# Blowfish titkosítás

## Bevezetés

A Blowfish egy szimmetrikus kulcsú blokkos titkosító, amit 1993-ban Bruce Schneier tervezett, amivel a DES titkosítási technikát szándékozta leváltani. Számos titkosító készletben és titkosítási termékben szerepel, ilyenek például a jelszókezelők, a fájl/lemez titkosítók (GnuPG, Bcrypt, CryptoForge), operációs rendszerek (Linux, OpenBSD).

A Blowfish jó titkosítási arányt biztosít, gyors, kevés memóriát használ és egyszerűen megvalósítható. A DES-nél lényegesen gyorsabb és a mai napig nem találtak hatékony kriptoanalízis technikát rá. Az algoritmust az alkotója nyilvánossá tette, nem képezi szabadalom tárgyát annak érdekében, hogy bárki szabadon fel tudja használni.

Az utódja a Twofish, ami 64 bit-es blokkok helyett 128 bit-es blokkokat használ.

## Működésének alap elemei

* A bemeneti szöveget a Blowfish 64 bit-es blokkonként dolgozza fel.
* A kulcs hosszúsága szabadon választható 32 és 448 bit között.
* Az alkulcsok száma 18, ezeket P-vel jelölik (P[0]-P[17])
* Helyettesítő dobozokkal (S-dobozokkal) rendelkezik. Mindegyik 256 elemmel rendelkezik és minden elem 32 bit hosszúságú. Az S-dobozok 16x16-os lookup table-ök.
* Mind a kulcs, alkulcs és a helyettesítő dobozok értékei is hexadecimális formában tárolódnak
* A bemeneti 64 bit bináris formátumban tárolódik

## Működése

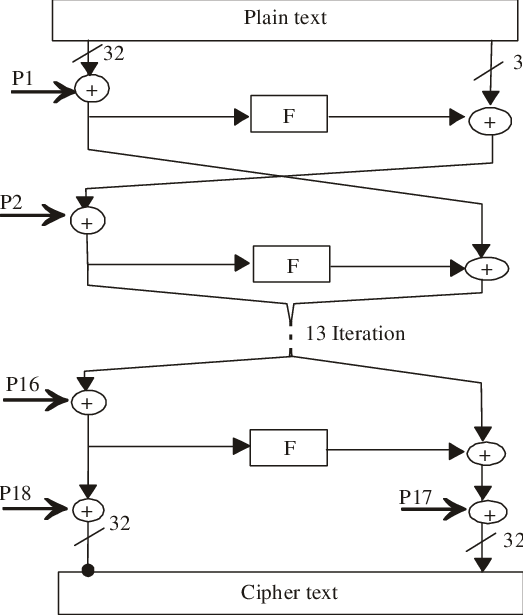
1. Kulcsot generálunk, amit mi határozunk meg. (Meghatározzuk azt, hogy hány bit-es legyen 32 és 448 bit között.)
2. Alkulcsokat generálunk a PI felhasználásával, amit hexadecimálissá alakítunk és 32 bit-es szavakra osztunk fel. Az alkulcsokból összesen 18-at generálunk. Ezek a P-k lesznek.
3. Helyettesítő dobozokat generálunk, 4-et, amiket S0, S1, S2, S3-al jelölünk. Minden helyettesítő doboz 256 elemet tartalmaz. Minden elem 32 bit hosszú szó, amit a PI-ből alkotunk. Ezek a dobozok Lookup table-öknek szolgálnak majd.
4. A P-ket P[i]-t XOR-oljuk a kulcs [i]-edik elemével. A kulcs esetén, ha például 448 bit-et határoztunk meg, akkor 448/32=14. P1=P1 XOR k1 …. P14=P14 XOR k14 és ezután a P15-öt az első kulcstól elölről kezdjük. P15=P15 XOR k1 … P18 XOR k4.

Így megkapjuk a P-ket a kulccsal titkosítva.

1. A 64 bit-es bináris bemenetet 2 db 32 bit-es kimenetre osztjuk.

Az egyik 32 bit-es felét XOR-oljuk az adott szinten található alkulccsal, P-vel.

Ezt átvisszük egy F függvényen, aminek az eredményét XOR-oljuk a második 32 bit-el.



1. Az F függvényben a bemeneti 32 bit-et 4db 8 bitre osztjuk.

Mindegyik 8 bit-et egy S dobozba bevezetünk.

Itt a 8 bit első 4 bit-je jelöli a sort és a második 8 bit-je az oszlopot.

Megkeressük a 256 elemű S-doboz ’táblázatában’ az így kapott elemet, ami már 32 bit-es lesz.

Ezután mindegyik kimenő S-doboz utáni műveletet elvégezzük.

A picture containing text, diagram, plan, line

Description automatically generated

1. Ezt a 32 bit-es kapott elemet az 5. pont végén leírtak alapján össze XOR-oljuk a másik meglévő 32 bit-el.
2. Ezt mindaddig végezzük, amíg az utolsó P-ig nem érünk (18 alkalommal elvégezzük).

A végén 2 db kódolt 32 bit-es elemet kapunk, ami a kódolt üzenet.

## Továbbfejlesztési lehetőségek

Kép titkosításnál ez a technika nem annyira alkalmas a hatalmas adatok miatt, nagyobb redundancia és a kép képpontjainak kemény korrelációja miatt.

Ezért különböző módosításokat próbálnak meg alkalmazni az algoritmuson annak érdekében, hogy megfelelő legyen erre a célra is.

Az egyik ilyen javasolt rendszer a Blowfish algoritmus módosítása egy 3-D kaotikus rendszer (Improved Lorenz) használata, amely kulcs menetrendként működik a Blowfish alkulcsok elkészítéséhez, annak érdekében, hogy a véletlenszerűségét növelje ezeknek a generálásánál. Ezzel memóriát és erőforrást is megspórolva.

Ez a megközelítés az alkulcs generálásának lecserélésével próbálja meg javítani az algoritmust.

<https://ieeexplore.ieee.org/document/9971261>

Egy másik megközelítés esetében pedig, először a digitális képet véletlenszerűen több kulcsalapú blokkra bontják, majd minden egyes blokkot a Blowfish-algoritmuson átvezetnek.

A javasolt rendszert úgy tervezték, hogy kihasználja a nagy teljesítményű lehetőség előnyeit, amelyet egy kaotikus térkép támogat, ami egy sokkal jobb biztonság/teljesítmény kompromisszumot eredményez.

<https://ieeexplore.ieee.org/document/9075747>

# Felhasznált irodalom

<https://www.encryptionconsulting.com/education-center/what-is-blowfish/>

<https://www.geeksforgeeks.org/blowfish-algorithm-with-examples/>

<https://en.wikipedia.org/wiki/Blowfish_(cipher)>

<https://www.researchgate.net/publication/278671149_A_New_Approach_for_Improving_Data_Security_using_Iterative_Blowfish_Algorithm>