**Описание и уязвимости.**

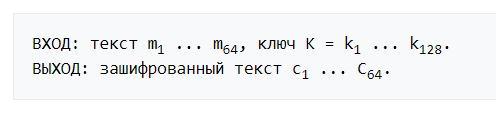
CAST — это 64-битовый шифр, допускающий размеры ключа вплоть до 128 бит и строится на основе данных в S блоках.

Алгоритм CAST использует 64-битовый блок и 128-битовый ключ. CAST устойчив к дифференциальному и линейному криптоанализу. Сила алгоритма CAST заключена в его S-блоках. У CAST нет фиксированных S-блоков и для каждого приложения они конструируются заново. Созданный для конкретной реализации CAST S-блок уже больше никогда не меняется. Другими словами, S-блоки зависят от реализации, а не от ключа.

Из всего выше сказанного следует, что у CAST нету уязвимостей, но это не значит, что у него нету недостатков.

Одним из весомых недостатков является медленная скорость шифрования, связанная с разверткой ключа на каждый блок, а также высокие требования к оперативной памяти.

**Алгоритм.**



1. CAST-128 использует пару подключей за раунд: 32-битные величины Km используется в качестве "маскировки" ключа и Kr используют как "перестановки" ключа, из которых используются только начальные 5-бит.

2. Развертка ключа составляет 16 пар подключей {Kmi, Kri} полученных из K.

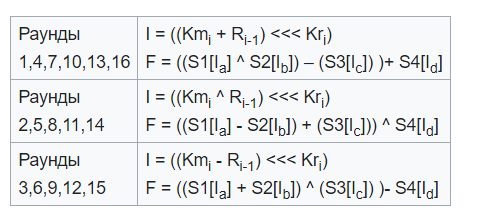
3. (L0, R0) <- (m1. .. m64). (Разделяет текст на левую и правую 32-битные половины L0 = m1 ... m32 и R0 = m33 ... m64).

4. (16 раундов) for i from 1 to 16, вычислить Li и Ri следующим образом: Li = Ri-1; Ri = Li-1 ^ F(Ri-1,Kmi,Kri), где F определяется при реализации ключа.

5. c1 ... c64 <- (R16, L16). (Меняем окончательные блоки местами L16, R16 и объединяем, чтобы сформировать зашифрованный текст.)

Расшифрование совпадает с алгоритмом шифрования, приведенным выше, кроме того, что раунды (и, следовательно, пары подключей), используются в обратном порядке, чтобы вычислить (L0, R0) из (R16, L16).

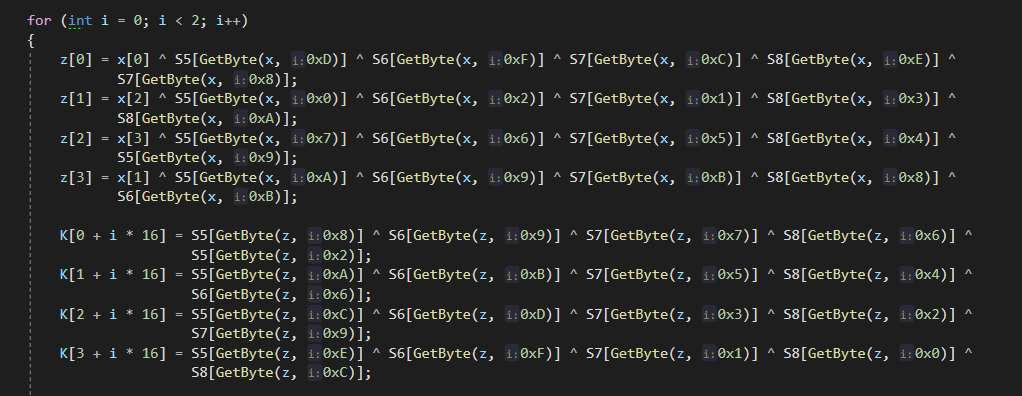
Моя криптосистема реализует алгоритм, описанный в статье <https://tools.ietf.org/html/rfc2144>, где F для раундов определены следующим образом:

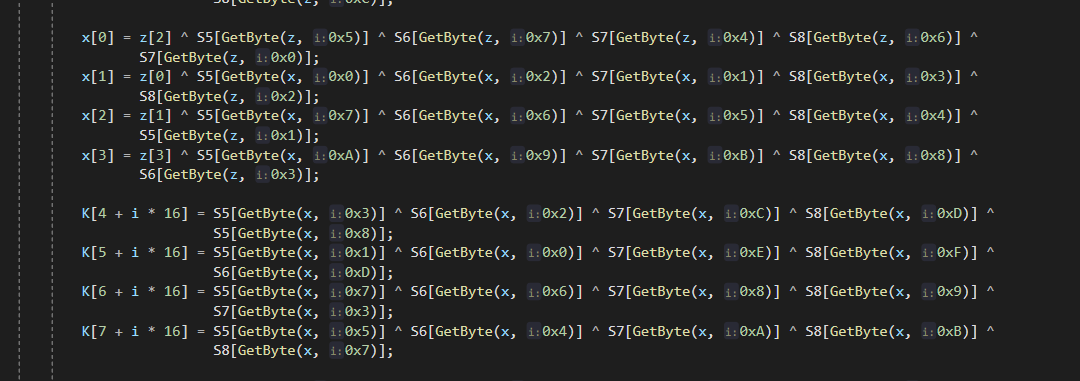


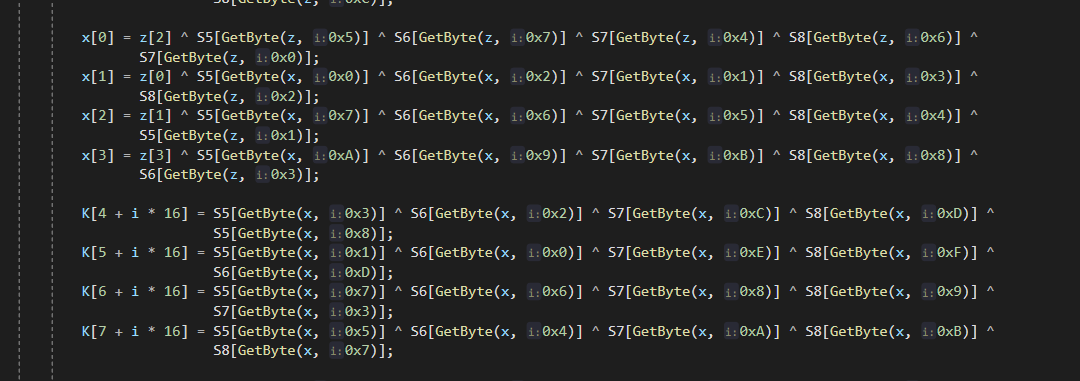
S блоки также брались от туда же.

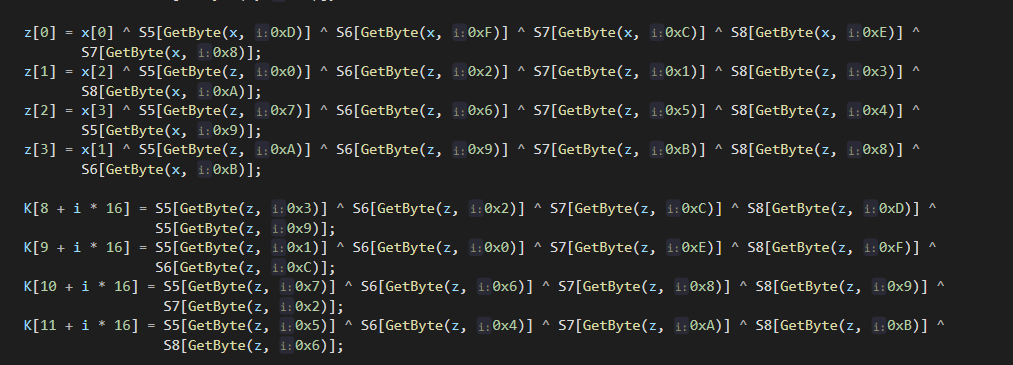
**Код.**

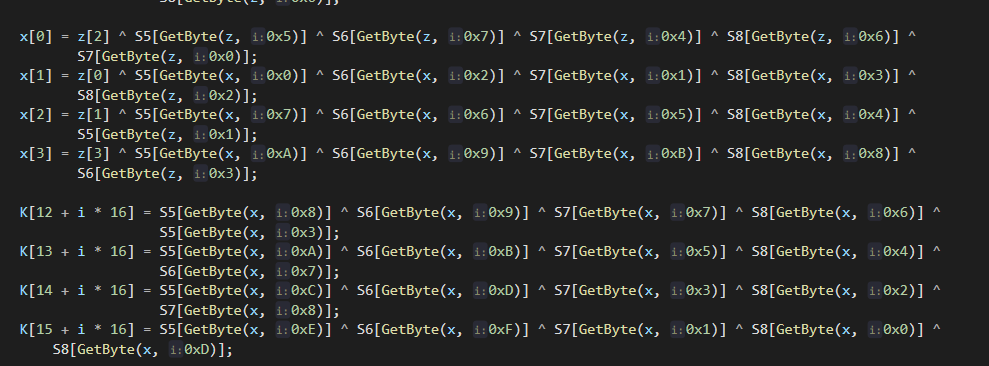
Развертка.



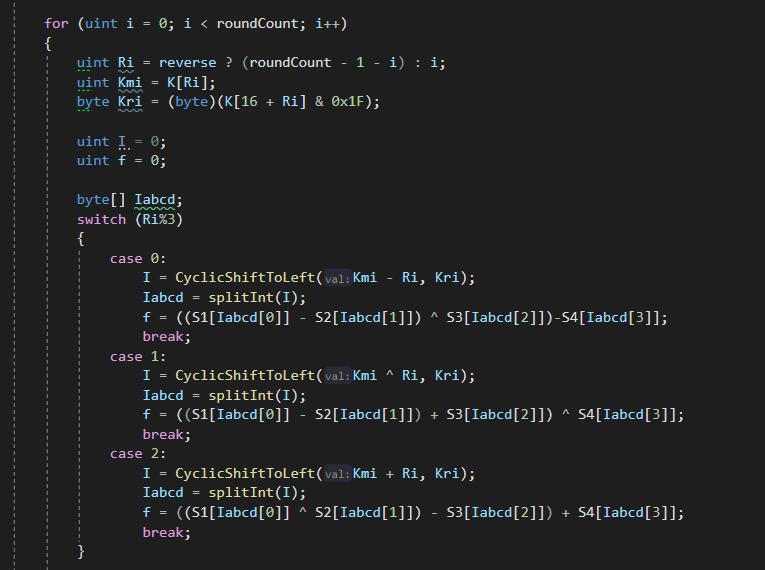


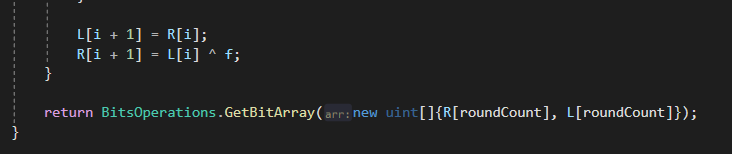




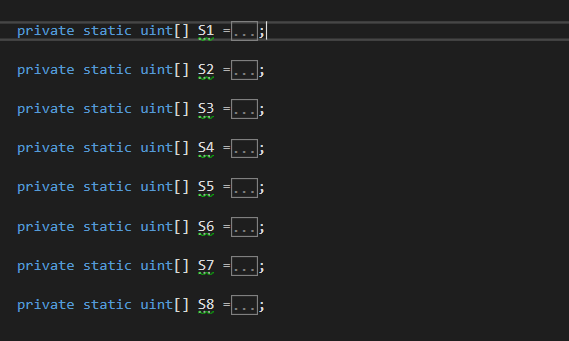


Шифрование блока.





S блоки.



**Тесты.**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **The longest ones run test** | **Runs** | **Frecuency monobit** | **Binary Matrix Rank** | **Universal** | **Liner complexity** | **Serial** | **Cumulative sums** | **Random excursions** | **Rank test** |
| **Probability Test** | true | false | true | true | false | false | false | false | true | true |
| **Probability Test Low Hamming Weight Block** | true | false | true | true | false | false | false | false | true | true |
| **Probability Test Hight Hamming Weight Block** | true | false | true | true | false | true | true | true | true | true |
| **Probability Test Low Hamming Weight Key** | true | false | false | false | false | false | false | false | true | true |
| **Probability Test High Hamming Weight Key** | true | false | false | false | false | false | false | false | true | true |
| **Error Spread With Key Changing Test** | true | true | true | true | false | true | true | false | true | true |
| **Error Spread With Block Changing Test** | true | true | true | true | false | true | true | false | true | true |
| **Simple Replace Test** | true | false | false | false | false | true | true | false | true | true |
| **Chain Processing Test** | true | false | false | false | false | true | true | false | true | true |