- ✓ дешифрование.

Более того, данные программы могут использоваться в качестве примера для разработки реальных криптографических приложений.

Література: ***1.*** *Ховард М., Лебланк Д. Защищенный код/Пер. с англ. – М.: Издательско-торговый дом «Русская редакция», 2003. – 704 стр.: ил.* ***2.*** *Щербаков А. Ю., Домашев А. В. Прикладная криптография. Использование и синтез криптографических интерфейсов. – М.: Издательско-торговый дом «Русская редакция», 2003. – 416 с.: ил.* ***3.*** *Platform SDK: Security. Cryptography - Microsoft Platform SDK, February 2001 Edition.* ***4.*** *PGP Software Development Kit, Reference Guide. Version 1.7.1 Int. - Copyright © 1990-1999 Network Associates, Inc. and its Affiliated Companies. All Rights Reserved. – www.pgp.org* ***5.*** *PGP Software Development Kit, User's Guide. Version 1.7.1 Int. - Copyright © 1990-1999 Network Associates, Inc. and its Affiliated Companies. All Rights Reserved. – www.pgp.org* ***6.*** *Хеширование, шифрование и цифровая подпись с использованием CryptoAPI и .NET – www.rsdn.ru*

УДК 681.3.06

СТОЙКИЙ К КОЛЛИЗИЯМ АЛГОРИТМ WHIRLPOOL

Геннадий Халимов, Евгений Котух

Харьковский национальный университет радиоэлектроники

Аннотация: **Рассмотрены требования к хеш функциям устойчивым к коллизиям, алгоритм Whirlpool и аспекты безопасности криптопримитива .**

Suumary: **Requirements for collision-resistant hash functions, Whirlpool algorithm and aspects of crypto primitive security are considered.**

Ключевые слова: **алгоритм Whirlpool, бесключевая хеш-функция.**

I Введение

Разнообразие подходов к решению проблем аутентификации обусловлено требованиями к системе защиты информации. Среди основных подходов к построению MAC кодов, широкое распространение получили коды на основе бесключевых хэш-функций (MDC-коды), с применением универсальных хэш-функций, а также MAC коды, использующие блочные шифры.

Применение бесключевых хеш-функций является эффективным методом для построения MAC кодов, так как обеспечивает высокую стойкость к коллизиям и скорость вычислений. В проекте Nessie такие MAC коды представлены как HMAC коды и их конструкция описана в RFC 2104 [1]:

Безопасность HMAC алгоритма определяется секретностью бесключевой хеш-функции. В проекте NESSIE лучшим в категории «Устойчивая к коллизиям хэш-функция» был признан алгоритм Whirlpool. Алгоритм Whirlpool вычисляет 512-битный хеш-код с использованием в качестве функции сжатия блочного шифра, котрый является модификацией алгоритма Rijndael.

С этой целью в разделе II рассмотрены требования к хэш-функциям, устойчивым к коллизиям. В разделе 2 приводится описание алгоритма Whirlpool, оценки и сравнение хэш-функций. В разделе 3 изучены аспекты безопасности криптопримитива Whirlpool.

II Требования безопасности к хэш-функциям, устойчивым к коллизиям

Конструктивными элементами HMAC кодов являются хеш-функции, функции сжатия и итерационные хеш-функции. Определения отмеченных хеш-функций и их криптографических свойств приведем в изложении Блэка и Рогавэя [2].

Определение 1. Хеш-функцией называется функция отображения $h: D \rightarrow R$, где область значений $D = \{0,1\}^*$, а $R = \{0,1\}^n$ для некоторого $n \geq 1$.

Определение 2. Функцией сжатия называется функция отображения $f: D \rightarrow R$, где $D = \{0, 1\}^a \times \{0,1\}^b$ и $R = \{0,1\}^n$ для некоторых a, b, $n \geq 1$ и $a + b \geq n$.

Определение 3. Итерационной хеш-функцией от функции сжатия $f: (\{0,1\}^n \times \{0, 1\}^b) \rightarrow \{0,1\}^n$ является хеш-функция $h: (\{0, 1\}^b)^* \rightarrow \{0,1\}^n$ определенная $h(X_1 \ldots X_t) = H_t$, где $H_i = f(H_{i-1}, X_i)$ при $1 \leq i \leq t$ ($H_0 = IV$).

Определяющими требованиями к хеш-функциям являются их стойкость к вычислению прообраза, второго прообраза, а также стойкости к коллизиям.

Определение 4 (Стойкость к вычислению прообраза) [3]. Хеш-функция $h: \{0,1\}^* \rightarrow R$ является

стойкой к вычислению прообраза силой (t, ϵ), если не существует вероятностного алгоритма I_h, с входными значениями Y Є $_R$ R и значениями на выходе X Є {0,1}*, временем выполнения не более чем t, где h (X) = Y и вероятностью не менее ϵ, оцененной при случайном выборе Y и I_h.

Стойкость хеш-функций к вычислению прообраза имеет важное значение для систем аутентификации, использующих хэш-значения паролей и секретных ключей.

Определение 5 (Стойкость к вычислению второго прообраза) [3]. Пусть S – конечное подмножество {0,1}*. Хеш-функция h: {0,1}*→ R является стойкой к вычислению второго прообраза силой (t, ϵ, S), если не существует вероятностного алгоритма S_h, с X Є$_R$ S и X' Є {0,1}*, временем выполнения не более чем t, где X' ≠ X и h (X') = h (X) и вероятностью не менее ϵ, оцененной при случайном выборе X и S_h.

Стойкость хеш-функций к вычислению второго прообраза определяет безопасность систем аутентификации с цифровой подписью.

Определение 6 (Стойкость к коллизиям) [3]. Хеш-функция h: {0,1}* → R является стойкой к коллизиям силой (t, ϵ), если не существует вероятностного алгоритма C_h с известными выходными значениями X, X' Є {0,1}*, временем выполнения не более чем t, где X' ≠ X и h (X) = h (X') и вероятностью не менее ϵ, оцененной при случайном выборе Ch.

Определение 7. Хеш функция называется простой или слабой если является стойкой к вычислению прообраза и стойкой к вычислению второго прообраза.

Определение 8. Хеш функция называется сильной если является стойкой к вычислению прообраза, стойкой к вычислению второго прообраза и стойкой к коллизиям.

Определение сильной хэш-функции показывает, что вычислительно невозможно найти какую-либо коллизию и защищает против класса атак, известных как атака «день рождения».

Основными атаками на функции выработки MAC кодов являются следующие:

1. Нахождение корректной пары прообраза и MAC $(x, h(x,k))$ по одной или более заданным корректным парам $(x_i, h(x_i,k))$ для любого $x \neq x_i$ при неизвестном секретном ключе k;

2. Нахождение неизвестного сеансового ключа k по одной или более заданным корректным парам прообразов и кодов аутентификации $(x_i, h(x_i,k))$.

Атаки на функции MAC могут выполняться при следующих условиях:

Атака с известным текстом – злоумышленнику заданы только одна или несколько корректных пар прообразов и кодов аутентификации $(x_i, h(x_i,k))$;

Атака с выбираемым текстом – злоумышленник имеет возможность получить корректные пары $(x_i, h(x_i,k))$ для выбранных значений x_i (атака на нахождение ключа);

Атака с адаптивным выбором текста – злоумышленник может получить корректные пары $(x_i, h(x_i,k))$ для любых x_i, выбранных в зависимости от результатов предшествующих запросов (атака с целью нахождения ключа).

Все атаки на хэш-функции можно разделить на две группы: атаки, базирующиеся на уязвимости алгоритма преобразований (аналитические) и атаки, независящие от алгоритма.

К атакам независящим от алгоритма относятся следующие.

1. *Атака "грубой силой"* [4] может быть выполнена для нахождения прообраза по заданному хэш-значению или для нахождения прообраза, дающего заданное хэш-значение. Суть атаки заключается в последовательном или случайном переборе входных сообщений и сравнения результата выполнения хэш-функции с заданным. Сложность такой атаки оценивается 2^{l-1} операций вычисления хэш-значений, где l - длина хэш-значения в битах.

2. *Атака методом "дня рождения"* [4] выполняется для нахождения двух различных сообщений с одинаковыми хэш-значениями. Эта атака основана на парадоксе "дня рождения" и заключается в том, что в двух сгенерированных множествах хэш-значений, содержащих n_1 и n_2 элементов соответственно, вероятность нахождения совпадающих элементов между этими множествами оценивается выражением: $P \approx 1 - e^{-\frac{n_1 n_2}{2^l}}$.

При $n_1 = n_2 = 2^{\frac{l}{2}}$ сложность атаки оценивается как $2^{\frac{l}{2}+1}$ операций вычисления хэш-значений, а

вероятность успеха равна $P \approx 1 - \frac{1}{e} \approx 0{,}63$.

3. *Атака полного перебора ключей* осуществляется для нахождения неизвестного секретного сеансового ключа функции выработки MAC. Для нахождения ключа атакующий, имеющий не менее одной пары (сообщение, MAC), последовательно перебирает ключи. Так как пространство сообщений неоднозначно отображается в пространство хэш-значений, то может быть обнаружено множество подходящих значений ключей. Чтобы точно найти правильный ключ, необходимо выполнить проверку найденных ключей на большом множестве различных пар (сообщение, MAC). Как показано в [4] максимальное число попыток точного определения ключа составляет $m + \frac{2^k - 1}{1 - 2^{-l}}$, где k - длина секретного ключа в битах, m - количество различных пар (сообщение, MAC). В среднем достаточно $\frac{k}{l}$ различных пар (сообщение, MAC) чтобы точно определить секретный ключ.

4. *Атака "встреча посередине"* [5] является модификацией атаки методом "дня рождения" и используется для хэш-функций с циклической структурой, если цикловая функция $f()$ инвертируема по отношению к промежуточному значению X или блоку сообщения M_i. Эта атака по сложности сопоставима с атакой методом "дня рождения".

5. *Атака с коррекцией блока* используется в случае, если атакующий обладает сообщением и хочет изменить в нем один или более блоков без изменения хеш-значения. Один цикл MD5 уязвим к этой атаке: атакующий берет блок сообщения M_i (16 слов по 32 бита), оставляет 11 слов, модифицирует одно слово, и вычисляет оставшиеся 4. В результате получается блок M_i', отображающийся в то же самое хеш-значение, что и M_i. Полная версия MD5 не уязвима к этой атаке.

6. *Атака с фиксированной точкой* [4] может применяться при условии, что цикловая функция f имеет одну или несколько фиксированных точек. Фиксированной точкой называется блок сообщения M_i, для которого выполняется $f(X_i, M_i) = X_i$, т.е. существует блок сообщения M_i, не изменяющий промежуточный результат X_i. Таким образом, в сообщение M можно добавлять или удалять блоки M_i без изменения хэш-значения. Защитой от таких атак служит вычисление длины сообщения и добавления ее в конце сообщения.

7. *Атака на базовый алгоритм шифрования* [6] используется для атаки на хэш-функции, базирующиеся на блочных симметричных шифрах. Так как алгоритмы шифрования разрабатывались как двунаправленные (поддерживают обратное преобразование), то это может увеличить уязвимость в функцию сжатия с их применением.

III Описание алгоритма Whirlpool

Whirlpool – является хэш-функцией, устойчивой к коллизиям, оперирует с блоками по 512 бит и генерирует 512-битную хеш-последовательность. Whirlpool состоит из итеративной функции сжатия, основанной на 512-и битном блочном шифре, который выполняется за 10 циклов.

Whirlpool построен по схеме хеширования Миагучи-Принеля:

$$\eta_i = \mu(m_i), \quad 1 \le i \le t, \tag{1}$$

$$H_0 = \mu(IV), \tag{2}$$

$$H_i = W[H_{i-1}](\eta_i) \oplus H_{i-1} \oplus \eta_i, \quad 1 \le i \le t, \tag{3}$$

где IV - вектор инициализации, m_i – блок данных, W - блочный шифр, μ - функция сжатия..

Внутренний блочный шифр W использует ключевую матрицу K для шифрования матрицы состояния. Блочный шифр оперирует с 512-битными блоками и имеет 512-битный ключ. Вычисляется 10 циклов, на каждом из которых вызывается функция сжатия μ, а на 11-ом цикле происходит аффинное добавление ключа K^0.

$$W_K = \left(\bigcirc_{1}^{r=10} \rho[K^r] \right) o \sigma[K^0], \quad (4)$$

где K^0 … K^r – ключи, полученные после развертывания ключа, R – число циклов, по умолчанию равно 10.

Вход и выход функции.

Состояние хэш-функции описывается матрицей M_{8x8} [GF(2^8)]. Для формирования матрицы из 512-и битного блока данных используется функция μ.

$$\text{Если } \mu(a)=b, \text{ то } b_{ij}=a_{8i+j},\ 0 \le i, j \le 7, \quad (5)$$

где a_k – k-тый байт блока данных,

b_{ij} – элемент матрицы M.

Функция μ. состоит в замене каждого байта в матрице состояния M_{8x8} [GF(2^8)] на байт, полученный из блока подстановки S.

$$\nu(a)=b \Leftrightarrow b_{ij}=S(a_{ij}),\ 0 \le i, j \le 7, \quad (6)$$

где a_{ij}, b_{ij} – элементы матрицы M.

Таблица подстановки S построена из более мелкой 4-х битной таблицы подстановки и определена в последней спецификации алгоритма.

Циклическая перестановка π. Перестановка π независимо циклически сдвигает каждый столбец матрицы M так, что в итоге j-тый столбец сдвигается вниз на j позиций.

$$\pi(a)=b \Leftrightarrow b_{ij} = a_{(i-j) \bmod 8, j},\ 0 \le i, j \le 7. \quad (7)$$

Цель такого преобразования рассеять байты каждого ряда по всем рядам.

Линейное рассеивание θ. Линейное рассеивание θ - это функция линейного отображения M_{8x8} [GF(2^8)]→ M_{8x8} [GF(2^8)], основанная на коде MDS с матрицей генерации G_C=[IC], где

$$C = \begin{bmatrix} 01_x & 01_x & 03_x & 01_x & 05_x & 08_x & 09_x & 05_x \\ 05_x & 01_x & 01_x & 03_x & 01_x & 05_x & 08_x & 09_x \\ 09_x & 05_x & 01_x & 01_x & 03_x & 01_x & 05_x & 08_x \\ 08_x & 09_x & 05_x & 01_x & 01_x & 03_x & 01_x & 05_x \\ 05_x & 08_x & 09_x & 05_x & 01_x & 01_x & 03_x & 01_x \\ 01_x & 05_x & 08_x & 09_x & 05_x & 01_x & 01_x & 03_x \\ 03_x & 01_x & 05_x & 08_x & 09_x & 05_x & 01_x & 01_x \\ 01_x & 03_x & 01_x & 05_x & 08_x & 09_x & 05_x & 01_x \end{bmatrix}, \quad (8)$$

т.о. $\theta(a)=b \Leftrightarrow b=a \bullet C$. (9)

Цель рассеивания – перемешать все байты каждого рядка матрицы.

Добавление ключа σ[k]. Аффинное добавление ключа состоит в побитовом сложении матрицы M с ключевой матрицей K.

$$\sigma[k](a)=b \Leftrightarrow b_{ij}=a_{ij} \oplus k_{ij},\ 0 \le i, j \le 7. \quad (10)$$

Цикловые константы c^r. Цикловая константа для цикла номер r будет матрица $c^r \in M_{8x8}$ [GF(2^8)] определенная как:

$$C_{0j}^r \equiv S[8(r-1)+j],\ 0 \le j \le 7, \quad (11)$$

$$C_{ij}^r \equiv 0,\ 1 \le i \le 7,\ 0 \le j \le 7. \quad (12)$$

Цикловая функция ρ[k]. На r-том цикле алгоритма цикловая функция выполняет последовательное отображение M_{8x8} [GF(2^8)]→ M_{8x8} [GF(2^8)] и использует ключевую матрицу K.

$$\rho[K] \equiv \sigma[K] o \theta o \pi o \gamma, \quad (13)$$

где ∘ – операция последовательного выполнения функций.

Развертывание ключа. Это процедура получения из 512-и битного ключа $K \in M_{8x8}$ [GF(2^8)] последовательности из десяти 512-и битных цикловых ключей K^0, K^1, … , K^{10}.

$$K^0 = K, \quad (14)$$

$$K^r = \rho[C^r](K^{r-1}),\ 1 \le r \le 10. \quad (15)$$

Дополнение и усиление. Перед вычислением хэша сообщения M длинны L ($L<2^{256}$) его дополняют одним единичным битом и нулевыми битами для достижения длинны L кратной 256. Затем сообщение

разбивают на t блоков m_1, m_2, … , m_t.

Вычисление хэш-значения. Хэш-значение определено как выход H_t функции сжатия, преобразованный из матрицы обратно в последовательность байт.

$$\text{Whirlpool}(M) \equiv \mu^{-1}(H_t). \quad (16)$$

В таблице 1 представлены характеристики, свойства и аналитические результаты оценки быстродействия алгоритма Whirlpool в сравнении с другими известными алгоритмами.

Таблица 1. Сравнительный анализ MAC алгоритмов

Алгоритм	Длина MAC кода(бит)	Длина ключа (бит)	Базовые преобразования	Celeron 600 MHz (Мбит/с)	Pentium III 1000 MHz (Мбит/с)
HMAC-Whirlpool	**512**	**512**	в конечных полях и матрицах	**28,013**	**46,961**
UMAC	64	128	в кольцах	989,37	1648,953
HMAC-MD4	128	512	лог. и арифм.	344,086	467,793
HMAC-MD5	128	512	лог. и арифм.	278,715	574,635
HMAC-RIPEMD	160	512	лог. и арифм.	147,465	246,568
HMAC-SHA-1	160	512	лог. и арифм.	206,285	344,433
HMAC-SHA-2	256 512	512 512	лог. и арифм.	81,308 41,159	135,557 68,701
CBCMAC-Rijndael	128	128	лог. и арифм.	139,376	231,255

Из проведенного анализа следует, что Whirlpool уступает многим алгоритмам по быстродействию при интерпретации данных алгоритмов на языке C, но одним из главных преимуществ алгоритма Whirlpool остается размер MAC кода, равный 512 битам.

IV Анализ безопасности Whirlpool алгоритма

Доказательство безопасности алгоритма Whirlpool основывается на недоказанности уязвимости схемы Миагучи-Принелла для построения хэш-функции из блочного шифра. Эта схема доказуемо стойкая, если полагать, что используемый алгоритм шифрования идеален. Блочный шифр W, используемый в Whirlpool, очень похож на алгоритм AES Rijndael.

В таблице 2 представлены сравнительные характеристики для блочных шифров Rijendael и W [7]. Основное отличие состоит в том, что Rijndael может работать с блоками длиной 128, 192, 256 бит, а W только 512 бит. Из-за сходства алгоритмов часть анализа безопасности алгоритма Rijndael можно отнести и алгоритму Whirlpool.

Таблица 2. Сравнительный анализ блочных шифров

Свойства	Rijndael	W
Размер блока (в битах)	128, 160, 192, 224, 256	только 512
Число циклов	10, 11, 12, 13, 14	только 10
Развертывание ключа	априорный алгоритм	непосредственно циклическая функция
GF(2^8) полином	$x^8 + x^4 + x^3 + x + 1$ (0x11B)	$x^8 + x^4 + x^3 + x^2 + 1$ (0x11D)
Базис для S-box	отображение $u \rightarrow u^{-1}$ аффинное преобразование в поле GF(2^8)	рекурсивная структура
Базис для циклов	полином x^i над GF(2^8)	последующее содержимое S-box
Рассеивание	умножение циркулянтной 4x4 матрицы MDS cir(2, 3, 1, 1)	умножение циркулянтной 8x8 матрицы MDS cir(1, 1, 4, 1, 8, 5, 2, 9)

Криптоанализ алгоритма Whirlpool, представленный в [7], рассматривает условия выбора таблицы подстановок S и наличие квадратичных зависимостей между входными и выходными блоками таблицы, а также влияние числа циклов встроенного блочного шифра на безопасность алгоритма.

Выбор S-box. Алгоритм Whirlpool использует блок подстановки для отображения 8 входных бит в 8

выходных бит. В оригинальной версии алгоритма использовалась случайно сгенерированная таблица подстановки, выбранная для невозможности успешного проведения линейного и дифференциального криптоанализа. Однако была допущена существенная ошибка, которая приводит к появлению линейного параметра. В представленной к рассмотрению в проекте NESSIE версии алгоритма использовалась другая таблица подстановки, состоящая из 4-х битных таблиц, которая устраняла этот недостаток.

Уменьшение числа циклов. Версия алгоритма с уменьшенным количеством циклов в функции сжатия обладает некоторыми не случайными свойствами. Исследования влияния числа циклов на стойкость функции сжатия представлен в [7]. Пусть есть 256 текстов длинной 64 байта, которые отличаются в одном байте, а в остальных равны. Тогда следует, что после 2-х циклов шифрования тексты примут все 256 значений в каждом из 64-х байт, и после 3-х циклов шифрования сумма всех 256 байт на каждой позиции всех текстов будет равна 0. Такая структура имеет название интеграл. Отметим, что остальные 63 структуры сходные до тех пор, пока позиция изменяемого байта может быть любой из 64 значений.

3-х цикловый интеграл, описанный выше, может быть преобразован в 4-х цикловый, предполагая структуру из 2^{64} текстов. Основное отличие состоит в том, что после первого цикла, она принимает все 2^{64} значений в верхнем ряду, а оставшиеся 3 цикла могут быть представлены как набор 2^{56} вариантов 3-х цикловых интегралов. Поскольку тексты в каждой интегральной сумме дают 0 в любом байте после 4-х циклов, то это же можно сказать и про сумму всех 2^{64} текстов.

Аналогично можно определить 3-х цикловый обратный интеграл, как 3 цикла инверсного шифра W. В версии алгоритма W с 6-ю циклами можно комбинировать первые 3 цикла 4-х циклового прямого интеграла и 3-х циклового обратного интеграла для перекрытия всех циклов шифра с вероятностью 1.

Таким образом, с временной сложностью 2^{120} можно найти набор 2^{120} входных значений W с уменьшенным количеством циклов до 6, поскольку каждый байт входных и выходных значений принимает все значения много раз. Предполагается, что для того, что бы найти структуру из 2^{120} текстов со свойствами схожими с W с уменьшенным количеством циклов до 6, потребуется генерация 2^{128} выходных значений и значительное количество памяти.

В версии W с уменьшенным количеством циклов до 7 можно комбинировать полный 4-х цикловый прямой интеграл и 3-х цикловый обратный интеграл для покрытия семи циклов с вероятностью 1. На данный момент нет результатов как такой интеграл с числом циклов 7 можно применить для эффективного различения W от случайной 512 битной перестановки.

Квадратичные зависимости. Исследовалось существование квадратичной зависимости во входных и выходных значениях блока подстановки алгоритма Whirlpool. Было показано, что для блоков подстановки Rijndael и Serpent существуют квадратичные уравнения для входных и выходных бит с вероятностью 1. Такие уравнения всегда существуют для n бит n-битного блока подстановки, если $n \leq 6$, но для $n > 6$ не всегда.

Были проведены исследования по выявлению квадратичных зависимостей в блоке перестановки Whirlpool. Блок перестановки это 8 битная перестановка. Существует максимум 137 возможных степеней свободы в многомерном выражении для 8 входных и 8 выходных бит. Простейший метод проверки наличия таких зависимостей заключается в вычислении 256 раз определителя 137 мерных двоичных матриц. Для полной таблицы подстановки Whirlpool не было найдено квадратичных зависимостей. Тем не менее, так как блок подстановки состоит из нескольких 4-х битовых подстановок, задача построения маленькой системы многомерных квадратичных уравнений существенно упрощается.

V Выводы

Алгоритм Whirlpool имеет очень высокую стойкость, сравнимую с SHA-2 (512), но невысокое быстродействие и, соответственно, его можно рекомендовать к применению в системах, где необходимо обеспечить стойкость в течение длительного периода времени, а критерий стойкости является определяющим и намного важнее скорости. По мнению разработчиков, алгоритм Whirlpool является более масштабируемым, чем даже самые современные хэш-функции. Математическая простота алгоритма, достигнутая в процессе разработки, позволяет упростить и процесс анализа стойкости. Длина MAC кода в 512 бит обеспечивает эффективную защиту от атак, основанных на парадоксе «день рождения», а также улучшает показатели устойчивости к коллизиям. Оптимальная длина ключевых данных, отвечающая современным требованиям, позволила данному алгоритму стать победителем в категории «Алгоритм, устойчивый к коллизиям» в рамках проекта NESSIE.

Литература: ***1.*** *R. Anderson. The classification of hash functions. Proc. of the IMA Conference on Cryptography and Coding, Cirencester, December 1993, Oxford University Press, 1995.* ***2****. J. Black and P.*

Rogaway, Ciphers with arbitrary finite domains." in Proceedings of CT-RSA'02 (B. Preneel, ed.), no. 2271 in Lecture Notes in Computer Science, pp. 114-130, Springer-Verlag, 2002. www.cs.ucdavis.edu/~rogaway/papers/subset.htm ***3****. D. R. L. Brown, Generic groups, collision resistance, and ECDSA. Available at http://eprint.iacr.org/2002/026/2002.* ***4****. В.Н. Вервейко, А.И. Пушкарев, Т.В. Цепурит. Функции хэширования: классификация, характеристика и сравнительный анализ.* ***5.*** *K. Ohta and K. Koyama. Meet-in-the-Middle Attack on Digital Signature Schemes. In Abstract of AUSCRYPT '90, pages 110-121, 1990.* ***6.*** *B. Preneel. Analysis and Design of Cryptographic Hash Functions. PhD thesis, Katholieke University Leuven, January 1993. 7. Paulo S.L.M. Barreto1 and Vincent Rijmen. The Whirlpool Hashing Function.*