**Линейный криптоанализ**

**Построение линейного пути для XSPL-шифра**

Построение линейного пути для заданных исходных значений

α = Матрица L=

∏= (15,9,1,7,13,12,2,8,6,5,14,3,0,11,4,10)

1. **Описание структуры шифра**

XSPL-шифр

Этот шифр состоит из следующих преобразований:

* сложение по модулю 2 с ключом;
* преобразование замены или подстановки. Обозначается S-преобразование;
* преобразование перестановки. Обозначается P-преобразование;
* линейное преобразование. Обозначается L-преобразование.

**1.1 Подстановка**

В нашем шифре мы разбиваем 36-битный блок данных на девять 4-битных подблока. Каждый подблок формирует вход в S-бокс 9×9 (подстановка с 4 входными и 4 выходными битами), который может быть легко реализован с помощью табличного поиска шестнадцати 4-битных значений, индексированных целым числом, представленным 4 входными битами. Наиболее фундаментальным свойством S-box является то, что он является нелинейным отображением, т. е. выходные биты не могут быть представлены в виде линейной операции над входными битами.

Для нашего шифра мы будем использовать одно и то же нелинейное отображение для всех S-блоков.

Атаки линейного и дифференциального криптоанализа одинаково применимы к тому, существует ли одно отображение или все S-боксы являются различными отображениями. Отображение, выбранное для нашего шифра, приведено в таблице 1.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| input | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| output | 15 | 9 | 1 | 7 | 13 | 12 | 2 | 8 | 6 | 5 | 14 | 3 | 0 | 11 | 4 | 10 |

**Таблица 1.** S-box (in hexademical)

Рассмотрим S-образное представление с входом X (подблок i) = [X1 X2 X3 X4] и соответствующим выходом Y (подблок i) = [Y1 Y2 Y3 Y4]. Все линейные аппроксимации могут быть исследованы для определения их полезности путем вычисления вероятностного смещения для каждого. Следовательно, мы рассматриваем все выражения вида уравнения (1), где X-вход и Y - выход S-бокса.

Xₙ₁+Xₙ₁+… + Xₙₘ+Yₚ₁+Yₚ₂ +…+Yₚₜ=0, где «+»- сложение по модулю два

X1 X2 X3 X4

4x4

S-box

Y1 Y2 Y3 Y4

**LAT (Linear Approximation Table)**

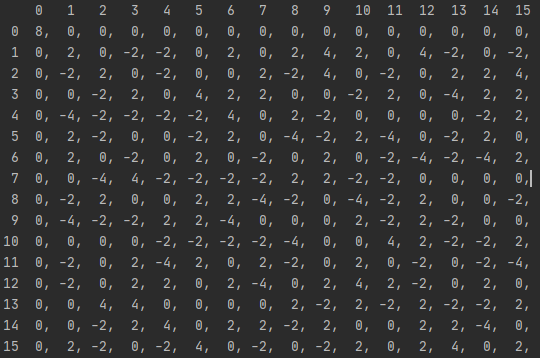
Полное перечисление всех линейных аппроксимаций S-бокса в нашем шифре приведено в таблице линейных аппроксимаций таблицы 4. Каждый элемент таблицы представляет собой число совпадений между линейным уравнением, представленным в шестнадцатеричном виде как "Входная сумма" и сумма выходных битов, представленных в шестнадцатеричном виде как "Выходная сумма" минус 8. Следовательно, деление значения элемента на 16 дает вероятностное смещение для конкретной линейной комбинации входных и выходных битов. Шестнадцатеричное значение, представляющее собой сумму, при просмотре в двоичном виде значение указывает на переменные, участвующие в сумме. Для линейной комбинация входных переменных, представленных как: a₁⋅Х₁ ⊕ a₂⋅Х₂ ⊕ a₃⋅Х₃ ⊕ a₄⋅Х₄, где aₙ ∈ {0,1} и "⋅" логическое умножение (конъюнкция)

Аналогичным образом, для линейной комбинации выходной последовательности бит - b₁⋅У₁ ⊕ b₂⋅У₂ ⊕ b₃⋅У₃ ⊕ b₄⋅У₄? где BI ∈ {0,1}, в шестнадцатеричной системе счисления представляет двоичный вектор b1b2b3b4.

Следовательно, смещение линейного уравнения X₁⊕ X₃⊕X₄ = Y₂ ⊕ Y₄ (шестнадцатеричный вход 11 и шестнадцатеричный выход 4) равен = , а вероятность того, что линейное уравнение верно 1/2 − 2/8 = 2/8.

Можно отметить некоторые основные свойства таблицы линейной аппроксимации. Например, вероятность того, что любая сумма непустого подмножества выходных битов равна сумме без входных битов, равна ровно 1/2, поскольку любая линейная комбинация выходных битов должна

иметь равное число нулей и единиц для биективного S-бокса. Кроме того, линейная комбинация без выходных битов всегда будет равна линейной комбинации без входных битов, приводящие к смещению +1/2 и табличному значению +8 в верхнем левом углу. Следовательно, верхняя строка таблицы-это все нули, за исключением самого левого значения. Аналогично, первая колонка — это все нули, за исключением самого верхнего значения. Можно также отметить, что сумма любой строки или любого столбца должна быть либо +8, либо -8.



**Таблица 2.** LAT

**1.2 Перестановка**

Раунд перестановки- это просто транспозиция подблоков между собой или перестановка позиций подблоков. Перестановка приведена в таблице 2 (где номера обозначают позиции подблоков, т.е. 1 самый левый подблок (4 бита) и 16- крайний правый). В данном случае, если входной текст представить, как матрицу, состоящую из 9 подблоков 4-битных значений, то данный этап можно объяснить иначе-это транспонирование исходной матрицы.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| input | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| output | 1 | 4 | 7 | 2 | 5 | 8 | 3 | 6 | 9 |

**Таблица 3.** Перестановка

**1.3 Линейное преобразование L**

В рассматриваемом нами шифре под данным этапом подразумевается перемножение входной матрицы на матрицу L.

Матрица L=

**2.Рассмотрим три раунда шифрования:**

**1 Раунд**:

***Шаг 1:*** αxor key

∆₁₁=

***Шаг 2:*** Sbox

Воспользуемся таблицей LAT (Linear Approximation Table).

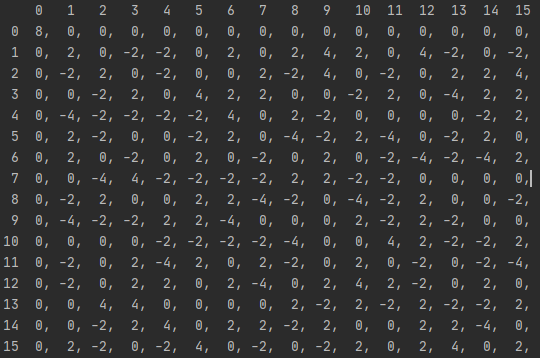


Рис.1--LAT (Linear Approximation Table)

Программный код для реализации LAT таблицы для заданной подстановки приложен далее.

Оба активных полубайта равны 3, значит наиболее подходящими значениями в нашем случае являются β=5;13, у которых преобладание |ε|=4.

Выберем β=13.

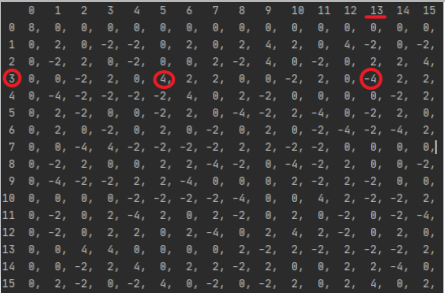


Рис.2 –Выбор значения β

P₁= ()× ()=

∆₁₂=∏(α)=

***Шаг 3***: Перестановка (транспонирование матрицы)

∆₁₃=

***Шаг 4***: Линейное преобразование (умножение на матрицу L)

∆₁₄=×=

**2 Раунд**:

***Шаг 1:*** α(∆₁₄)xor key

∆₂₁=

***Шаг 2:*** Sbox

Воспользуемся LAT (Linear Approximation Table)

Все четыре активных полубайта равны 13, значит наиболее подходящими значениями в нашем случае являются β=2;3, у которых преобладание |ε|=4.

Выберем β=2.

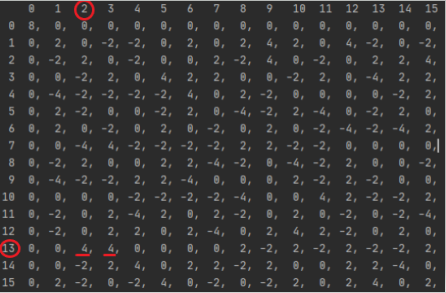


Рис.3 –Выбор значения β

P₂= ()× ()× ()× () =(

∆₂₂=∏(α)=

***Шаг 3***: Перестановка (транспонирование матрицы)

∆₂₃=

***Шаг 4***: Линейное преобразование (умножение на матрицу L)

∆₂₄=×=

**3 Раунд**:

***Шаг 1:*** α(∆₂₄)xor key

∆₃₁=

***Шаг 2:*** Sbox

Воспользуемся LAT (Linear Approximation Table).

Активные полубайта равны 2 и 4. Наиболее подходящими значениями для α=2 являются β=9;13, у которых преобладание |ε|=4. Для α=4 – β=1;6

Выберем β=9 и β=1 соответственно.

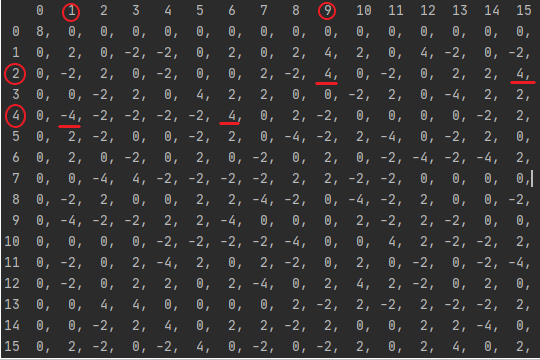


Рис.4 –Выбор значения β

∆₃₂=∏(α)=

P₃= ()× ()× ()× () ×()× () =(

***Шаг 3***: Перестановка (транспонирование матрицы)

∆₃₃=

***Шаг 4***: Линейное преобразование (умножение на матрицу L)

∆₃₄=×=

Итоговая вероятность:

P = P₁×P₂×P₃ ==

То есть нам необходимо 16 777 216 пар (открытый текст – зашифрованный текст), чтобы хотя бы в одном случае линейный путь был равен нашему.

**Приложение**

Программный код, вычисляющий LAT по заданной подстановке ∏= (15,9,1,7,13,12,2,8,6,5,14,3,0,11,4,10)

from array import \*  
S=array('i',[15,9,1,7,13,12,2,8,6,5,14,3,0,11,4,10])  
M=array('i',[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15])  
  
Array\_X = [0] \* 16  
for i in range(0,16):  
 Array\_X[i] = [0] \* 16  
  
print(" ","%4d%4d%4d%4d%4d%4d%4d%4d%4d%4d%5d%4d%4d%4d%4d%4d"%(0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15))  
  
for x in range(0,16):  
 print ('%3d'%x,end="")  
 x1= x%2  
 if (x1==1):  
 X1=x1  
 else:  
 X1 = 0  
  
 x2= (x // 2) % 2  
 if (x2==1):  
 X2 = x2  
 else:  
 X2 = 0  
  
  
 x3=(x//4) % 2  
 if (x3==1):  
 X3 = x3  
 else:  
 X3= 0  
  
  
 x4= x // 8  
 if (x4==1):  
 X4 = x4  
 else:  
 X4=0  
  
 for j in range(0, 16):  
  
 value=X4\*(M[j]//8 ) ^ X3\*( (M[j]//4)%2 ) ^ X2\*( (M[j]// 2) % 2) ^ X1\*( M[j]%2)  
 Array\_X[x][j]=value  
  
 for y in range(0, 16):  
 Sovpadenie = 0  
 y1 = y % 2  
 if (y1 == 1):  
 Y1 = y1  
 else:  
 Y1 = 0  
  
 y2 = (y // 2) % 2  
 if (y2 == 1):  
 Y2 = y2  
 else:  
 Y2 = 0  
  
 y3 = (y // 4) % 2  
 if (y3 == 1):  
 Y3 = y3  
 else:  
 Y3 = 0  
  
 y4 = y // 8  
 if (y4 == 1):  
 Y4 = y4  
 else:  
 Y4 = 0  
   
 for i in range(0, 16):  
  
 P = Y4 \* (S[i] // 8) ^ Y3 \* ((S[i] // 4) % 2) ^ Y2 \* ((S[i] // 2) % 2) ^ Y1 \* (S[i] % 2)  
 if (P == Array\_X[x][i]):  
 Sovpadenie += 1  
 print("%3d" % (Sovpadenie-8),end=",")  
 print('')