Блочный шифратор *Blowfish*

Мойсейчик Е. С.

Алгоритм шифрования. Алгоритм шифрования *Blowfish* построен аналогично блочному шифрованию *DES* на сетях Фейстеля и характеризуется следующими параметрами:

- *Размер блока*: 64 бит (8 байт).
- *Размер ключа*: 32 – 448 бит (4 – 56 байт).
- *Количество P-блоков (подключей)*: 18.
- *Количество раундов*: 16.
- *Количество S-блоков*: 4 (каждый по 512 записей в 32 бита).

В общем случае алгоритм шифрования состоит из 2 этапов: расширение ключа и шифрование данных. Опишем поэтапно данный процесс.

Шаг 1. Инициализация *P* и *S*-блоков.

Первоначально блоки $P_1 - P_{18}$ инициализируются некоторой фиксированной строкой, обычно состоящей из шестнадцатеричных цифр мантиссы числа Пи. Выбор строки может быть произвольным, ключевым является свойство её независимости от входных данных шифратора.

Далее для расширения ключа производится операция *XOR* над P_1 с первыми 32 битами ключа *K*, над P_2 со вторыми 32 битами и так далее. Если ключ *K* короче, то он накладывается циклически.

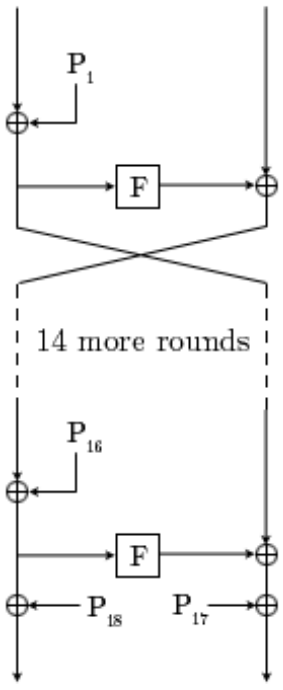
Блоки *S* представляют собой 32-битные таблицы замен вида:

$$S_1[0], S_1[1], \dots, S_1[255]$$

$$S_2[0], S_2[1], \dots, S_2[255]$$

$$S_3[0], S_3[1], \dots, S_3[255]$$

$$S_4[0], S_4[1], \dots, S_4[255]$$



Шаг 2. Шифрование входного блока.

1. Входной 64-битный блок разделяется на 2 32-битных блока L_0, R_0.

2. Для i = 1…16:

$$L_i = L_{i-1} \oplus P_i$$
$$R_i = R_{i-1} \oplus F(L_i)$$

3. После каждого раунда 32-битные блоки меняются местами.

4. В конце к получившимся 32-битным блоками прибавляются блоки P_{17}, P_{18}.

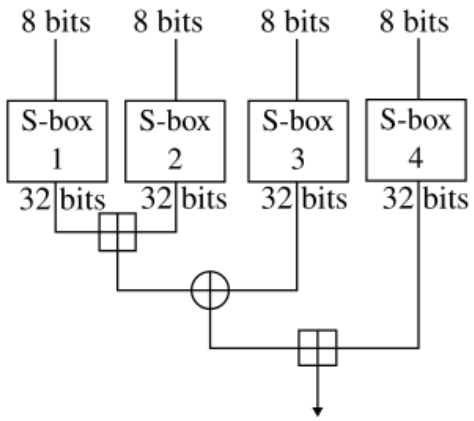
$$L_{17} = L_{16} \oplus P_{18}$$
$$R_{17} = R_{16} \oplus P_{17}$$

5. Выходной блок равен объединению L_{17} и R_{17}.

Раундовая функция зашифрования F представляет собой следующее преобразование:

1. 32-битный блок делится на 4 8-битных блока (X_1, X_2, X_3, X_4), каждый из которых является индексом соответствующего массива таблицы замен $S_1 - S_4$.

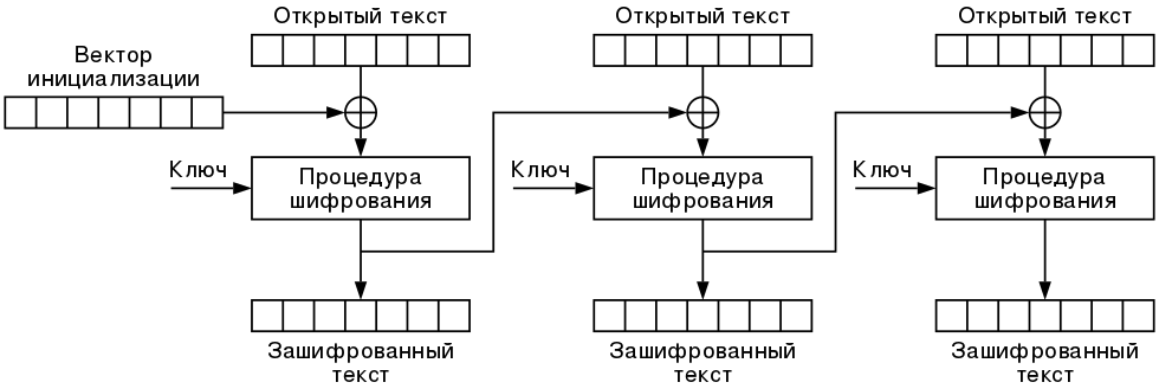
2. Далее полученные $S_1[X_1]$ и $S_2[X_2]$ складываются по модулю 2^{32}, после чего значение складывается по модулю 2 с $S_3[X_3]$ и, наконец, с $S_4[X_4]$ по модулю 2^{32}.



Алгоритм расшифрования. Расшифрование данным алгоритмом сводится к применению того же шифратора с подачей подключей $P_{17} - P_1$ в обратном порядке. После просеивания через 16 раундов правую и левую часть аналогично складывают по модулю 2 с ключами P_0 и P_1 соответственно.

Режим сцепленного шифрования блоков. Как и для всех аналогичных блочных шифраторов, для построенного шифратора характерно применение различных режимов шифрования.

Т.к. режим шифрования *EBC*, где происходит последовательное деление открытого текста на блоки и их независимое шифрование, может давать нежелательное сохранение статистических особенностей открытого текста, будем использовать режим *CBC*, суть которого опишем ниже.



Сообщения разбиваются на блоки одинакового размера. Шифрование каждого следующего блока выполняется с использованием предыдущего зашифрованного блока. Для первого блока в качестве такового выступает так называемый вектор инициализации, который устанавливается в начале шифрования и заполнятся случайными числами.

Т.о. в функцию шифрования каждый раз передаётся сумма по модулю 2 текущего блока сообщения и предыдущего зашифрованного. Расшифровка выполняется с использованием тех же ключа и вектора инициализации в обратном порядке.

Тестирование выходных последовательностей. Для тестирования построенного блочного шифратора применим указанные в задании сценарии генерации входных данных и набор тестов стандарта NIST.

Тестирование шифратора будем производить в режиме *EBC,* чтобы выявить основные особенности данной системы блочного шифрования.

Сценарий / Тест	Произвольный текст и ключ	Блок с малым весом Хэмминга	Блок с большим весом Хэмминга	Ключ с малым весом Хэмминга	Ключ с большим весом Хэмминга	Размножение ошибки в ключе	Размножение ошибки в открытом тексте	Корреляция открытого и шифр-текста	Режим цепочной обработки
Frequency (Monobits) Test	SUCCESS p_value = 0,5745221	FAILURE	FAILURE	SUCCESS p_value = 0,696424382	SUCCESS p_value = 0,9586959859	FAILURE	SUCCESS p_value = 0,20167381792	SUCCESS p_value = 0,514043463651	SUCCESS p_value = 0,925727129167
Runs Test	SUCCESS p_value = 0,4604707	FAILURE	FAILURE	SUCCESS p_value = 0,330941468	SUCCESS p_value = 0,7744410749	FAILURE	SUCCESS p_value = 0,57443499056	SUCCESS p_value = 0,108201617459	SUCCESS p_value = 0,397219427756529
Serial Test	SUCCESS p_value1 = 0,5833001	FAILURE	FAILURE	SUCCESS p_value1 = 0,05746265	SUCCESS p_value1 = 0,5724937482	FAILURE	SUCCESS p_value1 = 0,32263607195	SUCCESS p_value1 = 0,718424619353	SUCCESS p_value1 = 0,791011060509
	SUCCESS p_value2 = 0,56443007			SUCCESS p_value2 = 0,06096533	SUCCESS p_value2 = 0,20960357151		SUCCESS p_value2 = 0,70217028636	SUCCESS p_value2 = 0,499245289251	SUCCESS p_value2 = 0,442884821036
Binary Matrix Rank Test	SUCCESS p_value = 0,7880926	FAILURE	FAILURE	SUCCESS p_value = 0,67104808	SUCCESS p_value = 0,39402310347	SUCCESS p_value = 0,6128804906	SUCCESS p_value = 0,140196717268	SUCCESS p_value = 0,687277573971	SUCCESS p_value = 0,0953224716688299
Random Excursions Test	FAILURE	FAILURE	FAILURE	FAILURE	FAILURE	FAILURE	FAILURE	FAILURE	FAILURE
Linear Complexity Test	SUCCESS p_value = 0,264502628	FAILURE	FAILURE	SUCCESS p_value = 0,1206489677	SUCCESS p_value = 0,237530641575	SUCCESS p_value = 0,09734256954	SUCCESS p_value = 0,209646962797	SUCCESS p_value = 0,465921678640	SUCCESS p_value = 0,109713646007462
Test for the Longest Run of Ones in a Block	SUCCESS{0} p_value = 0,8521839	FAILURE	FAILURE	SUCCESS{0} p_value = 0,01682300023	SUCCESS{0} p_value = 0,41190141318	SUCCESS{0} p_value = 0,8577944689	SUCCESS{0} p_value = 0,67637406729	SUCCESS{0} p_value = 0,587112515099	SUCCESS{0} p_value = 0,624341122032951
Maurer's "Universal Statistical" Test	SUCCESS p_value = 0,0805954	FAILURE	FAILURE	SUCCESS p_value = 0,70236767299	SUCCESS p_value = 0,55337048781	FAILURE p_value = 1,76532174333916E-41	SUCCESS p_value = 0,981617277618233	SUCCESS p_value = 0,228855906329	SUCCESS p_value = 0,140285288696691

Cumulative Sums (Cusum) Test	SUCCESS p_value = 0,478641368	FAILURE	FAILURE	SUCCESS p_value = 0,6675931059	SUCCESS p_value = 0,97776490686	FAILURE p_value = 1,434725232307 7E-05	SUCCESS p_value = 0,086448164585 1951	SUCCESS p_value = 0,681452881269 361	SUCCESS p_value = 0,918123525240 887

Как и ожидалось для данного шифратора в режиме *EBC* он проявляет плохие свойства рассеивания и перемешивания текста, что было выявлено при тестировании сценария блоков открытого текста с малым и большим весом Хэмминга.

Также частично была замечена слабость данного шифра к размножению ошибки в ключе, что было выявлено частью тестов.

В режиме *CBC* данный шифратор приобретает лучшие свойства рассеивания и перемешивания информации и успешно проходит тесты на малый и большой вес Хэмминга, однако остаётся более уязвимым для тестов с размножением ошибки.

Т.о. данный шифратор обеспечивает лучшую криптостойкость, чем аналогичный шифратор *DES* за счёт увеличения длины блока и переменной длины ключа, которая может быть довольно большой. Также данный шифратор достаточно просто в реализации и быстр, если нет необходимости каждый раз производить подготовительный этап с инициализацией блоков.

В целом можно сделать вывод, что данный блочный шифратор показал достаточно хорошие свойства в шифровании и может быть использован во многих сферах, однако там, где требуется обеспечивать высокую надёжность, стоит отказаться от использования режима шифрования *EBC* в пользу других, или же использовать модификации данного алгоритма шифрования, такие как *Twofish,* выпущенные данным авторов для улучшения криптостойкости и характеристик данной системы.