Blowfish (cipher)

From Wikipedia, the free encyclopedia

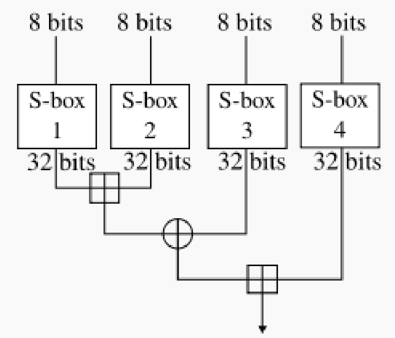
This article is about the computer cipher. For the fish species, see blowfish. For other uses, see Blowfish (disambiguation).

Blowfish is a symmetric-key block cipher, designed in 1993 by Bruce Schneier and included in a large number of cipher suites and encryption products. Blowfish provides a good encryption rate in software and no effective cryptanalysis of it has been found to date. However, the Advanced Encryption Standard (AES) now receives more attention.

Schneier designed Blowfish as a general-purpose algorithm, intended as an alternative to the aging DES and free of the problems and constraints associated with other algorithms. At the time Blowfish was released, many other designs were proprietary, encumbered by patents or were commercial or government secrets. Schneier has stated that, "Blowfish is unpatented, and will remain so in all countries. The algorithm is hereby placed in the public domain, and can be freely used by anyone."

Notable features of the design include key-dependent S-boxes and a highly complex key schedule.

Blowfish



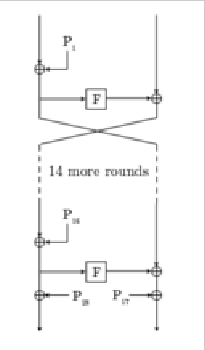
The round function (Feistel function) of Blowfish

General	
Designers	Bruce Schneier
First published	1993
Successors	Twofish
Cipher detail	
Key sizes	32–448 bits
Block sizes	64 bits
Structure	Feistel network
Rounds	16
Best public cryptanalysis	
Four rounds of Blowfish are susceptible to a second-order differential attack (Rijmen, 1997);[1] for a class of weak keys, 14 rounds of Blowfish can be distinguished from a pseudorandom permutation (Vaudenay, 1996).	

Contents [hide]

The algorithm [edit]

Blowfish has a 64-bit block size and a variable key length from 32 bits up to 448 bits.[2] It is a 16-round Feistel cipher and uses large key-dependent S-boxes. In structure it resembles CAST-128, which uses fixed S-boxes.



The Feistel structure of Blowfish

The diagram to the left shows the action of Blowfish. Each line represents 32 bits. The algorithm keeps two subkey arrays: the 18-entry P-array and four 256-entry S-boxes. The S-boxes accept 8-bit input and produce 32-bit output. One entry of the P-array is used every round, and after the final round, each half of the data block is XORed with one of the two remaining unused P-entries.

The diagram to the upper right shows Blowfish's F-function. The function splits the 32-bit input into four eight-bit quarters, and uses the quarters as input to the S-boxes. The outputs are added modulo 2^{32} and XORed to produce the final 32-bit output.

Decryption is exactly the same as encryption, except that P1, P2,..., P18 are used in the reverse order. This is not so obvious because xor is commutative and associative. A common misconception is to use inverse order of encryption as decryption algorithm (i.e. first XORing P17 and P18 to the ciphertext block, then using the P-entries in reverse order).

Blowfish's key schedule starts by initializing the P-array and S-boxes with values derived from the hexadecimal digits of pi, which contain no obvious pattern (see nothing up my sleeve number). The secret key is then, byte by byte, cycling the key if necessary, XORed with all the P-entries in order. A 64-bit all-zero block is then encrypted with the algorithm as it stands. The resultant ciphertext replaces P_1 and P_2. The same ciphertext is then encrypted again with the new subkeys, and the new ciphertext replaces P_3 and P_4. This continues, replacing the entire P-array and all the S-box entries. In all, the Blowfish encryption algorithm will run 521 times to generate all the subkeys - about 4KB of data is processed.

Because the P-array is 576 bits long, and the key bytes are XORed through all these 576 bits during the initialization, many implementations support key sizes up to 576 bits. While this is certainly possible, the 448 bits limit is here to ensure that every bit of every subkey depends on every bit of the key,[2] as the last four values of the P-array don't affect every bit of the ciphertext. This point should be taken in consideration for implementations with a different number of rounds, as even though it increases security against an exhaustive attack, it weakens the security guaranteed by the algorithm. And given the slow initialization of the cipher with each change of key, it is granted a natural protection against brute-force attacks, which doesn't really justify key sizes longer than 448 bits.

```
uint32_t P[18];
uint32_t S[4][256];

uint32_t f (uint32_t x) {
   uint32_t h = S[0][x >> 24] + S[1][x >> 16 & 0xff];
   return ( h ^ S[2][x >> 8 & 0xff] ) + S[3][x & 0xff];
}

void encrypt (uint32_t & L, uint32_t & R) {
   for (int i=0 ; i<16 ; i += 2) {
      L ^= P[i];
      R ^= f(L);
      R ^= P[i+1];
      L ^= f(R);
   }
   L ^= P[16];
   R ^= P[17];
   swap (L, R);
}

void decrypt (uint32_t & L, uint32_t & R) {
   for (int i=16 ; i > 0 ; i -= 2) {
      L ^= P[i+1];
      R ^= f(L);
      R ^= P[i];
      L ^= f(R);
   }
   L ^= P[1];
   R ^= P[0];
   swap (L, R);
}

 {
   // ...
   // initializing the P-array and S-boxes with values derived from pi; omitted in
the example
   // ...
   for (int i=0 ; i<18 ; ++i)
      P[i] ^= key[i % keylen];
   uint32_t L = 0, R = 0;
   for (int i=0 ; i<18 ; i+=2) {
      encrypt (L, R);
      P[i] = L; P[i+1] = R;
   }
   for (int i=0 ; i<4 ; ++i)
      for (int j=0 ; j<256; j+=2) {
         encrypt (L, R);
         S[i][j] = L; S[i][j+1] = R;
      }
}
```

Blowfish in practice [edit]

Blowfish is a fast block cipher, except when changing keys. Each new key requires pre-processing equivalent to encrypting about 4 kilobytes of text, which is very slow compared to other block ciphers. This prevents its use in certain applications, but is not a problem in others.

In one application Blowfish's slow key changing is actually a benefit: the password-hashing method used in OpenBSD uses an algorithm derived from Blowfish that makes use of the slow key schedule; the idea is that the extra computational effort required gives protection against dictionary attacks. *See* key stretching.

Blowfish has a memory footprint of just over 4 kilobytes of RAM. This constraint is not a problem even for older desktop and laptop computers, though it does prevent use in the smallest embedded systems such as early smartcards.

Blowfish was one of the first secure block ciphers not subject to any patents and therefore freely available for anyone to use. This benefit has contributed to its popularity in cryptographic software.

bcrypt is a cross-platform file encryption utility implementing Blowfish developed in 2002.[3] [4][5][6][7]

Weakness and successors [edit]

Blowfish is known to be susceptible to attacks on reflectively weak keys.[8] [9] This means Blowfish users must carefully select keys as there is a class of keys known to be weak, or switch to more modern alternatives like the Advanced Encryption Standard, Salsa20, or Blowfish's more modern successors Twofish and Threefish. Bruce Schneier, Blowfish's creator, is quoted in 2007 as saying "At this point, though, I'm amazed it's still being used. If people ask, I recommend Twofish instead."[10] The FAQ for GnuPG (which features Blowfish as one of its algorithms) recommends that Blowfish should *not* be used to encrypt files that are larger than 4 Gb because of its small 64-bit block size.[11]

See also [edit]

- AES
- Twofish
- Threefish
- MacGuffin

References [edit]

1. ^ Vincent Rijmen (1997). "Cryptanalysis and Design of Iterated Block Ciphers" (PostScript). *Ph.D thesis*.
2. ^ *a* *b* Bruce Schneier (1993). "Description of a New Variable-Length Key, 64-Bit Block Cipher (Blowfish)". *Fast Software Encryption, Cambridge Security Workshop Proceedings* (Springer-Verlag): 191–204.
3. ^ http://bcrypt.sourceforge.net bcrypt file encryption program homepage
4. ^ http://bcrypt463065.android.informer.com/
5. ^ http://www.t2-project.org/packages/bcrypt.html
6. ^ https://docs.oracle.com/cd/E51849_01/gg-winux/OGGLC/ogglc_licenses.htm
7. ^ http://www.solvusoft.com/en/file-extensions/file-extension-bfe/
8. ^ Tom Gonzalez (January 2007). "A Reflection Attack on Blowfish" (PDF). JOURNAL OF LATEX CLASS FILES.
9. ^ Orhun Kara and Cevat Manap (March 2007). "A New Class of Weak Keys for Blowfish" (PDF). FSE 2007.
10. ^ Dahna, McConnachie (2007-12-27). "Bruce Almighty: Schneier preaches security to Linux faithful". *Computerworld*. p. 3. Archived from the original on 2014-04-20. "At this point, though, I'm amazed it's still being used. If people ask, I recommend Twofish instead."
11. ^ http://www.gnupg.org/faq/gnupg-faq.html#define_fish

External links [edit]

- Bruce Schneier. "The Blowfish Encryption Algorithm".
- Bruce Schneier. "Products that Use Blowfish".
- "Standard Cryptographic Algorithm Naming: Blowfish".

Wikimedia Commons has media related to ***Blowfish (cipher)***.

v · t · e	**Block ciphers (security summary)**
Common algorithms	AES · **Blowfish** · DES (Internal Mechanics, Triple DES) · Serpent · Twofish
Less common algorithms	Camellia · CAST-128 · IDEA · RC2 · RC5 · SEED · ARIA · Skipjack · TEA · XTEA
Other algorithms	3-Way · Akelarre · Anubis · BaseKing · BassOmatic · BATON · BEAR and LION · CAST-256 · Chiasmus · CIKS-1 · CIPHERUNICORN-A · CIPHERUNICORN-E · CLEFIA · CMEA · Cobra · COCONUT98 · Crab · Cryptomeria/C2 · CRYPTON · CS-Cipher · DEAL · DES-X · DFC · E2 · FEAL · FEA-M · FROG · G-DES · GOST · Grand Cru · Hasty Pudding cipher · Hierocrypt · ICE · IDEA NXT · Intel Cascade Cipher · Iraqi · KASUMI · KeeLoq · KHAZAD · Khufu and Khafre · KN-Cipher · Ladder-DES · Libelle · LOKI (97, 89/91) ·

	Lucifer · M6 · M8 · MacGuffin · Madryga · MAGENTA · MARS · Mercy · MESH · MISTY1 · MMB · MULTI2 · MultiSwap · New Data Seal · NewDES · Nimbus · NOEKEON · NUSH · PRESENT · Q · RC6 · REDOC · Red Pike · S-1 · SAFER · SAVILLE · SC2000 · SHACAL · SHARK · Simon · SMS4 · Speck · Spectr-H64 · Square · SXAL/MBAL · Threefish · Treyfer · UES · Xenon · xmx · XXTEA · Zodiac
Design	Feistel network · Key schedule · Lai-Massey scheme · Product cipher · S-box · P-box · SPN · Avalanche effect · Block size · Key size · Key whitening (Whitening transformation)
Attack (cryptanalysis)	Brute-force (EFF DES cracker) · MITM (Biclique attack, 3-subset MITM attack) · Linear (Piling-up lemma) · Differential (Impossible · Truncated · Higher-order) · Differential-linear · Integral/Square · Boomerang · Mod *n* · Related-key · Slide · Rotational · Timing · XSL · Interpolation · Partitioning · Davies' · Rebound · Weak key · Tau · Chi-square · Time/memory/data tradeoff
Standardization	AES process · CRYPTREC · NESSIE
Utilization	Initialization vector · Mode of operation · Padding

Cryptography

History of cryptography · Cryptanalysis · Cryptography portal · Outline of cryptography

Symmetric-key algorithm · Block cipher · Stream cipher · Public-key cryptography · Cryptographic hash function · Message authentication code · Random numbers · Steganography

Categories: Block ciphers | Feistel ciphers | Free ciphers