***Шифр Andromeda***

Шифр Андромеда – это поточный шифр с высокой степенью стойкости практически к любым видам атак, за счет колоссального выхода генератора ключа ((2562)256^1500+256^1500). Наиболее эффективная атака на этот шифр (как и на любой другой) это перебор ключа методом грубой силы, но за счет некоторых особенностей, о которых будет сказано ниже, эффективность брутфорса для этого шифра крайне низка. Без пароля подобрать ключ за вменяемое время нельзя. Алгоритм также «прощает» использование простых паролей, выход генератора за счет широкополосного блока прогрева никак не дает понять, что использована простая входная последовательность бит. При подборе пароль более реально просто угадать, если пароль «123» например, чем перебрать все возможные значения ключа. Шифр построен по схеме сходной с одноразовым блокнотом в основу положена архитектура sponge, состоит шифр из независимых блоков по принципу криптопримитивов, что дает возможность, комбинируя блоки шифра, собирать сколь угодно сложные схемы для шифрования информации. По факту это шифр является гибридным (блочно-поточным), но авторами он позиционируется как поточный. Andromeda легко реализует любые режимы шифрования от CBC до OFB в любых вариациях. Этот шифр прост в реализации, как в программном так и аппаратном обеспечении. Впервые этот шифр был представлен в 2012 году на сайте компании oriondevteam.com, которая является авторам нашумевшего в то время программного обеспечения orion cipher box с алгоритмом комплексной криптографической защиты orion 7b. Есть информация, что шифр был выставлен на конкурс NIST, после чего им заинтересовались спецслужбы США, в следствии этого шифр был снят с конкурса, также компания-разработчик данного алгоритма куда-то внезапно пропала, после описанных выше событий.

В базовой реализации обычно состоит из пяти блоков:

− блок инициализации;

− блок шифрования;

− блок прогрева;

− блок перестановки;

− блок деинициализации.

Центральным узлом любого поточного шифра является генератор ключевого расписания, и Andromeda не исключение.

По своей схеме работы генератор ключа Andromeda напоминает генератор ключа в RC4, за исключением того, что Andromeda использует два блока ключа, каждый по 12 000 бит, хотя на сайте авторов была описана реализация генератора, в котором эти блоки имеют размер аж по 120 000 бит каждый. Еще одно отличие генератора ключевого расписания от каноничного генератора RC4 в том, что блоки заполняются при помощи алгоритмов хеширования в режиме PBKDFx, x~8000 с использованием функций-преобразователей ([хеш-функции](https://ru.m.wikipedia.org/wiki/%D0%A5%D0%B5%D1%88-%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%B8)).

Хешировать входной пароль для получения ключа возможно в любой комбинации, но автор шифра рекомендует именно режим PBKDFx, x~8000, причем каждый из массивов генератора, которые называются Pool и S-Box рекомендуется заполнять так:

Например у нас есть пароль 123, S-Box мы заполняем результирующим выходом от хеш-функции Whirlpool PBKDFx, x~8000 (123) – выход 1500 байт, после чего пароль реверсируется 123 => 321 и массив Pool заполняется так, SHA512 PBKDFx, x~8000 (321) – выход 1500 байт. Как и сказано выше, это рекомендуемые параметры, выполнять преобразование пароля для инициализации структур генератора можно сколь угодно сложным образом, разумеется можно менять используемые алгоритмы. Самое главное правило – функция хеширования должна быть криптографически стойкой, а результат ее работы в любом из режимов должен отвечать правилу для ГПСЧ – правило 1567.

Следующая особенность Andromeda в том, что шифр имеет байтовый весовой шифратор, принцип его работы, а также псевдокод мы рассмотрим ниже.

Теперь рассмотрим каждый блок шифра в отдельности.

**Блок инициализации**

Блок инициализации нужен для того, чтобы выполнить описанные выше действия над массивами ГПСЧ алгоритма, его работа отражена выше.

**Блок шифрования**

Одной из особенностей блока шифрования алгоритма является:

1. Использование трех ключевых байт для изменения одного открытого байта информации

2. Механизм шифрования, который в зависимости от значения двух ключей (один из Pool другой из SBox) выполняет либо один набор действий для шифрования открытых данных, либо другой.

Блок криптографического преобразования можно также изменять и реализовывать как угодно. Есть два варианта от авторов, в первом варианте криптографическое преобразование достигается за счет умножения открытого текста на ключ по-модулю, алгоритм Radix-Z

Байтовый ширатор\дешифратор (псевдокод).

byte DoEncode(X, k1, k2, k3:byte)

{

if (k1>k2) // плотность 3 байта на один шифруемый байт (3->1)

// в зависимости от веса ключевых байт шифруем их по разному

return = ((X xor k3) + k2) xor k1

else

return = ((X + k1) + k2) xor k3

}

byte DoDecode(X, k1, k2, k3: byte)

{

if (k1>k2) // выполняем обратные действия при расшифровке

return= ((X xor k1) - k2) xor k3

else

return= ((X xor k3)-k2)-k1;

}

**Блок прогрева**

Этот блок представляет собой работу генератора в холостом режиме, в режиме счетчика, т.е. выполняется процедура шифрования нулевого байта заданным паролем, в результате чего начальное состояние массивов генератора изменяется, после чего можно приступать к шифрованию основных данных. Это нужно для того, чтобы исключить даже минимальную корреляцию элементов ключевого расписания. Напомним, что **Корреля́ция** (от [лат.](https://ru.wikipedia.org/wiki/%D0%9B%D0%B0%D1%82%D0%B8%D0%BD%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *correlatio* «соотношение, взаимосвязь») или **корреляционная зависимость** — [статистическая](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B0%D1%82%D0%B8%D1%81%D1%82%D0%B8%D0%BA%D0%B0) взаимосвязь двух или более [случайных величин](https://ru.wikipedia.org/wiki/%D0%A1%D0%BB%D1%83%D1%87%D0%B0%D0%B9%D0%BD%D0%B0%D1%8F_%D0%B2%D0%B5%D0%BB%D0%B8%D1%87%D0%B8%D0%BD%D0%B0) (либо величин, которые можно с некоторой допустимой степенью точности считать таковыми).

**Блок перестановки**

блок, в котором происходит перестановка элементов массивов Pool и SBox для определения ключей.

Псевдокод реализации блока перестановки с шифратором приведен ниже:

byte Andromeda\_cipher(x:byte; ASwitch:TCiphMode – это на вход принимаем параметр запуска, либо шифрования, либо дешифровки)

{

{переменные}

key1,key2:Cardinal;

temp, key3:byte;

try

Si = (Si + 1) mod 1500;

Sj = (Sj + Pool[Si]) mod 1500;

key1 = Pool[Si];

Pool[Si] = Pool[Sj]; // перестановка Pool

Pool[Sj] = key1;

key1 = (Pool[Si] \* Pool[Sj] + SBox[Si] + SBox[Sj]{Pool[Si] + Pool[Sj]}) mod 1500; // Получаем номер ключевого байта в Pool

Si = (Si + 1) mod 1500;

Sj = (Sj + SBox[Si]) mod 1500;

key2 = SBox[Si];

SBox[Si] = SBox[Sj]; // перестановка SBox

SBox[Sj] = key2;

key2 = ({SBox[Si] + SBox[Sj]}Pool[Si] + Pool[Sj]+SBox[Si]\* SBox[Sj]) mod 1500; // получаем номер ключевого байта в SBox

temp = Pool[Si];

Pool[Si] = SBox[Sj]; // Перестановка между Pool и SBox

SBox[Sj] = temp;

key3 = (SBox[Si] xor Pool[Sj])mod 256; // усредненный синтетический ключ

switch (ASwitch) // собственно само шифрование

{

cpmEncr: Return=DoEncode(x,Pool[key1],SBox[key2],key3); break;

cpmDecr: Return:=DoDecode(x,Pool[key1],SBox[key2],key3); break;

}

//

// ВАЖНО! Счетчики Si и Sj не сбрасываем, так как данный код должен вызываться в цикле вызывающего его метода, если счетчик сбросить – шифрования не будет

// Si,Sj – счетчики – целое число без знака размером числа, предусмотренного конкретной // архитектурой процессора, в начале процесса шифрования они равны 0

//

catch(){

DeInitCipher;

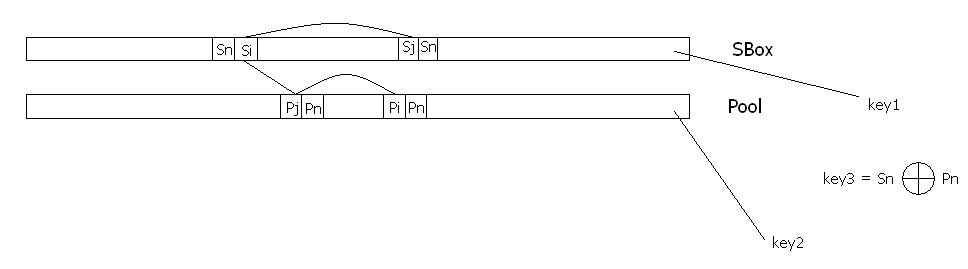
return=0;

throw Exception.Create(ECipher);

}

}

Схема работы блока перестановки:



**Блок деинициализации**

Блок очищение инициализированных генератором структур в памяти компьютера методом Гутмана, что означает удаление многократным перезаписываниемячеек памяти по адресам. Применение алгоритма – гутман или DoD не критично, в аппаратной реализации данный блок работает немного по-другому, он просто зануляет ячейки временной памяти в плате