

Hálózati ismeretek

Titkosítási technológiák áttekintése



Hash algoritmusok

Hash algoritmusok

- ▶ Bemenet: végtelen hosszú adat.
- ▶ Kimenet: fix hosszúságú adat.
- ▶ A képzési eljárás egyirányú, a tényleges adat nem nyerhető ki az ellenőrző összegből.
- ▶ Egy ellenőrző összeg több bemeneti kombinációhoz is tartozhat.



Hash algoritmusok használata

- ▶ Jelszavak titkosítására

- ▶ Regisztrációkor:

- ▶ Jelszó hash eltárolása adatbázisban

- ▶ Belépéskor:

- ▶ Beírt jelszó hash kiszámítása, majd összehasonlítás adatbázisban tárolt hash-el.
 - ▶ Ha a 2 megegyezik, akkor minden ok, ellenkező esetben hiba.



Hash algoritmusok használata

▶ Fájlok aláírása

- ▶ Leginkább futtatható fájlok esetén használják más titkosítási technológiákkal együtt
- ▶ Aláírt .exe és .dll fájlról meg tudja mondani a rendszer, hogy az előállítása óta megsérült - e, vagy módosította - e egy vírus.
- ▶ Kevés az ilyen .exe és .dll



Hash algoritmusok használata

- ▶ **Adatátvitel esetén integritás tesztelésre**
 - ▶ Ma már leginkább protokollsinten, de ettől függetlenül is alkalmazható
 - ▶ Küldő küldi a fájlt, és a Hash adatot
 - ▶ Fogadó a fogadott fájlra szintén kiszámolja a hash-t, majd a fogadott és kapott hash-t bitenként összehasonlítja.
 - ▶ Ha megegyeznek, akkor az átvitelben nem volt probléma. Ha igen, akkor újraküldés lesz.



Hash algoritmusok problémái

▶ Sebesség

- ▶ Minél komplexebb a funkció, annál lassabb lesz az algoritmus.

▶ Átfedések

- ▶ Több bemeneti adatnak is lehet azonos a hash-e definícióból adódóan.
- ▶ Cél: Minimalizálni ennek az esélyét. Kézenfekvő eszköz a kimeneti bitek számának növelése.

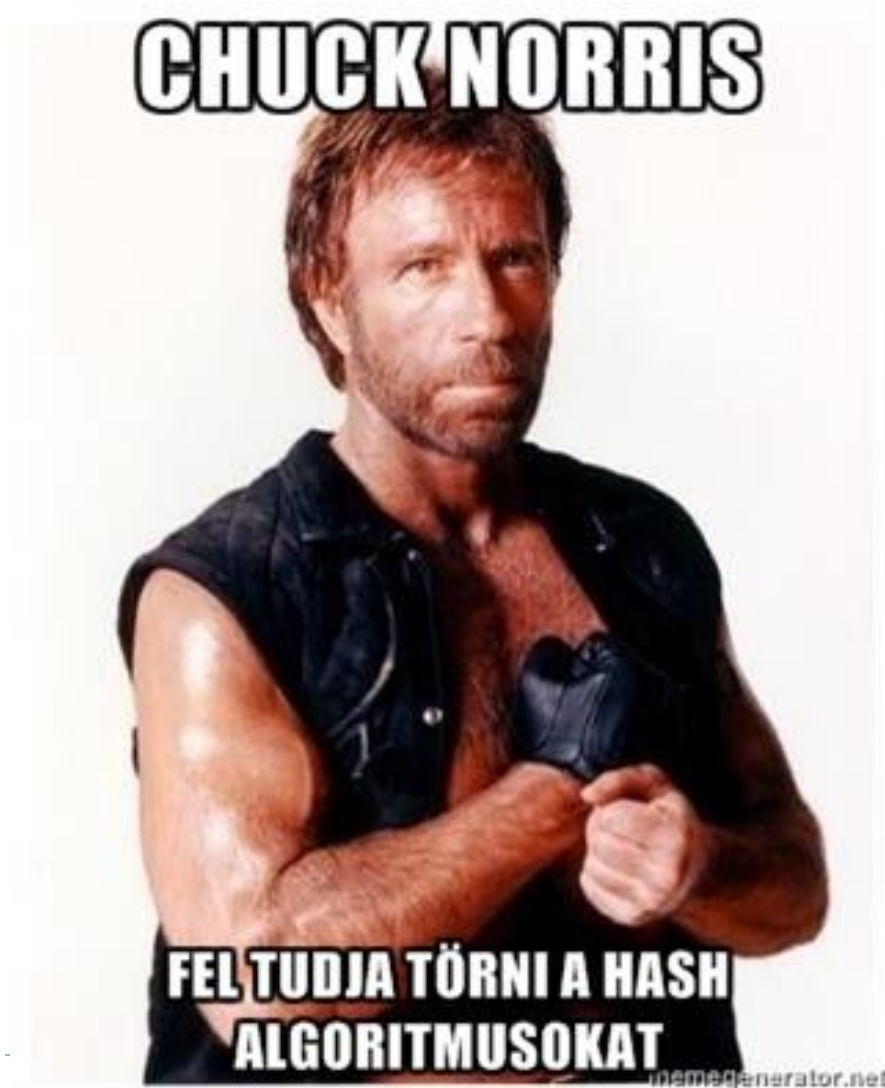


Hash algoritmusok problémái

- ▶ Idővel minden algoritmus elavul, hiszen egyre gyorsabbak a gépek
- ▶ Gyorsabb gépekkel könnyebb megtalálni azokat a bemeneti kombinációkat, amelyek ugyanazt a kimenetet produkálják.
- ▶ Ha ez nem lenne elég, van egy ennél sokkal komolyabb gond is...



Hash algoritmusok problémái



Hash algoritmusok

- ▶ Kb. annyi van, mint égen a csillag.
- ▶ Három darab algoritmusról lesz ma szó:
 - ▶ CRC32
 - ▶ MD5
 - ▶ SHA család



CRC-32

- ▶ A CRC hash algoritmusok családjának 32 bites változata.
- ▶ Őse, a CRC-8 története 1961-ig nyúlik vissza.
- ▶ Ezen algoritmusok azért jók, mivel könnyű őket hardveresen és szoftveresen is implementálni.
- ▶ Feltalálója W. Wesley Peterson.
- ▶ A CRC32 1975-ben jelent meg.



CRC-32

- ▶ Elsősorban átviteli hibák detektálására használják.
- ▶ Más célokra a kevés bitszám miatt nem alkalmas.
- ▶ Számos helyen alkalmazott:
 - ▶ Ethernet, SATA, MPEG2, ZIP, GZIP, PNG, Stb...
- ▶ Elődei és utódai még több helyen vannak alkalmazva.



CRC32 működése

- ▶ Biteltolások és XOR műveletek sorozatával dolgozik.
- ▶ Fontos eleme az alapszám, amely mindig egy bittel hosszabb, mint az algoritmus bitjeinek száma.
- ▶ Algoritmusonként eltér, kiválasztásánál fontos tényező az adatcsomagok mérete, hogy minimalizálják a hibákat.
- ▶ Az alapszám és az adaton elvégzett műveletek sorozataként áll elő a hash kód.



Egy C++ CRC-32 implementáció

```
▶ //összesen 17 sor ☺
▶ unsigned int CRC32_function(const unsigned char *buf, unsigned long len)
▶ {
▶     unsigned long crc_table[256];
▶     unsigned long crc;
▶     for (int i = 0; i < 256; i++) {
▶         crc = i;
▶         for (int j = 0; j < 8; j++) {
▶             crc = crc & 1 ? (crc >> 1) ^ 0xEDB88320UL : crc >> 1;
▶         }
▶         crc_table[i] = crc;
▶     }
▶     crc = 0xFFFFFFFFUL;
▶     while (len--) {
▶         crc = crc_table[(crc ^ *buf++) & 0xFF] ^ (crc >> 8);
▶     }
▶     return crc ^ 0xFFFFFFFFUL;
▶ }
```



MD5

- ▶ RFC 1321
- ▶ 1991-től
- ▶ 2008 óta titkosítási célokra nem ajánlott, mivel több súlyos hibát találtak benne.
- ▶ 128 bites hash = 16 byte
- ▶ Általában 32db hexa karakterként kifejezve



Az MD5 működése

- ▶ Első lépésben bemeneti adat 512 bites blokkokra bontása
- ▶ Amennyiben az adat nem osztható 512-vel, akkor kiegészíti a végét 0-val.
 - ▶ Utolsó 64 bit, ha lehetősége van rá, a bemeneti adat bitjeinek száma előjel nélküli egész számként.
- ▶ Fő algoritmus: 128 biten dolgozik 4x32 bit formában.



Az MD5 működése

- ▶ 4db 32 bites szám (A, B, C, D) előre meghatározott konstansról indul.
- ▶ Az adat feldolgozása 512 bites blokkokban történik, ami tovább van bontva 32 bitre.
- ▶ A feldolgozott adat módosítja a 4db 32 bites szám értékét.
- ▶ 4db módosító függvényt használt, amelyek 128 bit feldolgozása után váltakoznak.



Az MD5 működése

► Módosító függvények:

$$F(X, Y, Z) = (X \wedge Y) \vee (\neg X \wedge Z)$$

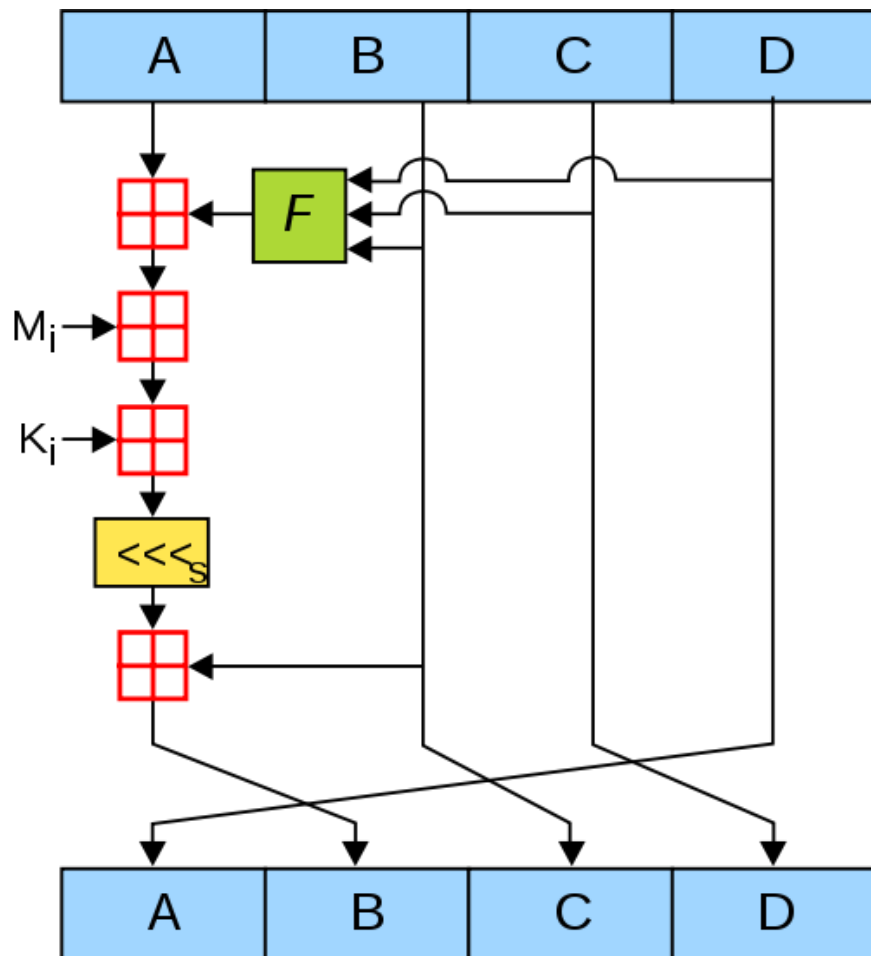
$$G(X, Y, Z) = (X \wedge Z) \vee (Y \wedge \neg Z)$$

$$H(X, Y, Z) = X \oplus Y \oplus Z$$

$$I(X, Y, Z) = Y \oplus (X \vee \neg Z)$$



Az MD5 működése



Mi: Adat 32 bitje

Ki: Konstans, ami a következő formában áll elő:

for i **from** 0 **to** 63

$k[i] := \text{floor}(\text{abs}(\sin(i + 1)) \times (2^{\text{pow } 32}))$

end for

$\lll s$: biteltolás balra s bittel. S értéke körönként változik

Egy 512 bites blokk feldolgozása 16 ilyen körből áll.

MD5 „feltörése”

- ▶ MD5 esetén ez 2008-ban sikerült.
- ▶ 128 bit esetén 2^{128} egyedi hash készíthető.
- ▶ Ez jó közelítéssel: $3,402 \times 10^{38}$
- ▶ A bolygó teljes levegőjében nincs annyi molekula, mint amennyinek egyedi MD5 hash-t tudnánk adni.



Felmerülhet a kérdés...



MD5 „feltörése”

- ▶ Valószínűségszámítás alapján 2 bemeneti kód azonos hash kimenetének az esélye $2^{n/2}$, ha mindkét bemenet egyforma esélyekkel indul.
- ▶ 128 bit esetén tehát legalább 2^{64} bemeneti kódot kell átnéznünk, hogy legyen legalább 2 olyan bemenetem, ami azonos kimenetet produkál.



Minimum mennyiségű hash, 2 azonos bemenet találásához

Bitek	Lehetséges kimenetek száma	Kívánt valószínűség a 2 bemenet egyezésére									
		10^{-18}	10^{-15}	10^{-12}	10^{-9}	10^{-6}	0.1%	1%	25%	50%	75%
16	6.6×10^4	2	2	2	2	2	11	36	1.9×10^2	3.0×10^2	4.3×10^2
32	4.3×10^9	2	2	2	2.9	93	2.9×10^3	9.3×10^3	5.0×10^4	7.7×10^4	1.1×10^5
64	1.8×10^{19}	6.1	1.9×10^2	6.1×10^3	1.9×10^5	6.1×10^6	1.9×10^8	6.1×10^8	3.3×10^9	5.1×10^9	7.2×10^9
128	3.4×10^{38}	2.6×10^{10}	8.2×10^{11}	2.6×10^{13}	8.2×10^{14}	2.6×10^{16}	8.3×10^{17}	2.6×10^{18}	1.4×10^{19}	2.2×10^{19}	3.1×10^{19}
256	1.2×10^{77}	4.8×10^{29}	1.5×10^{31}	4.8×10^{32}	1.5×10^{34}	4.8×10^{35}	1.5×10^{37}	4.8×10^{37}	2.6×10^{38}	4.0×10^{38}	5.7×10^{38}
384	3.9×10^{115}	8.9×10^{48}	2.8×10^{50}	8.9×10^{51}	2.8×10^{53}	8.9×10^{54}	2.8×10^{56}	8.9×10^{56}	4.8×10^{57}	7.4×10^{57}	1.0×10^{58}
512	1.3×10^{154}	1.6×10^{68}	5.2×10^{69}	1.6×10^{71}	5.2×10^{72}	1.6×10^{74}	5.2×10^{75}	1.6×10^{76}	8.8×10^{76}	1.4×10^{77}	1.9×10^{77}



MD5 feltörése

- ▶ CPU-k jelenleg is lassúak ezen feladatra
- ▶ Megoldás: GPU használata, mivel azonos számításokat kell elvégezni, sokszor.
- ▶ Nvidia GF 8800 ultra kártya másodpercenként 200 millió hash-t tud generálni.
- ▶ Ilyen sebesség mellett is 2 azonos bemenet találása legalább $5,75 \cdot 10^{12}$ év lenne
- ▶ A helyzet tovább romlott, hiszen szinte minden mai számítógép alkalmas a feladatra.



MD5 feltörése

- ▶ Amiért mégis lehetséges a dolog: ismétlődő mintát találtak a konstansokban, amelyek lépésenként alkalmazva vannak.
- ▶ 2^{21} hash szükséges csak 2 azonos bemenet találásához (2 097 152 próbálkozás csupán).
- ▶ További gond az úgynevezett szivárvány tábla (Rainbow table)
- ▶ Ez rengeteg gyakran használt jelszó MD5 értékét tartalmazza visszakereshetőség miatt.



SHA1

- ▶ Amerikai Nemzetbiztonsági Hivatal tervezte
- ▶ SHA: Secure Hash Algorithm
- ▶ Elődje az SHA0, amely széles körben sosem terjedt el, mivel matematikailag gyengének bizonyult.
- ▶ 160 bites algoritmus, szintén 512 bites blokkokban dolgozik



SHA1

- ▶ 2005-ben váltotta le az SHA2
- ▶ Váltás oka: létezik olyan algoritmus, amely segítségével 2^{80} próbálkozásnál kevesebbrel található két azonos bemenet.
- ▶ 2^{51} próbálkozással található azonos bemenet.
- ▶ Elméleti törés, hiszen 1 valódi ütközés találásának ideje: $8,69 \cdot 10^{35}$ év (200 millió/s sebesség mellett)



SHA Család további tagjai

Algoritmus	Kimenet hossz (bit)	Blokk méret (bit)	Max. üzenet hossz. (bit)	Szó méret (bit)	Körök száma	Műveletek
SHA-0						
SHA-1	160	512	$2^{64} - 1$	32	80	add, and, or, xor, rotate, mod
SHA-2	<i>SHA- 256/224</i>	256/224	$2^{64} - 1$	32	64	add, and, or, xor, shift, rotate, mod
	<i>SHA- 512/384</i>	512/384	$2^{128} - 1$	64	80	





A DES Algorithmus

A DES

- ▶ 4. Generációs titkosítási algoritmusok őse
- ▶ DES = Data Encryption Standard
- ▶ 1976-ban állt munkába
- ▶ 1997-ben sikerült először feltörni
- ▶ 2001-ben váltotta le az AES (Advanced Encryption Standard)
- ▶ Tervezésében részt vett az NSA

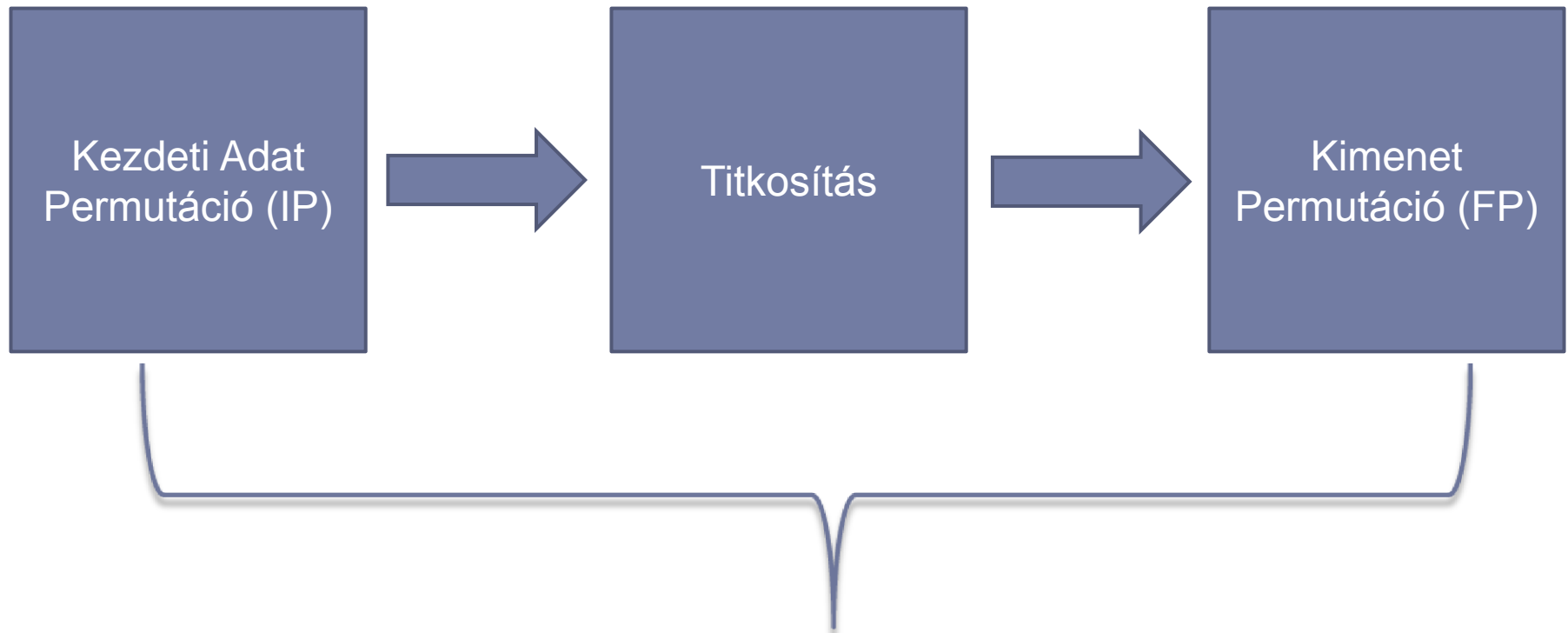


A DES

- ▶ 64 bites titkosítás.
- ▶ 64 bites blokkokban dolgozik egy 64 bites kulcs segítségével
- ▶ A kulcs valójában „csak” 56 bites, mivel a kulcs minden bájtjának utolsó bitje paritás bit.
- ▶ Teljesen nyílt szabvány, így az algoritmust mindenki megismerheti
- ▶ Tehát az adatok védelme csak a kulcs bonyolultságától függ.



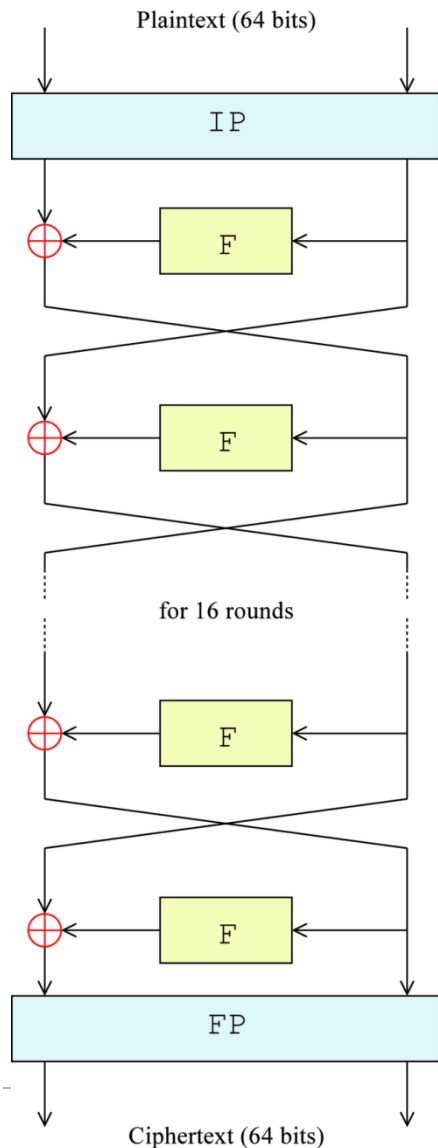
A Des működése



Egymás hatását kioltják, titkosításban nem sok szerepe van.
Optimalizációs céllal került az algoritmusba az 1970-es évek gépei miatt

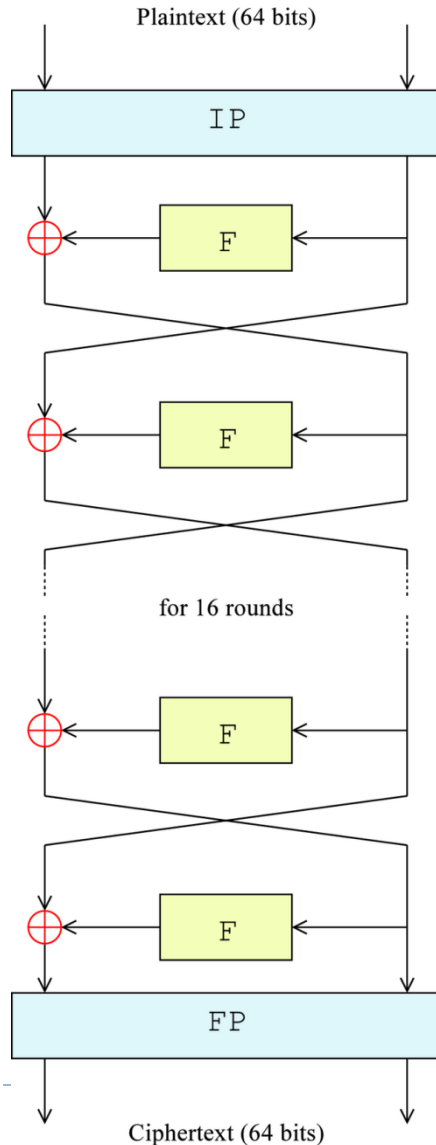


A Titkosító algoritmus



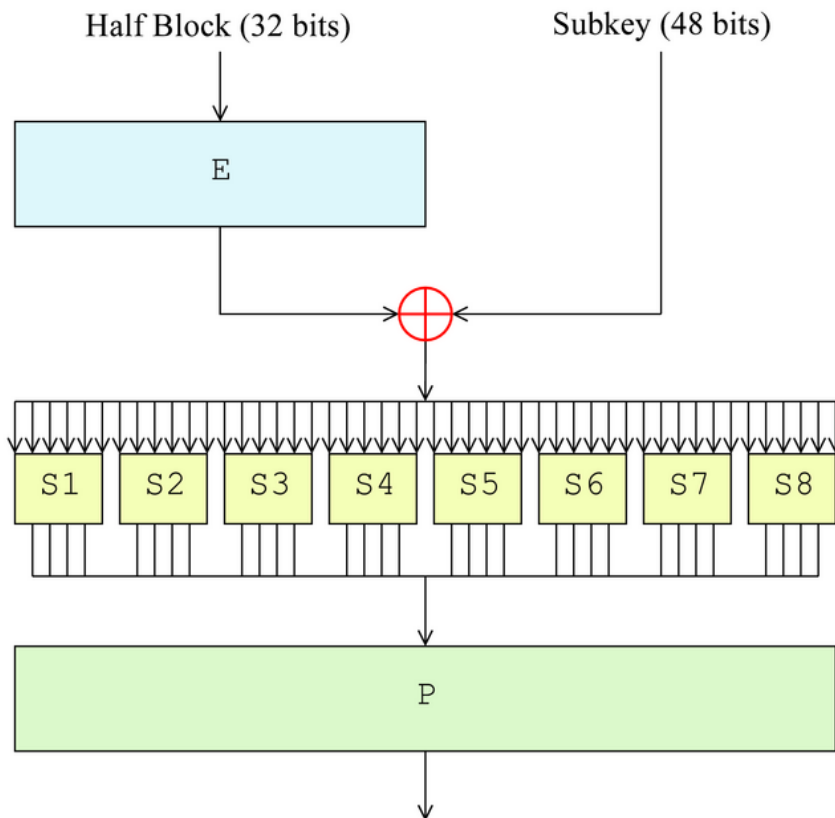
- ▶ A Bemeneti 64 bit hosszú blokk a titkosítás folyamán 2db 32 bites blokként van kezelve
- ▶ A 2db ugyanazzal a funkcióval van titkosítva
- ▶ A titkosítási algoritmus 16 azonos körből áll.
- ▶ XOR – al vannak összegezve az egyes blokkok, amelyek cserélődnek folyamatosan az algoritmus során

A Titkosító algoritmus



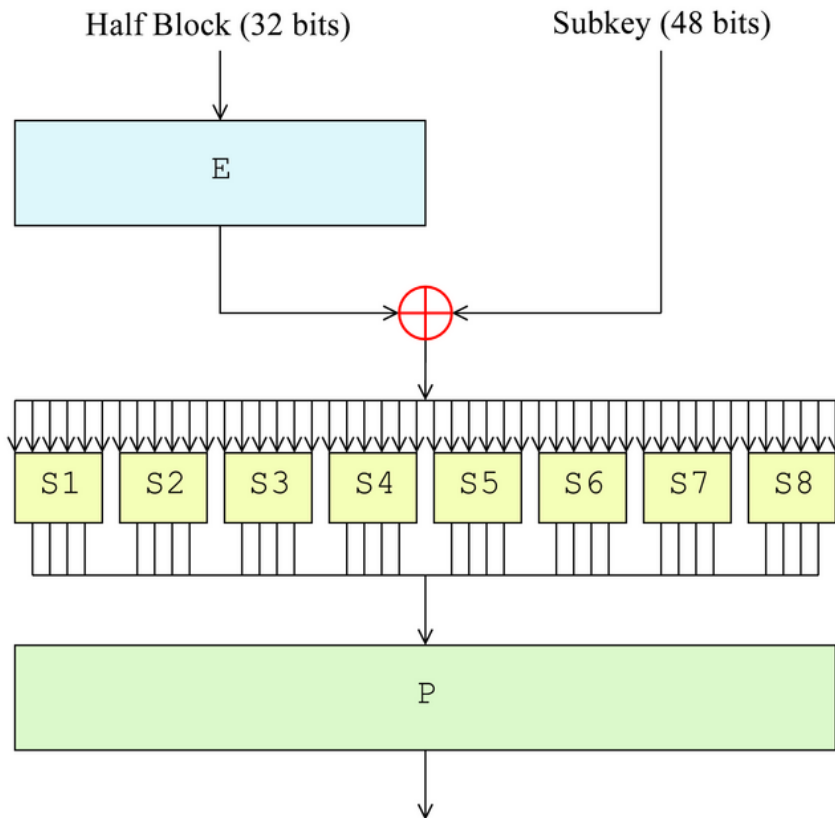
- ▶ Utolsó körben nem cserélődnek a blokkok.
- ▶ Az utolsó körben kihagyott csere és F funkció felépítése miatt fejthető vissza ugyanazon algoritmussal

Az F funkció



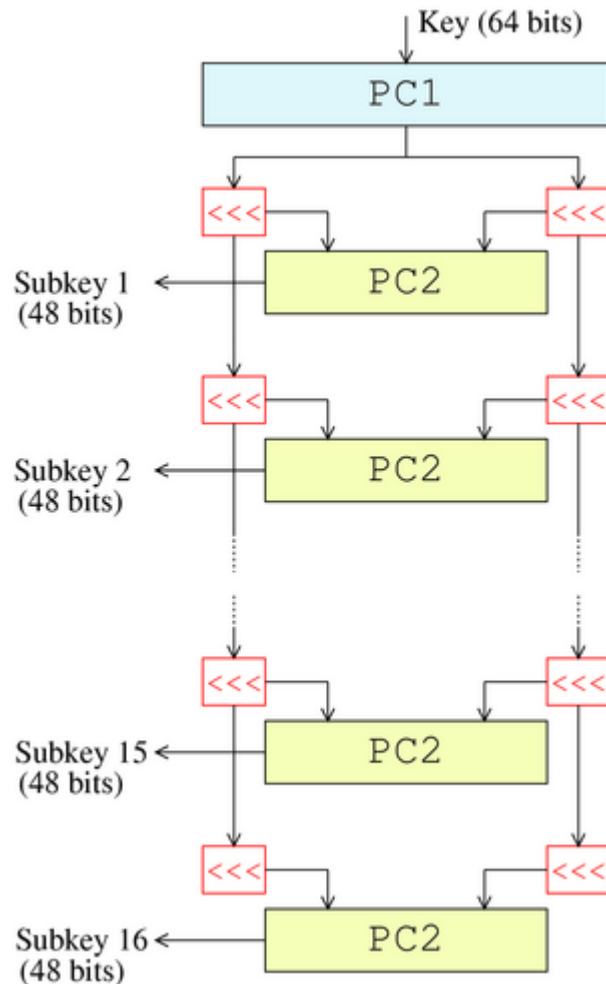
1. Bemeneti 32 bit 48 bitre bővítése (E), a bitek felének duplázásával
2. Kulcskeverés: alkulcs XOR adat elven. Minden F híváskor (16 van összesen) más az alkulcs.

Az F funkció



1. XOR után az adat 8×6 bitre van osztva. A 6 bit egy táblázat alapján cserélődik 4 bitre. A 6 bitből nem lineáris módon lesz 4 bit.
2. Végső permutáció a 8×4 bit kimeneten

Kezdeti kulcsból alkulcsok előállítása



1. Paritás leválasztása a 64 bitből. Eredmény: 56 bit.
2. 56 bit 2x24 bitre osztása
3. Bit eltolások 1 vagy 2 bittel.
4. Kimeneti 48 bit a 2db 24 bites szám permutációjaként áll elő

A DES biztonsága

- ▶ 56 bit kulcs, nagyjából $7,21 \cdot 10^{16}$ kulcs lehetőség
- ▶ Nyers, optimalizálatlan Brute Force - al ha 1 millió kulcsot próbálunk ki 1mp alatt, akkor is ~1150 év lenne megtörni.
- ▶ Speciális Cél Hardver segítségével Brute Force támadással 1998-ban törték meg először pár napon belül.



Brute Force Cél géppel

- ▶ 1991-ben már voltak rá tervek
- ▶ Akkor durván 1 millió \$-ra becsülték az építés költségét.
- ▶ Elvben 3,5 óra alatt tudta volna visszafejteni a kulcsot.
- ▶ Sosem épült meg pénzhiány miatt.

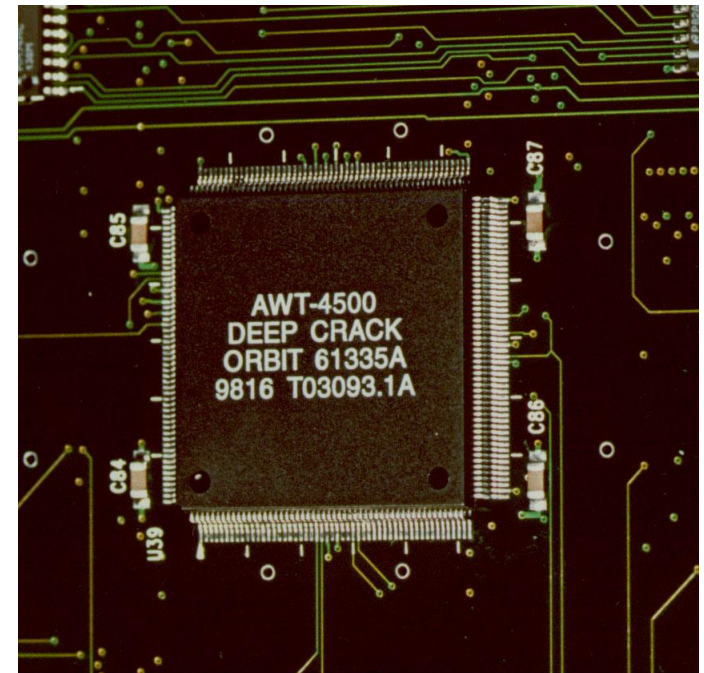
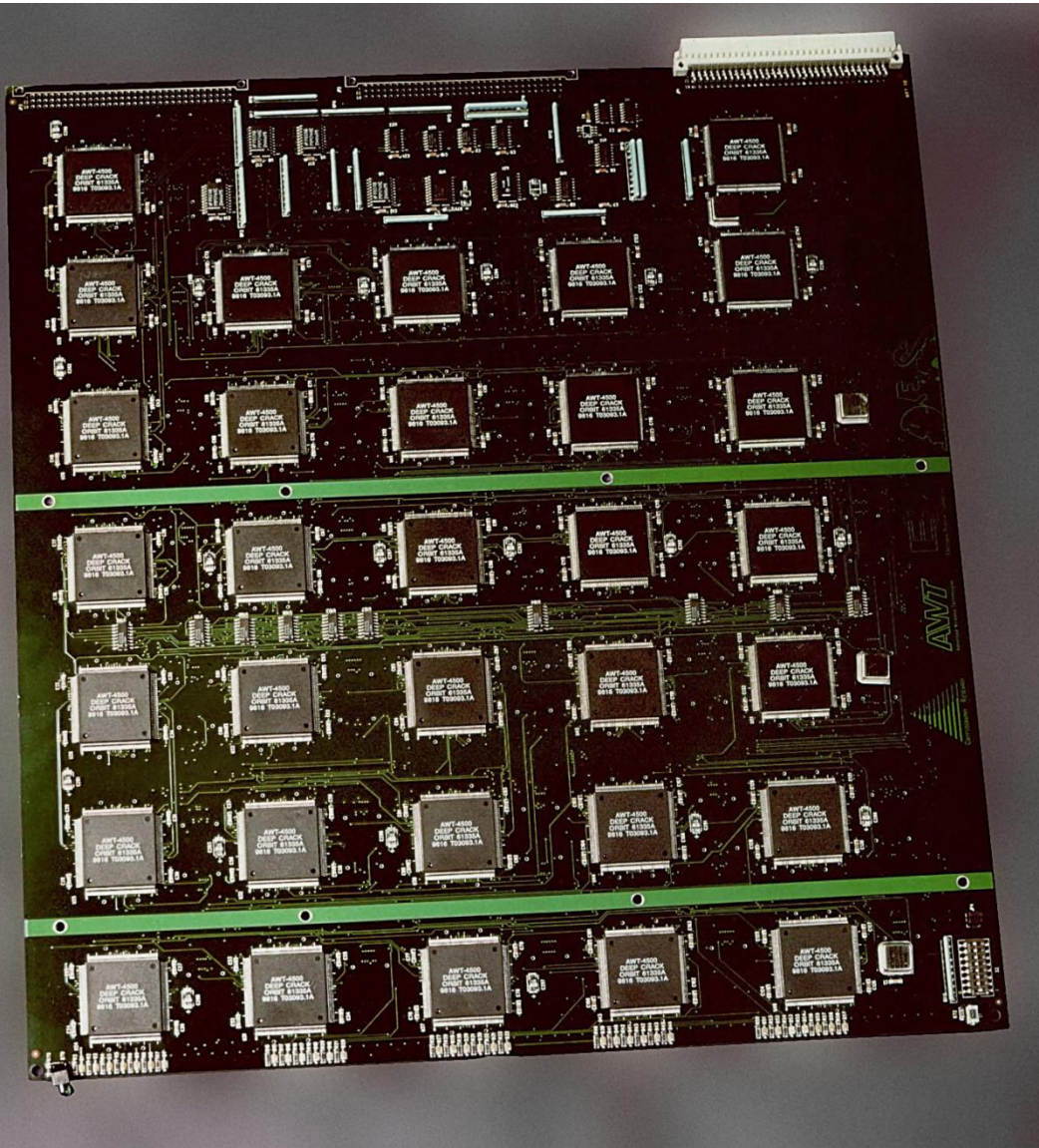


Brute Force Cél géppel

- ▶ DeepCrack elnevezésű gép – EFF alapítvány rendelte meg.
- ▶ 64 egyedi tervezésű processzort tartalmazott.
- ▶ Egy DES Kulcs megtörése 4-5 nap alatt bonyolultságtól függően.
- ▶ 250 000 \$ volt a megépítés költsége, jelenlegi árfolyamon durván 55 millió Ft.
- ▶ Később, durván 50 000 \$-ból építhető volt hasonló gép.



Deep Crack számítógép



Törési versenyek

- ▶ RSA Inc. Támogatta, célja az volt, hogy bebizonyítsák, hogy a DES elavult.
- ▶ Rekordok:
 - ▶ Pentium1 CPU + 16Mb ram -> 96 nap; 1997. január
 - ▶ Több géppel -> 41 nap; 1997. február
 - ▶ EFF DeepCrack -> 56 óra; 1998 július
 - ▶ Interneten összekapcsolt több géppel -> 20 óra 19 perc; 1999. január 19. (DeepCrack + 100000 PC)



Mégis hogy lehetséges?

- ▶ Összetett kriptóanalízissel sikerült optimalizálni a Brute Force eljárást
- ▶ Mindenki számára publikusan letölthető a Cracking DES c. könyvben
- ▶ Amazon.com-on nagyjából 4\$-ért megvehető.
- ▶ Számos publikus törőprogram. Pl:
<http://www.brianhpratt.net/cms/index.php?page=des-cracker>



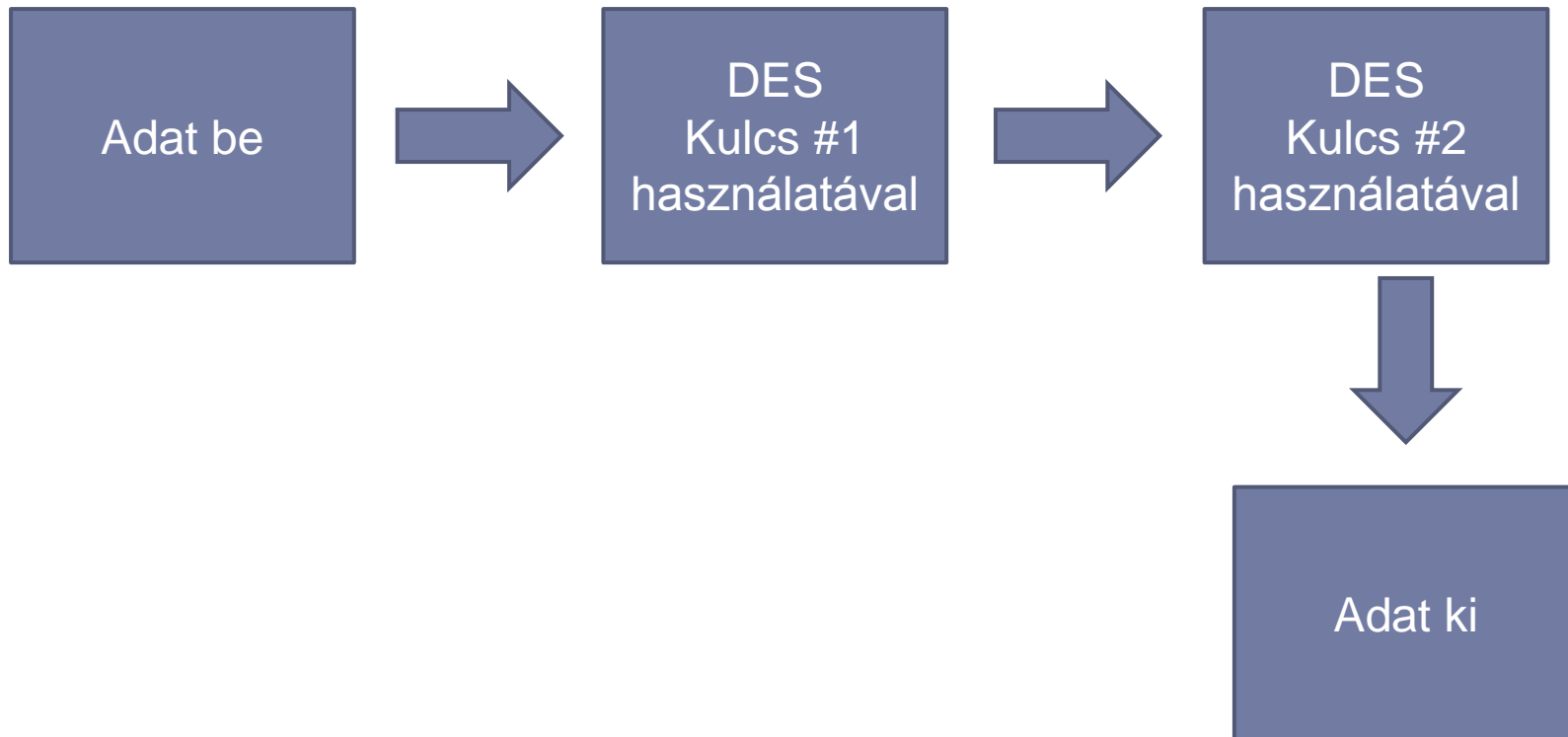
Des újra biztonságossá tétele

- ▶ Dupla DES (Double DES)
- ▶ Tripla DES (Triple DES)
- ▶ 3DES



Dupla DES

- ▶ DES titkosítással titkosított adat ismételt DES titkosítása más jelszóval.

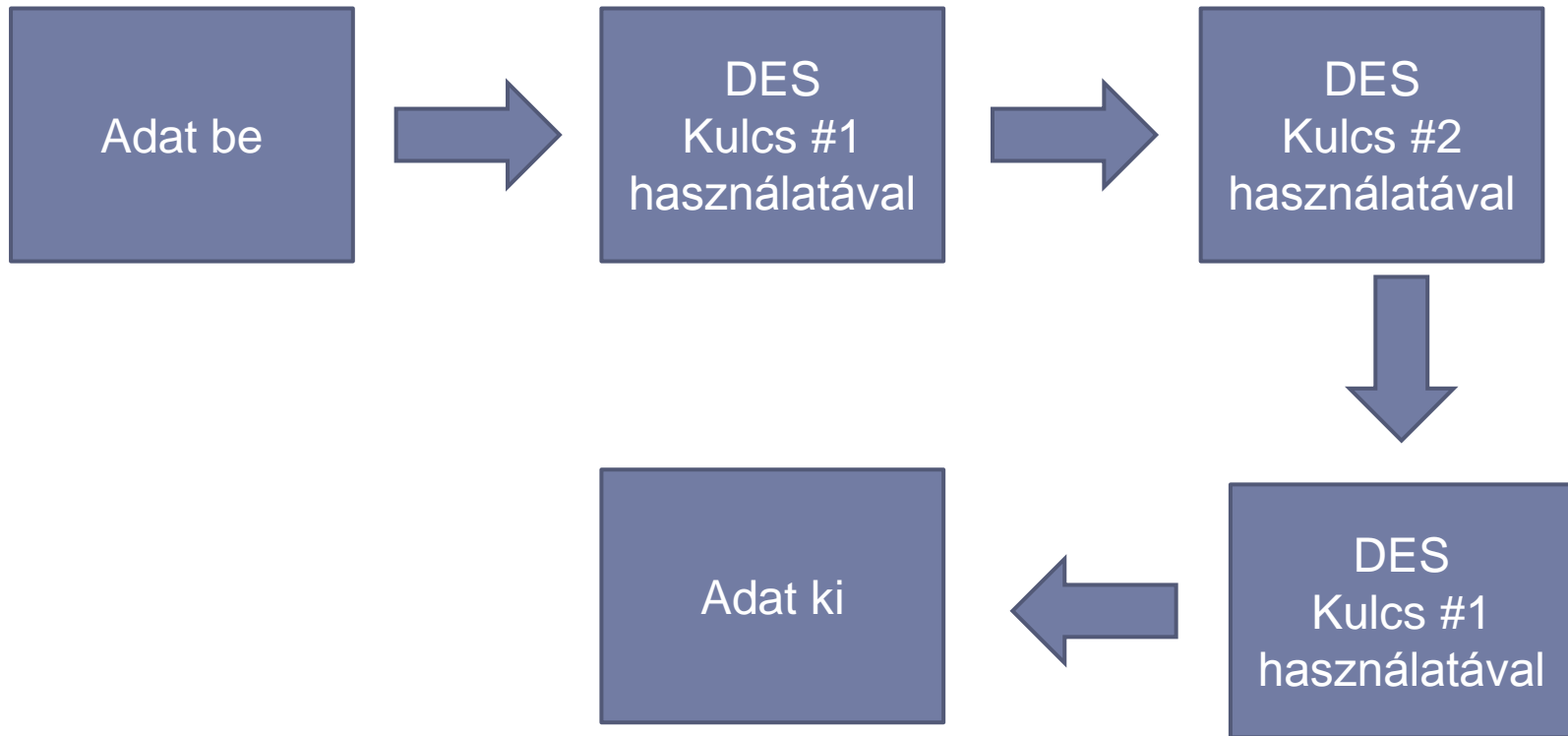


Tripla DES és 3DES

- ▶ Tripla DES: 3 körös DES, 3 különböző jelszóval
- ▶ 3DES
 - ▶ Nem azonos a Tripla DES algoritmussal
 - ▶ Kriptóanalízissel bebizonyították, hogy 3 jelszó felesleges, ugyanolyan erős a titkosítás, ha csak 2 jelszót használunk.
 - ▶ Ez lett a 3 DES



3DES



3DES Biztonsága

- ▶ 168 bites titkosítás
- ▶ Ezen elven Brute Force törés ellen tovább növelhető lenne a biztonsága extra körök beiktatásával.
- ▶ Olcsó megoldás új algoritmus helyett.
- ▶ Azonban elvénél fogva előbb-utóbb megtörhető ez is.
- ▶ Ideiglenes megoldásnak azonban jó volt.



A DES valódi utódja

- ▶ AES Titkosítás
- ▶ Erről majd egy másik előadáson lesz részletesen szó.
- ▶ Sokkal bonyolultabb, mint a DES.
- ▶ Szintén szabványosított.





Út az AES Felé

A DES Feltörése utáni idő

- ▶ Masszív fejlesztési láz
- ▶ Több helyettesítő algoritmus is született:
 - ▶ RC5, később RC6
 - ▶ Blowfish
 - ▶ Triple DES
 - ▶ IDEA
 - ▶ FEAL
- ▶ Egyik sem váltotta fel szabvány szinten a DES-t



Pár szó a DES alternatíváiról

- ▶ Nagyjából mind 64 bites blokkokban dolgozik
- ▶ Növelt kulcstér: 128 -> 2048 bites kulcs
- ▶ Elterjedés hiánya:
 - ▶ Nem megfelelő támogatottság (marketing)
 - ▶ 1-2 esetben részletes kriptóanalízis hiánya
 - ▶ AES szabvány kidolgozása
 - ▶ Némelyik a mai napig igen jónak bizonyult
 - ▶ Sok algoritmus továbbfejlesztett változata ma is használatos



A DES leváltása, az AES megszületése

- ▶ Gondok a DES algoritmussal
 - ▶ 56 bites kulcsok
 - ▶ Főként cél hardware-re lett tervezve
 - ▶ Szoftveresen jóval lassabb
 - ▶ Triple DES megoldotta a kulcstér problémát, de nem volt hozzá cél hardware
 - ▶ Szoftverből igencsak lassú volt 1997-1998 környékén.



A DES leváltása, az AES megszületése

- ▶ 1997 január 2: az Amerikai Szabványhivatal (NIST) bejelenti, hogy a DES-t le fogják váltani AES néven.
- ▶ Alternatíva kidolgozása helyett verseny hirdetése 1997 szeptember 12.-én
- ▶ 9 hónap ált rendelkezésre a fejlesztőknek
- ▶ Számos algoritmus született



A DES leváltása, az AES megszületése

- ▶ Az algoritmusokkal szemben támasztott elvárások a következőek voltak:
 - ▶ 128 bites blokkméretben dolgozzon
 - ▶ 128, 192 és 256 bites kulcsméretet támogasson
- ▶ Ezen feltételeknek megfelelő algoritmusok igencsak ritkák voltak abban az időben.



A nyertes kiválasztása

- ▶ A beérkezett algoritmusokat egy szakértő csoport vizsgálta a következő szempontok alapján:
 - ▶ Biztonság
 - ▶ Algoritmus gyorsasága különböző körülmények mellett:
 - ▶ Kevés memória
 - ▶ Lassú CPU
 - ▶ Eltérő architektúrák
 - ▶ Cél HW építése FPGA chip-ek segítségével



A nyertes kiválasztása

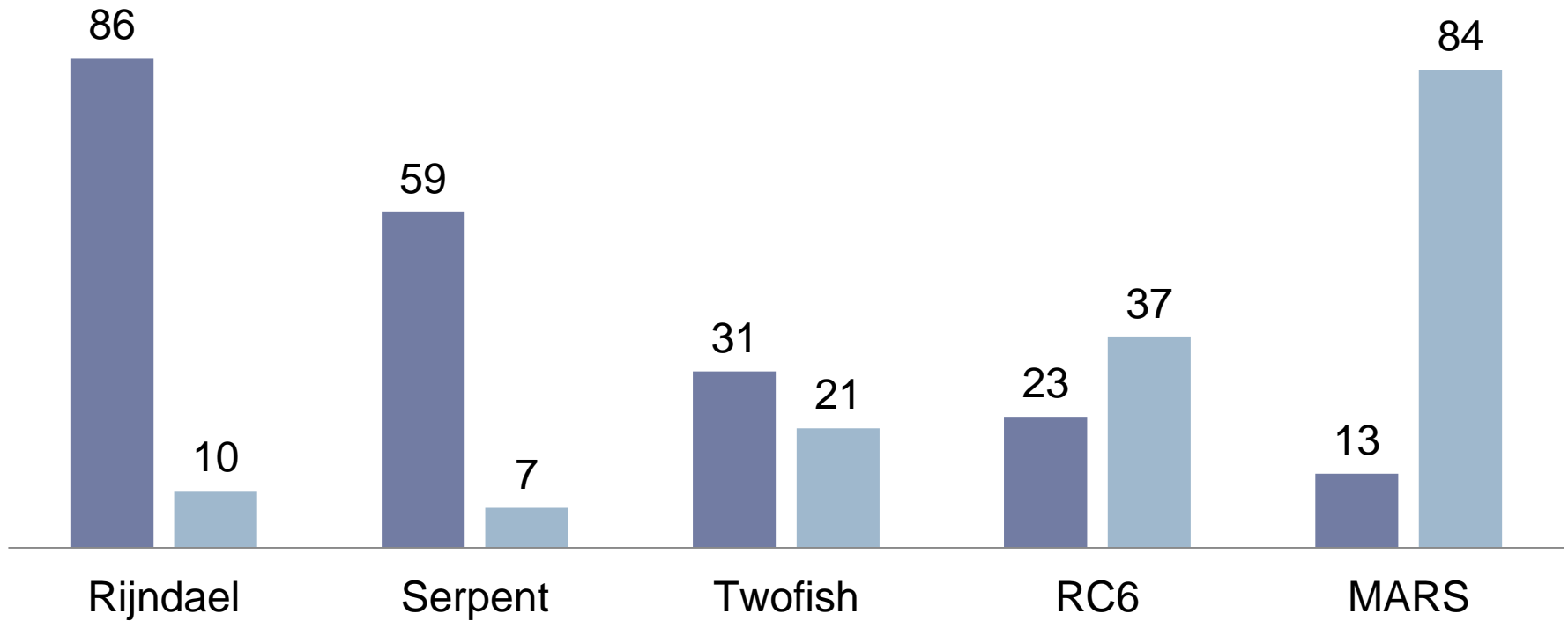
- ▶ A beküldött algoritmusok egy része a biztonság ponton vérzett el.
- ▶ Konferenciák a nyertes algoritmus kiválasztásáról:
 - ▶ AES1: 1998 Augusztus
 - ▶ AES2: 1999 Március
- ▶ Konferenciák eredményeképpen sikerült kiválasztani a döntős algoritmusokat



Döntős algoritmusok szavazatai

Szavazás alakulása

■ Pozitív ■ Negatív



A szabvány létrejötte

- ▶ 2000. október 2: A NIST bejelenti, hogy az AES szabvány algoritmus a Rijndael lesz.
- ▶ 2001. november 26: Az AES szabvány elfogadásra került.



Az algoritmus

- ▶ 2 belga matematikus munkájának eredménye
- ▶ Joan Daemen és Vincent Rijmen
- ▶ Az algoritmus neve az ő nevük kombinálásából született meg.



Az algoritmus hatékonysága

- ▶ 2002-ben az NSA elfogadta első nyílt forráskódú algoritmusként amely alkalmas szigorúan titkos információk védelmére.
- ▶ Nagyon biztonságos.
- ▶ Elméletben lehetséges törni, de a mai számítási teljesítmény mellett lehetetlen.
- ▶ Elvileg 10-20 év múlva lesznek csak olyan gépek, amelyek belátható időn belül végeznek a feltöréssel.



Elméleti támadások

- ▶ 128 bites kulcs esetén:
 - ▶ $2^{126,1}$ lehetséges kulcs
- ▶ 192 bites kulcs esetén:
 - ▶ $2^{189,7}$ lehetséges kulcs
- ▶ 256 bites kulcs esetén:
 - ▶ $2^{254,4}$ lehetséges kulcs.



Ez alapján feltörés ideje

- ▶ Egy 4 magos CPU-val, amely magonként 3 millió kulcsot próbál végig egy 128 bites kulcs esetén:
 - ▶ Nagyjából $2,4 \times 10^{23}$ év kellene a megtöréshez.
 - ▶ Ilyen CPU jelenleg nincs, mivel feltételeztük, hogy 1 órajel ciklus alatt 1 kulcsot kipróbál a rendszer.
 - ▶ Valóságban ennél 2,3x lassúbbak a processzorok
 - ▶ GPU sem elég gyors a célra



Az AES működése

- ▶ Nem hasonlítható a korábbi titkosítási algoritmusokhoz, mivel azok működése alapvetően soros volt.
- ▶ Az AES ezzel szemben mátrix transzformációk sorozata.
- ▶ 128 bites blokkokban dolgozik. Kulcsméret: 128 bit, 192 bit, 256 bit.
- ▶ Elvben a kulcsbitek száma a végtelenségig növelhető, a blokkméret viszont csak 256 bit-ig.



AZ AES Biztonsága

- ▶ Többmagos CPU-k terjedésével a törési kísérletek új lendületre kaptak.
- ▶ Matematikusok készítettek olyan algoritmust, amely 8 925 512 év alatt fel tudná törni a 128 bites titkosítást.
- ▶ Ez jóval kevesebb, mint ami kellene a 128 biteshez, azonban így is csak elméleti a támadás.



AZ AES Biztonsága

- ▶ Mivel a kulcs komplexitása a végtelenig növelhető, így ha mondjuk feltörük a 128 bites titkosítást, akkor egyszerűen csak növeljük a kulcsméretet.
- ▶ A kulcsméret duplázása 128 bitről 256 bitre $3,4 \cdot 10^{38}$ lehetőséggel bővíti a lehetséges kulcsok számát.
- ▶ Az USA jelenleg 256 bites titkosítást használ a szigorúan titkos dokumentumokhoz.



AZ AES, mint szabvány

- ▶ Nyílt szabvány, amit bárki szabadon implementálhat.
- ▶ Implementációtól függően eltérő lehet a biztonsága az egyes csomagoknak.
- ▶ Az implementáláshoz a NIST (National Institute of Standards and Technology) publikált egy tesztcsomagot is, amellyel tesztelhető a különböző implementációk biztonsága.
- ▶ Éles körülmények között érdemes ezzel tesztelni használat előtt.



Implementációk biztonsága

- ▶ Mivel a publikált teszt nem teljes körűen átfogó, ezért az NIST bevizsgálást is kínál.
- ▶ Egy ilyen bevizsgálás akár 30.000\$ vagy több is lehet.
- ▶ Ebből kifolyólag nem sok implementáció rendelkezik az NIST által elfogadott minősítéssel.



Implementációk biztonsága

- ▶ Miért fontos ez ?
- ▶ 2011: OpenBSD által fejlesztett implementáció körül kétségek merültek fel egy lehetséges hátsó kapu sebezhetőséggel kapcsolatban.
- ▶ A kivizsgálás még most is folyamatban van.
- ▶ A gond ezzel a BSD Licenc, ami miatt rengeteg termék használja ezen implementációt.



A BSD Licenc gondjai

- ▶ Nyílt forráskódú licenc.
- ▶ Lényegében azt írja le, hogy:
 - ▶ Az alkotó lemond a szoftver tulajdonjogáról és a közösségre ruházza át, akik azt csinálnak vele, amit akarnak.
 - ▶ Nincs megkötés, hogy ha módosítja valaki a programot, akkor publikálnia kell a módosított forrást.
 - ▶ Az eredeti alkotót sem kötelező megnevezni.



AZ AES Elterjedése

- ▶ Rengeteg programban és hardverben megtalálható.
- ▶ PI:
 - ▶ ZIP tömörítés
 - ▶ WPA2 WLAN titkosítás
 - ▶ Teljes lemezt titkosító megoldások
 - ▶ Processzorok
 - ▶ Telefonok



Az AES sebessége

- ▶ Jól optimalizálható.
- ▶ Rengeteg műveletsor gyorsító táblázatból megoldható.
- ▶ Pentium Pro processzor esetén 1 byte titkosítása 18 órajel ciklust vesz igénybe. (120 MHz mellett ez 6,6 MB/s sebesség)
- ▶ Újabb processzorok HW szinten gyorsítottak AES titkosítással.



AES a processzorokban

- ▶ Új utasítás készlet: AES-NI
- ▶ Összes Intel Core i5 és i7 processzor tartalmazza.
- ▶ Összes AMD Bulldozer CPU tartalmazza.
- ▶ 5x-6x gyorsabb titkosítást tesz lehetővé a tisztán szoftveres megoldásokhoz képest.
- ▶ Programnak támogatnia kell ezen utasításkészletet.



Népszerű programok amik támogatják

- ▶ TrueCrypt
- ▶ BitLocker
- ▶ WinZip
- ▶ WinRar
- ▶ 7zip



Kétkulcsos rendszerek

RSA

Kétkulcsos rendszerek

- ▶ A kétkulcsos rendszerek, más néven a nyílt kulcsú rendszerek.
- ▶ Ilyen rendszerek esetén az ember 2 kulccsal rendelkezik.
 - ▶ Titkos kulcs: titokban tartandó
 - ▶ Nyilvános kulcs: szabadon terjeszthető.
 - ▶ A kulcsok összefüggnek, azonban csak a nyilvános ismeretében nem határozható meg a titkos kulcs.



Kétkulcsos rendszerek

- ▶ Titkosításra a nyilvános kulcsot használjuk.
- ▶ A kész üzenet visszafejthetetlen a titkos kulcs ismerete nélkül.
- ▶ Ilyen rendszerek az RSA és a PGP



Probléma a szimmetrikus kulcsú rendszerekkel

- ▶ Hiába jó az algoritmus, ha a kulcs gyenge
- ▶ A kulcs ismeretében nem csak titkosítani tudok, hanem vissza is tudom azt fejteni.
- ▶ Emiatt a kulcs eljuttatása macerás 2 fél között anélkül, hogy egy 3. fél ne tudjon róla.
- ▶ Titkosítatlan csatornán további titkosítást, rejtést igényel: PI szteganográfiai módszerekkel.



KÉTKulcsos rendszerek

- ▶ Elv: a titkosító kulcs és titkosítást feloldó kulcs nem azonos.
- ▶ Elnevezések:
 - ▶ Titkos kulcs: Ez a titkosítás feloldó kulcs
 - ▶ Publikus kulcs: Ez a titkosító kulcs
- ▶ A kulcsok összefüggenek.



Az RSA megszületése

- ▶ 1978-ban találta ki 3 kriptográfus:
 - ▶ Ron Rivest
 - ▶ Adi Shamir
 - ▶ Leonard Max Adleman
- ▶ A név a nevük kezdőbetűiből jön.
- ▶ Az algoritmus prímszámokat és a prím tényezőös felbontás problémáját használja fel.



Prím tényezős bontás problémája

- ▶ Elv: Minden szám felbontható prímtényezők szorzatára.
- ▶ Gond: prímszámok megtalálása.
- ▶ A probléma halmozottan fent áll nagy számoknál.
- ▶ Konkrétan a legjobb algoritmus ideje egy N számra, amely b biten írható le:

$$O\left(\exp\left(\left(\frac{64}{9}b\right)^{\frac{1}{3}}(\log b)^{\frac{2}{3}}\right)\right).$$



AZ RSA működése

1. 2db egymástól távol álló prím választása: p és q
 1. p és q véletlenszerűen választott és nagyjából azonos bithosszúsággal lehet őket reprezentálni bináris formában
2. $n = p * q$ kiszámítása



AZ RSA működése

3. $\varphi(n) = (p-1)(q-1)$ számítása. A φ Euler függvény, amely megadja, hogy egy N számhoz mennyi nála kisebb relatív prímszám tartozik. **a és b relatív prím**, ha az 1-en és -1 -en kívül nincs más közös osztójuk.
4. Véletlenszerű e szám választása, amelyre igaz: $1 < e < \varphi(n)$ és $\gcd(e, \varphi(n)) = 1$;
5. e lesz a publikus kulcs kitevője, publikus kulcs: n^e



Az RSA Működése

6. d meghatározása az alábbi formában:

$$d \equiv e^{-1} \pmod{\varphi(n)}$$

- ▶ d nem más, mint a moduláris multiplikatív inverze a e mod $\phi(n)$ számnak.
- ▶ d szám lesz a privát kulcs kitevője. Privát kulcs: n^d



Az RSA Működése

- ▶ Titkosítás adott n^e publikus kulcs segítségével:
 - ▶ m üzenet titkosításához az üzenetet számmá kell alakítani, méghozzá úgy, hogy $0 < m < n$. Erre egy előre egyeztetett visszafordítható sémát alkalmazunk ún. helykitöltés
 - ▶ A titkos c üzenet ezután az alábbi módon áll elő:
$$c = m^e \pmod{n}$$



Az RSA működése

- ▶ Titkosításfeloldás ismert n^d titkos kulcs esetén:

$$m = c^d \pmod{n}$$



RSA Biztonsága

- ▶ Kulcstér: 1024 bit és 4096 bit között van tipikusan
- ▶ Ennél kisebb és nagyobb kulcstér is használható
- ▶ Kis kulcstér esetén törhető az algoritmus, mivel ekkor:
 - ▶ e és d értéke próbálkozással kiszámítható, mivel nincs keverés a kimenetben.



RSA Biztonsága

- ▶ Nagy kulcstér esetén azonban biztonságos.
- ▶ Biztonsági kockázat továbbá a helykitöltésünk jósága is. Ezért célszerű alkalmazni az úgynevezett **PKCS1** rendszert, amely biztonságos.
- ▶ További gond, ha a véletlenszám előállító algoritmusom hibás.



Gondok a nem igazán véletlen számokból

- ▶ 2008: Debian OpenSSL ügy
- ▶ Valamelyik fejlesztő „véletlenül” kitörölte a véletlenszám generátor inicializálásáért felelős kódrészletet.
- ▶ Ezáltal az összes ezen rendszeren generált RSA kulcs törhetővé vált.



RSA a mindennapi életben

- ▶ Számos helyen alkalmazott:
 - ▶ SSL protokoll titkosítás
 - ▶ HTTPS
 - ▶ SSH
 - ▶ Számos kereskedelmi megoldás, amit az RSA Laboratories fejleszt. Köztük: HW titkosított USB kulcsok, védelmi chip-ek (Playstation 3), stb...



A HTTPS működése vázlatosan

- ▶ Kliens: privát - és publikus kulcs.
- ▶ Szerver: privát - és publikus kulcs.
- ▶ Kliens felveszi a kapcsolatot a szerverrel, közli, hogy milyen titkosítást támogat.
- ▶ Szerver erre elküldi az ő publikus kulcsát, majd a kliens is az ő publikus kulcsát.
- ▶ Kommunikáció ezután indul meg.



A HTTPS működése vázlatosan

- ▶ A kapcsolat akkor megbízható, ha a felhasználó valóban azzal kommunikál, akivel szeretne.
- ▶ Ezért kellene rendelkeznie a szervernek egy tanúsítvánnyal, ami az identitásának helyességét és a kapcsolat biztonságát garantálja.
- ▶ Ezt egy úgynevezett megbízható 3. félnek kellene kiállítania.



A HTTPS működése vázlatosan

- ▶ Sok szerver nem rendelkezik ilyennel
- ▶ Egy ilyen aláírás nem olcsó mulatság.
- ▶ 1 éves érvényességgel 500\$ és 1500\$ dollár között mozognak a tanúsítványárak szolgáltatótól függően.
- ▶ Újabb böngészők (Firefox 2.0 óta kb) figyelmeztetnek a nem megfelelően aláírt tanúsítványok esetén. NEM KELLENE FIGYELMEN KÍVÜL HAGYNI!



Az RSA Elterjedése

- ▶ Jogdíjas szabvány volt 1983 és 2000 között.
- ▶ A szabvány lejárta után igencsak kezdett terjedni a rendszer.
- ▶ Kezdetben csak üzenetek titkosítására használták.
- ▶ Mára inkább már digitális aláírásokban használt.
- ▶ Titkosításban helyette inkább a PGP használt



Digitális aláírás RSA-val

- ▶ Adott x publikus kulccsal, és adott y , aki küldeni akar neki üzenetet.
- ▶ Mivel x publikus kulcsa mindenki által ismert, ezért x nem tudhatja, hogy valóban y akar-e vele kapcsolatba lépni, vagy más, aki y -nak adja ki magát.
- ▶ Itt jön be a digitális aláírás.



Digitális aláírás RSA-val

- ▶ y ezért az üzenetből egy hash-t képez, ezután a titkos kulcsát (n^d) felhasználva a hash értéket kódolja (mintha a hash lenne a titkos üzenet), majd ezt az üzenethez csatolja
- ▶ X szintén hash számítást végez az üzeneten és y publikus kulcsát használva kódolja a hash-t. Ennek eredményeképpen megkapja az y által számított hash értéket, amit ellenőrizni tud a saját hash értékével.





A PGP

PGP

- ▶ Pretty Good Privacy szavakból jön a neve
- ▶ 1991-ben készítette el Phil Zimmermann
- ▶ Szabad szoftver
- ▶ Több technológia kombinálásából jött létre
- ▶ Az RSA-t is felhasználta
- ▶ Mára a legnépszerűbb e-mail titkosítási rendszer



A PGP megszületése

- ▶ 1980-as évek második felében a hidegháború erősödni látszott.
- ▶ Phil Zimmermann ebben az időben igen sok tüntetésre járt
- ▶ Egy atombomba ellenes csoport tagja is volt. + 1 pici üldözési mániája is volt.
- ▶ Ezért merült fel benne az ötlet, hogy létrehoz egy titkosítási rendszert, amit az NSA sem tud megtörni.



A PGP Megszületése

- ▶ Ebben az időben a DES volt a titkosítási etalon.
- ▶ A DES-el az 56 bites kulcstér mellett probléma volt az, hogy az NSA tervezte.
- ▶ Összeesküvéselméletek szerint van benne egy rejtett kapu, amivel az NSA ki tud bontani minden titkosított üzenetet a jelszó ismerete nélkül.



A PGP Megszületése

- ▶ 1991-ben született meg a program.
- ▶ Mivel nem volt internet kapcsolata, ezért egyik barátját kérte meg, hogy tegye fel az internetre.
- ▶ Rohamosan elkezdett terjedni a rendszer.
- ▶ Eredetileg fizetős programként akarta kiadni, de aztán meggondolta magát és ingyenes maradt.



A PGP Megszületése

- ▶ 2 hibát követett el a PGP létrehozásakor:
 - ▶ Nem kért licenszjogot az RSA használatára
 - ▶ Az amerikai törvények egy kalap alá veszik a kódoló/dekódoló programokat az atombombákkal, rakétákkal és ezek exportja fegyverkereskedelemnek számít.
 - ▶ Abban az időben a 40 bitnél erősebb titkosítási algoritmusokra vonatkozott ez a törvény.
 - ▶ A PGP már ekkor nem használt 128 bitnél kevesebbet 😊



A PGP Megszületése

▶ Következmények:

- ▶ 1993: FBI letartóztatja fegyverkereskedelem vádjával
- ▶ Az RSA is beperelte
- ▶ Később az RSA visszavonta a pert, mivel ekkor már igencsak elterjedt volt a rendszer
- ▶ 1996-ban felmentette az FBI is a fegyver kereskedelem vádja alól



A PGP Megszületése

- ▶ Folyamatos FBI piszkálás miatt, a forráskódot neten nem lehetett terjeszteni.
- ▶ Ezt úgy játszották ki, hogy a forráskódot könyv formájában árulták 😊
- ▶ A könyvek exportja már nem számított fegyverkereskedelemnek. Sőt, a könyv exportjának a tiltása az alkotmány szólásszabadság jogába ütközött volna 😊



A PGP megszületése

- ▶ Ezután apróbb hibák merültek fel az algoritmusban, így Zimmermann megnyitotta a rendszer forráskódját.
- ▶ RFC 4880
- ▶ 6.5.1-es változata óta van nemzetközi változat és amerikai. (a szoftverszabadalmak miatt)



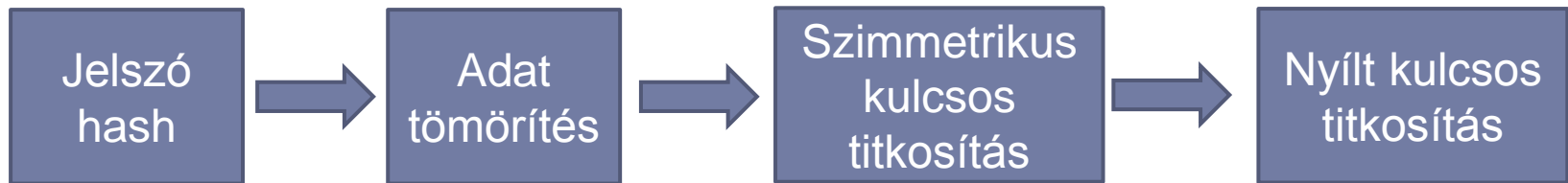
Szálak kuszálódása a PGP körül

- ▶ A PGP-t megvették, majd létrejött a PGP Corporation
- ▶ OpenPGP Alliance megszületik
- ▶ Létrejön egy szervezet is, amit PGP international-nek neveznek



A PGP működése

- ▶ Több technológiát használ fel, folyamatosan fejlődik.
- ▶ Blokkvázlatban a működése:



A PGP Működése

- ▶ Minden lépésben több algoritmust támogat
- ▶ Emiatt az egymással PGP titkosítással kommunikáló feleknek meg kell egyezniük a használt PGP beállításokon.
- ▶ Mivel folyamatosan fejlődik, így az újabb PGP verzióval készített fájlok nem bonthatóak ki régebbi PGP változatokkal, még a jelszó ismeretében sem.



A PGP alkalmazási területei

- ▶ Teljes lemez titkosítás
- ▶ E-mail titkosítás
- ▶ E-mail aláírás



E-mail titkosítás PGP-vel

- ▶ Joggal merülhet fel a kérdés, hogy miért szükséges, mikor régóta van HTTPS és TLS?
- ▶ Az ok: SPAM levelek terjedése
- ▶ Magyarországon és nemzetközileg is bevett gyakorlat az, hogy az Internetszolgáltató cégeknek szűrniük kell a kéretlen leveleket.
- ▶ Ez szép és jó, azonban ez azt is magával vonja, hogy titkosítottan nem lehet levelet küldeni.



E-mail titkosítás PGP-vel

- ▶ Nem lehet, mivel egy TLS titkosított csatornán keresztül menő levélbe nem tud belenézni a SPAM szűrő.
- ▶ Ezért általában blokkolva van a titkosított e-mail küldési lehetőség.
- ▶ Itt jön be a PGP alkalmazása.



A PGP Biztonsága

- ▶ Talán a legbiztonságosabb rendszer
- ▶ Nincs ismert eset arról, hogy valaki jelszó hiányában fel tudta volna törni.
- ▶ Csak olyan algoritmusok vannak benne, amelyek önmagukban is biztonságosak lennének.
- ▶ Emiatt csak és kizárólag Brute Force módszerrel törhető fel.



A PGP Biztonsága

- ▶ Több bűnügyben bebizonyosodott, hogy az FBI sem tudja feltörni a PGP-vel titkosított üzeneteket.
- ▶ Egy 2006-os incidens kapcsán az Amerikai Vámügyi Hivatal megjegyezte, hogy majdhogynem lehetetlen feltörni a PGP titkosított fájlokat.



A PGP Biztonsága

- ▶ Az amerikai alkotmány és számos más alkotmány alapján egy ember sem vehető rá, arra, hogy kiadja a jelszavát.
- ▶ Ez ütközik az alkotmány azon feltételezésével, hogy mindenki ártatlan, míg be nem bizonyítják bűnösségét.
- ▶ Érdemes használni.



Az ACTA és a SOPA, valamint a jövő

- ▶ Talán a jövőben mindenki rá lesz kényszerítve a titkosításra, köszönhetően ezen zseniális törvényeknek.
- ▶ Ezekről későbbi előadáson lesz szó.
- ▶ Azonban ha valósággá válnak gyakorlatban is, akkor hatalmas öngól lesz ez a kormányoknak.
- ▶ A viszonylag elhanyagolható titkosított forgalom helyett minden titkosított lesz.



Használati leírások

- ▶ Elsősorban a GPG program használatát érdemes átnézni.
- ▶ Ez egy GNU PGP implementáció
- ▶ Főként Unix és Linux rendszerekre van fejlesztve elsősorban
- ▶ Azonban van Windows változat is, de ennek használata macerásabb
- ▶ <http://moser.cm.nctu.edu.tw/gpg.html>

