|
| w[0] | 80000000 | w[8] | 00000000 |
| w[1] | 00000000 | w[9] | 00000000 |
| w[2] | 00000000 | w[10] | 00000000 |
| w[3] | 00000000 | w[11] | 00000000 |
| w[4] | 00000000 | w[12] | 00000000 |
| w[5] | 00000000 | w[13] | 00000000 |
| w[6] | 00000000 | w[14] | 00000000 |
| w[7] | 00000000 | w[15] | 00000000 |

# 2. Példa: 0-15. iterációk

| i | w[i] | temp | a | b | c | d | e |
|---|---|---|---|---|---|---|---|
| 0 | 80000000 | 1FB498B3 | 1FB498B3 | 67452301 | 7BF36AE2 | 98BADCFE | 10325476 |
| 1 | 00000000 | 5D43E370 | 5D43E370 | 1FB498B3 | 59D148C0 | 7BF36AE2 | 98BADCFE |
| 2 | 00000000 | 158D2F62 | 158D2F62 | 5D43E370 | C7ED262C | 59D148C0 | 7BF36AE2 |
| 3 | 00000000 | CDECFB5D | CDECFB5D | 158D2F62 | 1750F8DC | C7ED262C | 59D148C0 |
| 4 | 00000000 | 4953565E | 4953565E | CDECFB5D | 85634BD8 | 1750F8DC | C7ED262C |
| 5 | 00000000 | E44AB766 | E44AB766 | 4953565E | 737B3ED7 | 85634BD8 | 1750F8DC |
| 6 | 00000000 | C09D7F27 | C09D7F27 | E44AB766 | 9254D597 | 737B3ED7 | 85634BD8 |
| 7 | 00000000 | 87074800 | 87074800 | C09D7F27 | B912ADD9 | 9254D597 | 737B3ED7 |
| 8 | 00000000 | 41376611 | 41376611 | 87074800 | F0275FC9 | B912ADD9 | 9254D597 |
| 9 | 00000000 | CBDBFF31 | CBDBFF31 | 41376611 | 21C1D200 | F0275FC9 | B912ADD9 |
| 10 | 00000000 | 40166973 | 40166973 | CBDBFF31 | 504DD984 | 21C1D200 | F0275FC9 |
| 11 | 00000000 | ADC0E0CA | ADC0E0CA | 40166973 | 72F6FFCC | 504DD984 | 21C1D200 |
| 12 | 00000000 | 84C05EB2 | 84C05EB2 | ADC0E0CA | D0059A5C | 72F6FFCC | 504DD984 |
| 13 | 00000000 | 1512C8B9 | 1512C8B9 | 84C05EB2 | AB703832 | D0059A5C | 72F6FFCC |
| 14 | 00000000 | 40182905 | 40182905 | 1512C8B9 | A13017AC | AB703832 | D0059A5C |
| 15 | 00000000 | D8FD6547 | D8FD6547 | 40182905 | 4544B22E | A13017AC | AB703832 |

# 2. Példa: 16-31. iterációk

| i | w[i] | temp | a | b | c | d | e |
|---|---|---|---|---|---|---|---|
| 16 | 00000001 | 06BF9173 | 06BF9173 | D8FD6547 | 50060A41 | 4544B22E | A13017AC |
| 17 | 00000000 | 28A9520E | 28A9520E | 06BF9173 | F63F5951 | 50060A41 | 4544B22E |
| 18 | 00000000 | 0B3088DD | 0B3088DD | 28A9520E | C1AFE45C | F63F5951 | 50060A41 |
| 19 | 00000002 | E758E8DA | E758E8DA | 0B3088DD | 8A2A5483 | C1AFE45C | F63F5951 |
| 20 | 00000000 | 90EB9850 | 90EB9850 | E758E8DA | 42CC2237 | 8A2A5483 | C1AFE45C |
| 21 | 00000000 | 7DBB787D | 7DBB787D | 90EB9850 | B9D63A36 | 42CC2237 | 8A2A5483 |
| 22 | 00000004 | 1C64D028 | 1C64D028 | 7DBB787D | 243AE614 | B9D63A36 | 42CC2237 |
| 23 | 00000000 | 1E97B73A | 1E97B73A | 1C64D028 | 5F6EDE1F | 243AE614 | B9D63A36 |
| 24 | 00000002 | 62D7F53F | 62D7F53F | 1E97B73A | 0719340A | 5F6EDE1F | 243AE614 |
| 25 | 00000008 | 34F3D6D8 | 34F3D6D8 | 62D7F53F | 87A5EDCE | 0719340A | 5F6EDE1F |
| 26 | 00000000 | 4F2ED1C1 | 4F2ED1C1 | 34F3D6D8 | D8B5FD4F | 87A5EDCE | 0719340A |
| 27 | 00000000 | C7B11E2D | C7B11E2D | 4F2ED1C1 | 0D3CF5B6 | D8B5FD4F | 87A5EDCE |
| 28 | 00000010 | 874B786F | 874B786F | C7B11E2D | 53CBB470 | 0D3CF5B6 | D8B5FD4F |
| 29 | 00000000 | CA4556CB | CA4556CB | 874B786F | 71EC478B | 53CBB470 | 0D3CF5B6 |
| 30 | 0000000A | 6A2E466E | 6A2E466E | CA4556CB | E1D2DE1B | 71EC478B | 53CBB470 |
| 31 | 00000020 | 62EA3D59 | 62EA3D59 | 6A2E466E | F29155B2 | E1D2DE1B | 71EC478B |

# 2. Példa: 32-47. iterációk

| i | w[i] | temp | a | b | c | d | e |
|---|---|---|---|---|---|---|---|
| 32 | 00000006 | B77BAC25 | B77BAC25 | 62EA3D59 | 9A8B919B | F29155B2 | E1D2DE1B |
| 33 | 00000000 | 4B1347E2 | 4B1347E2 | B77BAC25 | 58BA8F56 | 9A8B919B | F29155B2 |
| 34 | 00000040 | 391EF0C4 | 391EF0C4 | 4B1347E2 | 6DDEEB09 | 58BA8F56 | 9A8B919B |
| 35 | 00000008 | ABBAB988 | ABBAB988 | 391EF0C4 | 92C4D1F8 | 6DDEEB09 | 58BA8F56 |
| 36 | 00000028 | 04F07669 | 04F07669 | ABBAB988 | 0E47BC31 | 92C4D1F8 | 6DDEEB09 |
| 37 | 00000080 | B201788B | B201788B | 04F07669 | 2AEEAE62 | 0E47BC31 | 92C4D1F8 |
| 38 | 00000008 | 62273351 | 62273351 | B201788B | 413C1D9A | 2AEEAE62 | 0E47BC31 |
| 39 | 00000000 | 9BDBDD71 | 9BDBDD71 | 62273351 | EC805E22 | 413C1D9A | 2AEEAE62 |
| 40 | 00000108 | 95AA398B | 95AA398B | 9BDBDD71 | 5889CCD4 | EC805E22 | 413C1D9A |
| 41 | 00000000 | 5E28E858 | 5E28E858 | 95AA398B | 66F6F75C | 5889CCD4 | EC805E22 |
| 42 | 000000A0 | 95642485 | 95642485 | 5E28E858 | E56A8E62 | 66F6F75C | 5889CCD4 |
| 43 | 00000200 | FA950ABA | FA950ABA | 95642485 | 178A3A16 | E56A8E62 | 66F6F75C |
| 44 | 00000064 | DE1E3A01 | DE1E3A01 | FA950ABA | 65590921 | 178A3A16 | E56A8E62 |
| 45 | 00000000 | AFE695AB | AFE695AB | DE1E3A01 | BEA542AE | 65590921 | 178A3A16 |
| 46 | 00000408 | A195BA90 | A195BA90 | AFE695AB | 77878E80 | BEA542AE | 65590921 |
| 47 | 00000088 | E6D39F43 | E6D39F43 | A195BA90 | EBF9A56A | 77878E80 | BEA542AE |

# 2. Példa: 48-63. iterációk

| i | w[i] | temp | a | b | c | d | e |
|---|---|---|---|---|---|---|---|
| 48 | 0000029C | 0BCA9922 | 0BCA9922 | E6D39F43 | 28656EA4 | EBF9A56A | 77878E80 |
| 49 | 00000800 | 6AE826FF | 6AE826FF | 0BCA9922 | F9B4E7D0 | 28656EA4 | EBF9A56A |
| 50 | 00000080 | 01FF3253 | 01FF3253 | 6AE826FF | 82F2A648 | F9B4E7D0 | 28656EA4 |
| 51 | 00000028 | E2581CE0 | E2581CE0 | 01FF3253 | DABA09BF | 82F2A648 | F9B4E7D0 |
| 52 | 00001088 | 56CE73AB | 56CE73AB | E2581CE0 | C07FCC94 | DABA09BF | 82F2A648 |
| 53 | 00000000 | AE56E542 | AE56E542 | 56CE73AB | 38960738 | C07FCC94 | DABA09BF |
| 54 | 00000A40 | 8590C0E8 | 8590C0E8 | AE56E542 | D5B39CEA | 38960738 | C07FCC94 |
| 55 | 00002000 | BE4A4BEA | BE4A4BEA | 8590C0E8 | AB95B950 | D5B39CEA | 38960738 |
| 56 | 00000668 | 168CE0BB | 168CE0BB | BE4A4BEA | 2164303A | AB95B950 | D5B39CEA |
| 57 | 00000080 | E1AFAB22 | E1AFAB22 | 168CE0BB | AF9292FA | 2164303A | AB95B950 |
| 58 | 00004088 | 982BCBCA | 982BCBCA | E1AFAB22 | C5A3382E | AF9292FA | 2164303A |
| 59 | 00000880 | 9B9D2913 | 9B9D2913 | 982BCBCA | B86BEAC8 | C5A3382E | AF9292FA |
| 60 | 000028C8 | D37DB937 | D37DB937 | 9B9D2913 | A60AF2F2 | B86BEAC8 | C5A3382E |
| 61 | 00008000 | 85B9D227 | 85B9D227 | D37DB937 | E6E74A44 | A60AF2F2 | B86BEAC8 |
| 62 | 000008A8 | CD98FBB7 | CD98FBB7 | 85B9D227 | F4DF6E4D | E6E74A44 | A60AF2F2 |
| 63 | 00000080 | BB0F226F | BB0F226F | CD98FBB7 | E16E7489 | F4DF6E4D | E6E74A44 |

# 2. Példa: 64-79. iterációk

| i | w[i] | temp | a | b | c | d | e |
|---|---|---|---|---|---|---|---|
| 64 | 000108E8 | EB59446C | EB59446C | BB0F226F | F3663EED | E16E7489 | F4DF6E4D |
| 65 | 00000000 | D37225CB | D37225CB | EB59446C | EEC3C89B | F3663EED | E16E7489 |
| 66 | 0000A000 | 111341F3 | 111341F3 | D37225CB | 3AD6511B | EEC3C89B | F3663EED |
| 67 | 00020080 | E79AFBF0 | E79AFBF0 | 111341F3 | F4DC8972 | 3AD6511B | EEC3C89B |
| 68 | 00006400 | 8BA00627 | 8BA00627 | E79AFBF0 | C444D07C | F4DC8972 | 3AD6511B |
| 69 | 00000000 | 503C7AE0 | 503C7AE0 | 8BA00627 | 39E6BEFC | C444D07C | F4DC8972 |
| 70 | 00040800 | 3CD517F9 | 3CD517F9 | 503C7AE0 | E2E80189 | 39E6BEFC | C444D07C |
| 71 | 00008800 | B47DDF0E | B47DDF0E | 3CD517F9 | 140F1EB8 | E2E80189 | 39E6BEFC |
| 72 | 00029C10 | 5E3A0780 | 5E3A0780 | B47DDF0E | 4F3545FE | 140F1EB8 | E2E80189 |
| 73 | 00080000 | 63DB37B2 | 63DB37B2 | 5E3A0780 | AD1F77C3 | 4F3545FE | 140F1EB8 |
| 74 | 00008080 | 15E98D17 | 15E98D17 | 63DB37B2 | 178E81E0 | AD1F77C3 | 4F3545FE |
| 75 | 00002820 | B0149467 | B0149467 | 15E98D17 | 98F6CDEC | 178E81E0 | AD1F77C3 |
| 76 | 001088C0 | 14B7106A | 14B7106A | B0149467 | C57A6345 | 98F6CDEC | 178E81E0 |
| 77 | 00000000 | 666B8BC6 | 666B8BC6 | 14B7106A | EC052519 | C57A6345 | 98F6CDEC |
| 78 | 000A40C0 | 6E9D9F84 | 6E9D9F84 | 666B8BC6 | 852DC41A | EC052519 | C57A6345 |
| 79 | 00200080 | 72F480ED | 72F480ED | 6E9D9F84 | 999AE2F1 | 852DC41A | EC052519 |

A keresett hash érték:

"DA39A3EE 5E6B4B0D 3255BFEF 95601890 AFD80709"