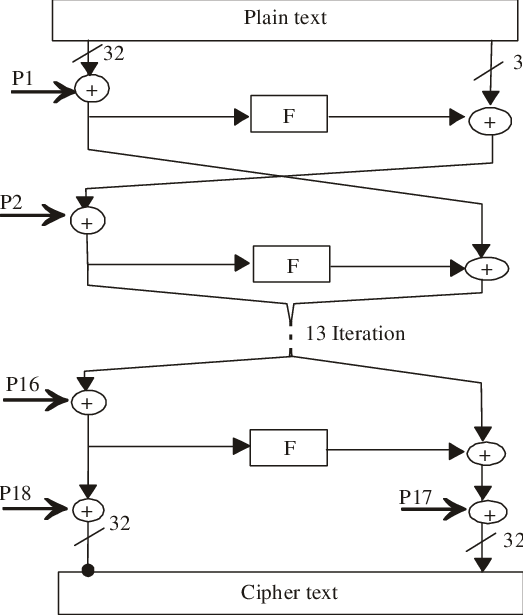
**Blowfish** is a [symmetric-key](https://en.wikipedia.org/wiki/Symmetric-key_algorithm) [block cipher](https://en.wikipedia.org/wiki/Block_cipher), designed in 1993 by [Bruce Schneier](https://en.wikipedia.org/wiki/Bruce_Schneier) and included in many cipher suites and encryption products. Blowfish provides a good encryption rate in software and no effective [cryptanalysis](https://en.wikipedia.org/wiki/Cryptanalysis) of it has been found to date. However, the [Advanced Encryption Standard](https://en.wikipedia.org/wiki/Advanced_Encryption_Standard) (AES) now receives more attention, and Schneier recommends [Twofish](https://en.wikipedia.org/wiki/Twofish" \o "Twofish) for modern applications.[[2]](https://en.wikipedia.org/wiki/Blowfish_(cipher)#cite_note-schneier-interview-dec-2007-2)

Schneier designed Blowfish as a general-purpose algorithm, intended as an alternative to the aging [DES](https://en.wikipedia.org/wiki/Data_Encryption_Standard) 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](https://en.wikipedia.org/wiki/Patent) 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](https://en.wikipedia.org/wiki/Public_domain), and can be freely used by anyone."[[3]](https://en.wikipedia.org/wiki/Blowfish_(cipher)#cite_note-blowfish-paper-3)

Blowfish has a 64-bit [block size](https://en.wikipedia.org/wiki/Block_size_(cryptography)) and a variable [key length](https://en.wikipedia.org/wiki/Key_length) from 32 bits up to 448 bits.[[3]](https://en.wikipedia.org/wiki/Blowfish_(cipher)#cite_note-blowfish-paper-3) It is a 16-round [Feistel cipher](https://en.wikipedia.org/wiki/Feistel_cipher) and uses large key-dependent [S-boxes](https://en.wikipedia.org/wiki/Substitution_box). In structure it resembles [CAST-128](https://en.wikipedia.org/wiki/CAST-128), which uses fixed S-boxes.



The Feistel structure of Blowfish

The adjacent diagram shows Blowfish's encryption routine. Each line represents 32 bits. There are five subkey-arrays: one 18-entry P-array (denoted as K in the diagram, to avoid confusion with the Plaintext) and four 256-entry S-boxes (S0, S1, S2 and S3).

Every round *r* consists of 4 actions:

|  |  |
| --- | --- |
| **Action 1** | XOR the left half (L) of the data with the *r* th P-array entry |
| **Action 2** | Use the XORed data as input for Blowfish's F-function |
| **Action 3** | XOR the F-function's output with the right half (R) of the data |
| **Action 4** | Swap L and R |

The F-function splits the 32-bit input into four eight-bit quarters, and uses the quarters as input to the S-boxes. The S-boxes accept 8-bit input and produce 32-bit output. The outputs are added [modulo](https://en.wikipedia.org/wiki/Modular_arithmetic) 232 and XORed to produce the final 32-bit output (see image in the upper right corner).[[4]](https://en.wikipedia.org/wiki/Blowfish_(cipher)#cite_note-4)

After the 16th round, undo the last swap, and XOR L with K18 and R with K17 (output whitening).

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](https://en.wikipedia.org/wiki/Key_schedule) starts by initializing the P-array and S-boxes with values derived from the [hexadecimal](https://en.wikipedia.org/wiki/Hexadecimal) digits of [pi](https://en.wikipedia.org/wiki/Pi), which contain no obvious pattern (see [nothing up my sleeve number](https://en.wikipedia.org/wiki/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 P1 and P2. The same ciphertext is then encrypted again with the new subkeys, and the new ciphertext replaces P3 and P4. 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. The reason for that is a discrepancy between the original Blowfish description, which uses 448-bit keys, and its reference implementation, which uses 576-bit keys. The test vectors for verifying third party implementations were also produced with 576-bit keys. When asked which Blowfish version is the correct one, Bruce Schneier answered: "The test vectors should be used to determine the one true Blowfish".

Another opinion is that the 448 bits limit is present to ensure that every bit of every subkey depends on every bit of the key,[[3]](https://en.wikipedia.org/wiki/Blowfish_(cipher)#cite_note-blowfish-paper-3) 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.

**RC5**

In [cryptography](https://en.wikipedia.org/wiki/Cryptography), **RC5** is a [symmetric-key](https://en.wikipedia.org/wiki/Symmetric-key_algorithm) [block cipher](https://en.wikipedia.org/wiki/Block_cipher) notable for its simplicity. Designed by [Ronald Rivest](https://en.wikipedia.org/wiki/Ron_Rivest) in 1994,[[2]](https://en.wikipedia.org/wiki/RC5#cite_note-fse1994-2) *RC* stands for "Rivest Cipher", or alternatively, "Ron's Code" (compare [RC2](https://en.wikipedia.org/wiki/RC2) and [RC4](https://en.wikipedia.org/wiki/RC4_(cipher))). The [Advanced Encryption Standard](https://en.wikipedia.org/wiki/Advanced_Encryption_Standard) (AES) candidate [RC6](https://en.wikipedia.org/wiki/RC6) was based on RC5.

**Description**

Unlike many schemes, RC5 has a variable [block size](https://en.wikipedia.org/wiki/Block_size_(cryptography)) (32, 64 or 128 [bits](https://en.wikipedia.org/wiki/Bit)), [key size](https://en.wikipedia.org/wiki/Key_size) (0 to 2040 bits) and number of rounds (0 to 255). The original suggested choice of parameters were a block size of 64 bits, a 128-bit key and 12 rounds.

A key feature of RC5 is the use of data-dependent rotations; one of the goals of RC5 was to prompt the study and evaluation of such operations as a [cryptographic primitive](https://en.wikipedia.org/wiki/Cryptographic_primitive). RC5 also consists of a number of [modular](https://en.wikipedia.org/wiki/Modular_arithmetic) additions and [eXclusive OR (XOR)s](https://en.wikipedia.org/wiki/XOR" \o "XOR). The general structure of the algorithm is a [Feistel](https://en.wikipedia.org/wiki/Feistel_cipher)-like network. The encryption and decryption routines can be specified in a few lines of code. The key schedule, however, is more complex, expanding the key using an essentially [one-way function](https://en.wikipedia.org/wiki/One-way_function) with the binary expansions of both [e](https://en.wikipedia.org/wiki/E_(mathematical_constant)) and the [golden ratio](https://en.wikipedia.org/wiki/Golden_ratio) as sources of "[nothing up my sleeve numbers](https://en.wikipedia.org/wiki/Nothing_up_my_sleeve_number)". The tantalising simplicity of the algorithm together with the novelty of the data-dependent rotations has made RC5 an attractive object of study for cryptanalysts. The RC5 is basically denoted as RC5-w/r/b where w=word size in bits, r=number of rounds, b=number of 8-bit bytes in the key.

**Encryption**

Encryption involved several rounds of a simple function. 12 or 20 rounds seem to be recommended, depending on security needs and time considerations. Beyond the variables used above, the following variables are used in this algorithm:

* A, B - The two words composing the block of plaintext to be encrypted.

A = A + S[0]

B = B + S[1]

**for** i = 1 to r do:

A = ((A ^ B) <<< B) + S[2 \* i]

B = ((B ^ A) <<< A) + S[2 \* i + 1]

*# The ciphertext block consists of the two-word wide block composed of A and B, in that order.*

**return** A, B

The example C code given by Rivest is this.

void RC5\_ENCRYPT(WORD \*pt, WORD \*ct)

{

WORD i, A = pt[0] + S[0], B = pt[1] + S[1];

**for** (i = 1; i <= r; i++)

{

A = ROTL(A ^ B, B) + S[2\*i];

B = ROTL(B ^ A, A) + S[2\*i + 1];

}

ct[0] = A; ct[1] = B;

}

**Decryption**

Decryption is a fairly straightforward reversal of the encryption process. The below pseudocode shows the process.

**for** i = r down to 1 do:

B = ((B - S[2 \* i + 1]) >>> A) ^ A

A = ((A - S[2 \* i]) >>> B) ^ B

B = B - S[1]

A = A - S[0]

**return** A, B

The example C code given by Rivest is this.

void RC5\_DECRYPT(WORD \*ct, WORD \*pt)

{

WORD i, B=ct[1], A=ct[0];

**for** (i = r; i > 0; i--)

{

B = ROTR(B - S[2\*i + 1], A) ^ A;

A = ROTR(A - S[2\*i], B) ^ B;

}

pt[1] = B - S[1]; pt[0] = A - S[0];

}

REFERENCES

<http://www.brainkart.com/article/Classical-Encryption-Techniques_8339/>

<https://www.tutorialspoint.com/cryptography/index.htm>

<https://www.geeksforgeeks.org/cryptography-introduction/>

[https://www.techopedia.com/definition/1770/cryptography#:~:text=Cryptography%20involves%20creating%20written%20or,information%20to%20be%20kept%20secret.&text=Information%20security%20uses%20cryptography%20on,transit%20and%20while%20being%20stored](https://www.techopedia.com/definition/1770/cryptography).

<https://www2.slideshare.net/lineking/classical-encryption-techniques-in-network-security?qid=e388c29f-793d-4f2b-bcaf-9d22e9ca07b5&v=&b=&from_search=1>

<https://www.researchgate.net/publication/334418542_A_Review_Paper_on_Cryptography>

<https://www.academia.edu/Documents/in/Cryptography>