Введение

Современные криптосистемы для достижения высокой надежности, как правило, многократно применяют относительно простые криптографические преобразования, такие как предложенные Клодом Шенноном подстановки и перестановки[1], а также операции циклического сдвига или гаммирования.

Однако, помимо широко распространенных алгоритмов, основанных на подобных принципах, имеются и другие подходы к решению проблемы построения надежной криптосистемы. Одним из них является применение клеточных автоматов.

Существует 2 основных метода использования клеточных автоматов в симметричных криптосистемах:

1) Алгоритм шифрования строится исключительно на преобразованиях по правилам клеточных автоматов.

2) Клеточный автомат используется в качестве одного из элементов криптосистемы на отдельном этапе шифрования.

Первый подход подразумевает использование обратного исходному правила клеточного автомата для дешифрации, в связи с чем возникает следующая сложность: необходимо сохранить в тайне прямое и обратное правила, поскольку их раскрытие злоумышленником ведет к возможности атаки шифрованного текста прямым перебором. В связи с этим некоторые авторы[2] предлагают использовать правила клеточного автомата в качестве ключа.

Второй подход является более простым, так как не предполагает обязательного сохранения в тайне правил клеточного автомата.

В данной работе рассматривается симметричная блочная криптосистема, использующая клеточный автомат с необратимыми хаотическими правилами на этапе шифрования ключа. Описывается алгоритм вычисления раундового ключа правилами клеточного автомата. Показаны преимущества метода перед классическими криптографическими

алгоритмами и возможность его практического применения в задачах шифрования.

Алгоритм криптосистемы

Рассмотрим симметричную блочную криптосистему, выполняющую N раундов шифрования над каждым блоком шифротекста. В общем виде она имеет алгоритм, указанный на рисунке 1.

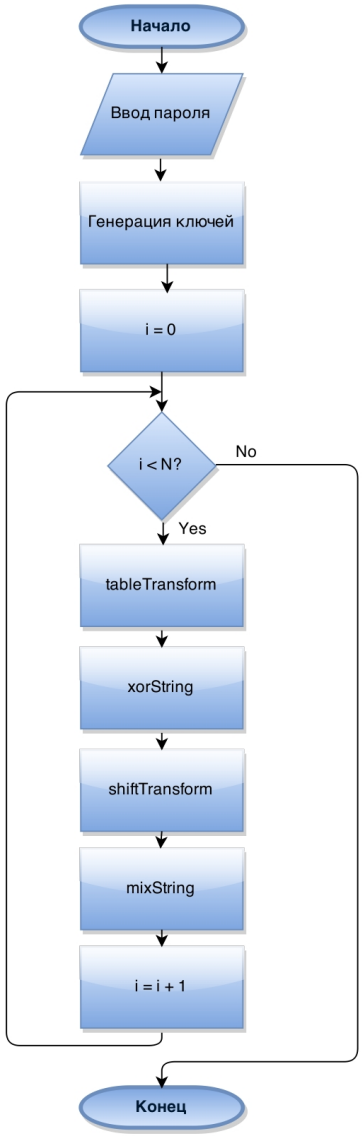

Рисунок 1. Блок-схема алгоритма криптосистемы

Опишем подробно каждый этап алгоритма.

Преобразования с использованием клеточного автомата

Вначале рассмотрим правила преобразования строки в решетку ячеек клеточного автомата, а также состояния клеточного автомата обратно в строку.

Каждый символ строки в соответствующей кодировке может быть рассмотрен как двоичное восьмибитовое число. Таким образом любой символ можно однозначно представить в виде некоторой конфигурации живых и мертвых клеток клеточного автомата. Разобьем решетку ячеек клеточного автомата на квадраты размером 8 на 8 клеток. Запишем в подобные квадраты, начиная с точки с координатами (0, 0), двоичные представления символов строки, располагая по восемь символов в каждом таком квадрате сверху вниз. На рисунке 2 представлен пример преобразования строки в решетку клеточного автомата.

Пример строки	Представление строки в виде клеточного автомата (1 - живая клетка, 0 - мертвая)							
a	0	1	1	0	0	0	0	1
b	0	1	1	0	0	0	1	0
c	0	1	1	0	0	0	1	1
d	0	1	1	0	0	1	0	0
e	0	1	1	0	0	1	0	1
f	0	1	1	0	0	1	1	0
g	0	1	1	0	0	1	1	1
h	0	1	1	0	1	0	0	0

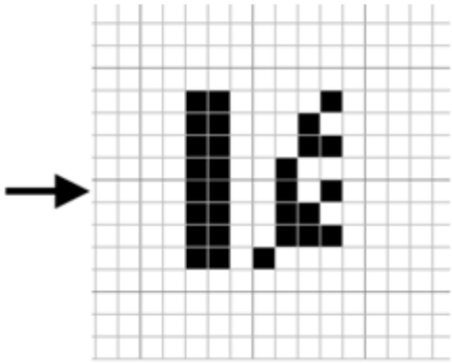

Рисунок 2. Преобразование строки в решетку клеточного автомата (черная клетка - живая, белая - мертвая)

Для преобразования состояния клеточного автомата в строку также разобьем решетку на блоки 8 на 8 ячеек. Затем, начиная с верхнего непустого блока, пойдем по решетке сверху вниз и слева направо, поблочно преобразовывая состояние ячеек в символы и игнорируя пустые участки. Пример подобного преобразования представлен на рисунке 3.

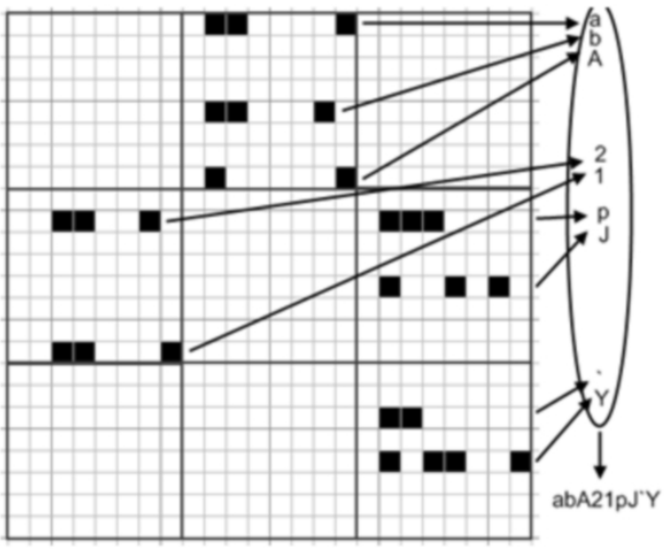

Рисунок 3. Преобразование состояния клеточного автомата (черная клетка - живая, белая - мертвая) в строку

Генерация раундовых ключей

На начальном этапе имеется пароль, которым будет зашифрован текст. Для получения достаточно большого размера ключа и снижения вероятности взлома прямым перебором добавим к исходной строке-паролю добавочную строку, вычисленную путем преобразования пароля в решетку клеточного автомата и генерации 15 поколений по правилу "B12/S5" [3] (мертвая клетка становится живой при количестве соседей 1 или 2, живая остается живой при 5 соседях, используется окрестность Мура порядка 1, такое правило позволяет гарантированно увеличить размер ключа). Полученную таким образом строку назовем password. Каждый i-й раундовый ключ вычисляется путем преобразования строки password в клеточный автомат и генерации 100 + i поколений по правилу "B2/S" (мертвая клетка становится живой при 2 соседях, живая погибает на следующем шаге, используется окрестность Мура порядка 1, данное правило создает хаотические состояния и является необратимым [4]). Длина блока шифротекста устанавливается равной длине нулевого раундового ключа (отсчет ведется с нуля) или длине шифруемого текста, если его длина превышает длину ключа.

Перестановка по таблице tableTransform

Возьмем квадратную таблицу с длиной и шириной, равной округленному в большую строну квадратному корню из длины блока в байтах:

$$tableSize = \lceil \sqrt{length(blockSize)} \rceil$$

Запишем в эту таблицу блок шифротекста построчно, после чего считаем его по столбцам и сохраним вместо исходного блока (рисунок 4).

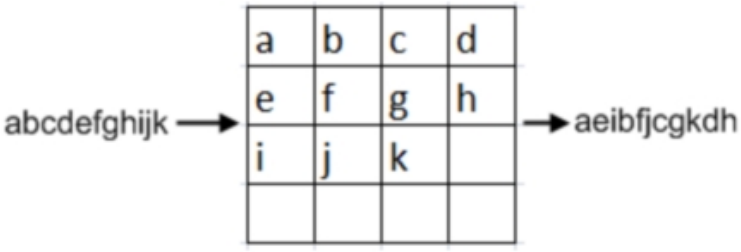

Рисунок 4. Пример перестановки по таблице размером 4 на 4

Сложение с раундовым ключом xorString

На данном этапе происходит сложение с помощью "исключающего или" каждого бита блока шифротекста с соответствующим ему битом текущего раундового ключа:

$$block_i = block_i \oplus roundKey_{ji}, где\ j - номер\ раунда, i - порядковый\ номер\ бита$$

Циклический сдвиг shiftTransform

В таблицу, идентичную таблице из этапа tableTransform, построчно записывается блок шифротекста, после чего каждая i-я строка циклически сдвигается влево на s байт, где $s = tableSize - i$ (рисунок 5):

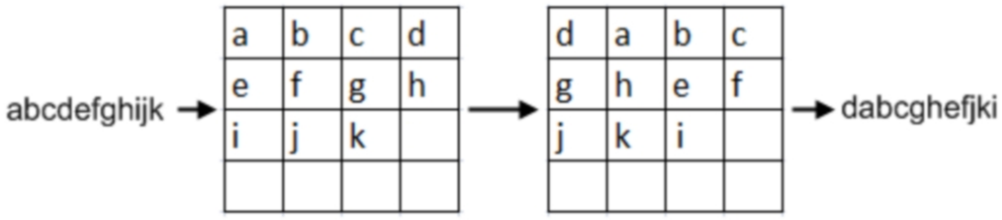

Рисунок 5. Пример циклического сдвига по таблице размером 4 на 4

Перемешивание строки mixString

Аналогично предыдущему этапу происходит запись блока в таблицу, после чего к каждому байту строки добавляется сумма всех байтов строки, при этом при четной длине или ширине таблицы последний байт строки полностью игнорируется (рисунок 6):

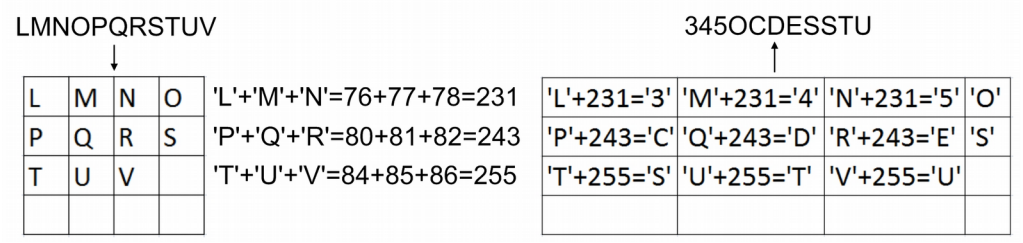

Рисунок 6. Пример перемешивания строки

Оценка алгоритма

В отличие от существующих блочных криптосистем, в которых размер блока и длина ключа фиксированы и задаются заранее [5], в рассматриваемом алгоритме данные параметры зависят от пароля. При изменении хотя бы одного бита в пароле меняется как содержимое ключа, так и его длина.

Например, пароль "aaa" генерирует следующий нулевой ключ длиной 80 байт (в шестнадцатеричном формате):

15 98 81 10 20 40 8b 04 04 42 30 2a 20 10 11 89 04 28 40 80 60 50 01 01 e5 28 28 47 c4 09 0d 85 71 0d 21 01 04 18 20 40 40 20 58 21 e1 54 50 61 10 10 80 80 a4 01 e0 03 28 28 70 85 04 20 01 10 11 22 14 20 e0 20 1a 5a 22 24 25 03 10 e2 46 0d

Отличающийся от него всего одним битом пароль "aac" – семидесятипятибайтный ключ:

04 48 50 d1 b9 a0 b8 71 8a 0f 01 0e 0e 02 24 42 80 0c 89 08 08 02 28 46 04 a8 d5 08 60 3e 80 e1 48 b0 39 28 a0 01 02 81 45 e3 85 3c 48 4b a4 62 18 14 14 11 24 24 79 20 e1 40 22 4c 28 44 5c 08 17 60 6a 0c 20 c0 40 02 44 08 16

Зависимость длины блока шифротекста от пароля позволяет значительно повысить стойкость криптосистемы, поскольку размер блока является существенным параметром для всех этапов алгоритма.

Также данный алгоритм обладает лавинным эффектом, то есть изменение одного бита в шифруемом тексте влияет на весь блок. Например, шифрование строки "00000000000000000000000000000000" паролем "a" в течение двенадцати раундов приводит к следующей зашифрованной строке в шестнадцатеричном формате:

d6 07 45 4e 36 7f 83 63 71 4c e6 ef 25 bd 02 e8 79 1b 00 82 c8 4e 3b 91 91 3c 21 61 85 68 31 39

Шифрование строки "10000000000000000000000000000000" (отличается от предыдущей одним битом) тем же паролем приводит к такому результату:

47 02 e7 d0 ac 88 ae a1 5d 5b 56 72 67 ae b3 31 fe b8 9a e7 ea cc a0 f9 f8 90 f1 3a 9c b8 57 cc

Алгоритм показывает хорошие статистические свойства и может быть использован в практических целях.

Заключение

В данной работе представлен алгоритм симметричной блочной криптосистемы с использованием клеточных автоматов для шифрования ключа. Показаны преимущества метода перед классическими криптосистемами, в частности, динамический размер блока, а также возможность практического применения в задачах шифрования.

Список литературы

1. Шеннон К. Работы по теории информации и кибернетике, М., Издательство иностранной литературы, 1963, 832 с.

2. Евсютин О.О., Шелупанов А.А. Приложения клеточных автоматов в области информационной безопасности и обработки данных //Доклады ТУСУРа, 2012, № 1(25), часть 2, с. 119 - 125.

3. LifeWiki - the wiki for Conway's Game of Life [Электронный ресурс] //URL: http://www.conwaylife.com/wiki/Cellular_automaton#Rules (дата обращения: 07.02.2015).

4. Martínez, Genaro J.; Seck-Tuoh-Mora, Juan C.; Zenil, Hector (2013), "Computation and Universality: Class IV versus Class III Cellular Automata", Journal of Cellular Automata 7 (5–6): 393–430

5. Шнайер Б. Прикладная криптография, М., Триумф, 2002, 816 с.

References:

1. Shannon C. (1963). Raboti po teorii informtsii i kibernetike (Works about information theory and cybernetics), Moscow, "Izdatelstvo inostrannoy literatury", 832 p.

2. Evsutin O.O., Shelupanov A.A. (2012). Prilozheniya kletochnih avtomatov v oblasti informatsionnoi bezopasnosti i obrabotki dannih (Applications of cellular automata in the field of information security and data processing) //Doklady TUSURa, № 1(25), part2, pp. 119 - 125.

3. LifeWiki - the wiki for Conway's Game of Life. Available at: http://www.conwaylife.com/wiki/Cellular_automaton#Rules (accessed: 07.02.2015).

4. Martínez, Genaro J.; Seck-Tuoh-Mora, Juan C.; Zenil, Hector (2013), "Computation and Universality: Class IV versus Class III Cellular Automata", Journal of Cellular Automata 7 (5–6): 393–430.

5. Schneier B. (2002). Prikladnaya kriptographiya (Applied Cryptography), Moscow, Triumf, 816 p.