УДК 681.3.06

А.А. Кузнецов[1], О.Г. Король[2], В.В. Босько[3]

[1]*Харьковский университет Воздушных Сил им. И. Кожедуба, Харьков*
[2]*Харьковский национальный экономический университет, Харьков*
[3]*Кировоградский национальный технический университет, Кировоград*

МОДЕЛЬ ФОРМИРОВАНИЯ КОДОВ АУТЕНТИФИКАЦИИ СООБЩЕНИЙ С ИСПОЛЬЗОВАНИЕМ УНИВЕРСАЛЬНЫХ ХЕШИРУЮЩИХ ФУНКЦИЙ

Исследуются методы и модели формирования кодов аутентификации сообщений (MAC – Message Authentication Code), в том числе с использованием универсальных хеширующих функций (UMAC – Message Authentication Code using Universal Hashing). Посредством масштабирования применяемых преобразований разрабатывается уменьшенная модель UMAC (мини-UMAC) для исследования коллизионных свойств формируемых кодов аутентификации сообщений.

Ключевые слова: *MAC-коды, универсальные хеш-функции, модель UMAC, коллизионные свойства.*

Введение

Постановка проблемы в общем виде и анализ литературы. Для обеспечения целостности и аутентичности данных в современных информационных системах применяют специальные механизмы, в том числе коды аутентификации сообщений (MAC-коды) [1 – 7]. Перспективным направлением исследований является обоснование структуры и используемых блоков преобразований при формировании кодов аутентификации сообщений, исследование их коллизионных свойств и выработка практических рекомендаций по построению эффективных механизмов обеспечения аутентичности и целостности данных.

Проведенный анализ [8 – 13] показал, что одним из перспективных направлений по исследованию механизмов обеспечения безопасности информации является разработка уменьшенных моделей соответствующих блоков преобразований, исследование их свойств и обоснование предложений по построению полных версий соответствующих механизмов защиты информации. Так, в работах [11 – 13] показано, что применение уменьшенных моделей преобразований позволяет, сохранив алгебраическую структуру криптоалгоритмов, проводить исследования основных показателей их эффективности.

В настоящей работе предлагается дальнейшее развитие данного направления, состоящее в использовании уменьшенных моделей отдельных слоев преобразований для оценки коллизионных свойств кодов аутентификации сообщений UMAC.

Основной материал

1. Коды аутентификации сообщений UMAC. Одна из первых версий алгоритма формирования кодов подлинности сообщений с использованием универсального хеширования UMAC была представлена в работе [1]. В дальнейшем, после некоторой доработки [2 – 4], алгоритм UMAC был представлен в финальном отчете европейского конкурса NESSIE - New European Schemes for Signatures, Integrity, and Encryption (новые европейские схемы для подписей, целостности, и шифрования) [5]. Одна из последних электронных версий алгоритма UMAC доступна в [6]. Наиболее подробно отдельные компоненты UMAC изложены в диссертационной работе [7].

Рассмотрим общую схему алгоритма UMAC, проанализируем основные аналитические соотношения, описывающие внутреннюю структуру и применяемые преобразования при формировании кодов подлинности сообщений.

1.1. Общая схема формирования кодов подлинности сообщений с использованием алгоритма UMAC. Код подлинности сообщений (обозначим его Tag) по спецификации алгоритма UMAC формируется посредством вычисления следующей функции:

$$\mathrm{Tag} = \mathrm{UMAC}\left(\mathrm{K}, \mathrm{M}, \mathrm{Nonce}, \mathrm{Taglen}\right) = \mathrm{Y} \oplus \mathrm{Pad},$$

где K – секретный ключ, длина которого Keylen равна стандартной длине секретного ключа используемого блочного симметричного шифра (спецификацией UMAC рекомендуется использовать алгоритм шифрования AES (FIPS-197), в этом случае длина секретного ключа Keylen принадлежит множеству допустимых значений $\{16, 24, 32\}$ байт); M – информационное сообщение, подлежащее аутентификации, представленное в виде массива-строки, размерностью от одного до 2^{67} бит (2^{64} байт); Nonce – неповторяющееся (для всех вводимых информационных сообщений M) восьмибайтное число; Taglen – целое число из множества до-

пустимых значений $\{4, 8, 12, 16\}$, задающее длину кода подлинности сообщений Tag в байтах; $\mathrm{Hash}(\mathrm{K}, \mathrm{M}, \mathrm{Taglen})$ – функция ключевого универсального хеширования информационного сообщения M с использованием секретного ключа K; $\mathrm{PDF}(\mathrm{K}, \mathrm{Nonce}, \mathrm{Taglen})$ – функция формирования псевдослучайной подложки (Pad) по введенному значению Nonce и секретному ключу K; «$\oplus$» – побитовое сложение (XOR) результата ключевого хеширования сообщения $\mathrm{Y} = \mathrm{Hash}(\mathrm{K}, \mathrm{M}, \mathrm{Taglen})$ и сформированной подложки

$$\mathrm{Pad} = \mathrm{PDF}(\mathrm{K}, \mathrm{Nonce}, \mathrm{Taglen}), \text{ т.е.}$$

$$\mathrm{Tag} = \mathrm{Hash}(\mathrm{K}, \mathrm{M}, \mathrm{Taglen}) \oplus \mathrm{PDF}(\mathrm{K}, \mathrm{Nonce}, \mathrm{Taglen}).$$

Длина хеш-кода Y, подложки Pad и кода Tag принадлежат множеству допустимых значений $\{32, 64, 96, 128\}$ бит. Эти фиксированные значения Taglen соответствуют случаю формированию кодов подлинности сообщений UMAC – 32, UMAC – 64, UMAC – 96 или UMAC – 128, соответственно.

Рассмотрим схему формирования хеш-кодов $\mathrm{Y} = \mathrm{Hash}(\mathrm{K}, \mathrm{M}, \mathrm{Taglen})$ и подложки

$$\mathrm{Pad} = \mathrm{PDF}(\mathrm{K}, \mathrm{Nonce}).$$

1.2. Схема формирования хеш-кодов $\mathrm{Y} = \mathrm{Hash}(\mathrm{K}, \mathrm{M}, \mathrm{Taglen})$. Вычисление значения функции $\mathrm{Hash}(\mathrm{K}, \mathrm{M}, \mathrm{Taglen})$ ключевого универсального хеширования информационного сообщения M с использованием секретного ключа K выполняется в три этапа (используется три уровня (слоя) ключевого хеширования) $\mathrm{Hash}_{\mathrm{L1}}$, $\mathrm{Hash}_{\mathrm{L2}}$ и $\mathrm{Hash}_{\mathrm{L3}}$, соответственно. Второй уровень хеширования $\mathrm{Hash}_{\mathrm{L2}}$ выполняется, только если длина хешируемого сообщения M превосходит 1024 байт.

Длина хеш-кода Y кратна 32 битам, его значение $\mathrm{Y} = \mathrm{Hash}(\mathrm{K}, \mathrm{M}, \mathrm{Taglen})$ для любой длины Taglen формируется посредством объединения (конкатенации) нескольких (от одной до четырех) последовательностей $\mathrm{Y}_{\mathrm{L3_i}}$

$$\mathrm{Y} = \mathrm{Hash}(\mathrm{K}, \mathrm{M}, \mathrm{Taglen}) = \mathrm{Y}_{\mathrm{L3_1}} \| \mathrm{Y}_{\mathrm{L3_2}} \| \ldots \| \mathrm{Y}_{\mathrm{L3_{It}}};$$

$$\mathrm{It} = \mathrm{Taglen} / 4,$$

где $\mathrm{Y}_{\mathrm{L3_i}}$ – результат многоуровневого хеширования сообщения M на i-ой итерации с использованием соответствующих ключей, $\mathrm{i} = 1, 2, \ldots, \mathrm{It}$.

Структурная схема итеративного формирования хеш-кодов Y, псевдослучайной подложки Pad и кода подлинности сообщений Tag представлена на рис. 1.

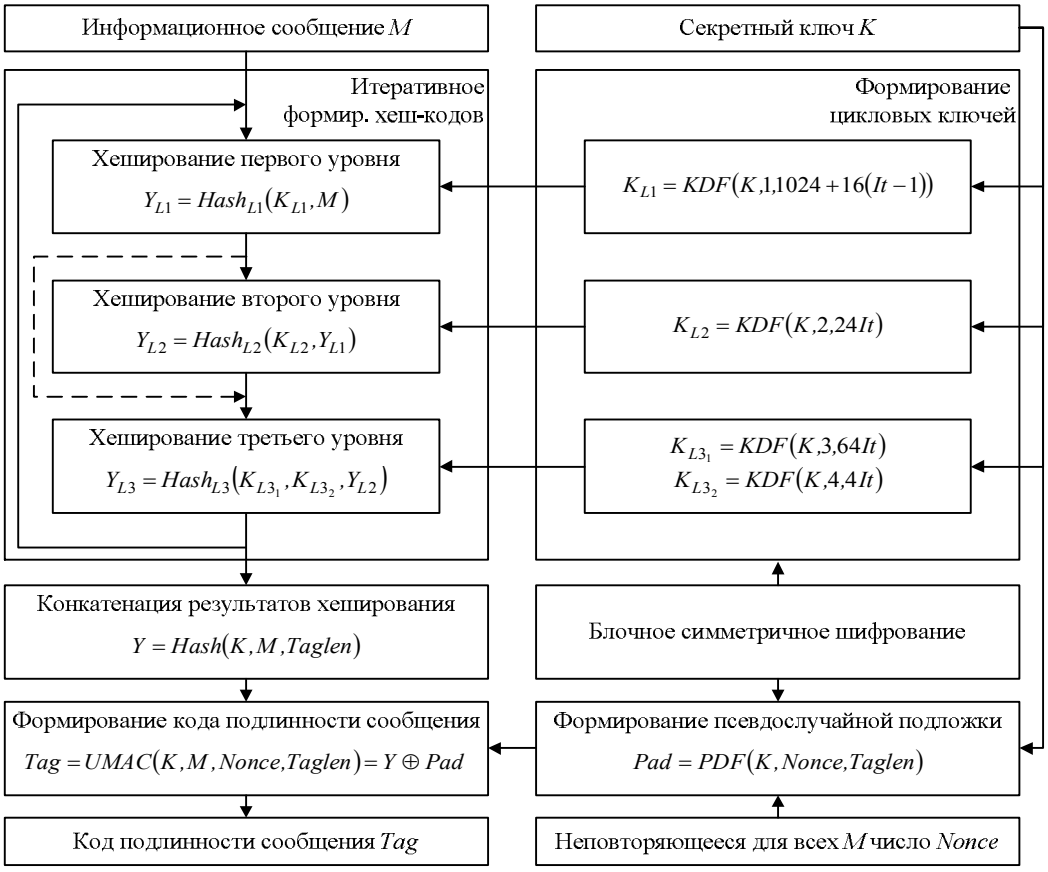

Рис. 1. Структурная схема итеративного формирования хеш-кодов Y, псевдослучайной подложки Pad и кода подлинности сообщений Tag

Рассмотрим формирование хеш-кода Y_{L3_i} на i-ой итерации. Для этого обозначим результат многоуровневого хеширования на произвольной i-ой итерации следующим образом:

$$Y_{L3_i} = Y_{L3} = Hash_{L3}\left(K_{L3_1}, K_{L3_2}, Hash_{L2}\left(\begin{matrix}K_{L2}, Hash_{L1}\\ (K_{L1}, M)\end{matrix}\right)\right),$$

где $Hash_{L1}(K_{L1}, M)$, $Hash_{L2}(K_{L2}, Y_{L1})$ и $Hash_{L3}(K_{L3_1}, K_{L3_2}, Y_{L2})$ – функции ключевого хеширования первого, второго и третьего уровня, управляемые и зависящими от номера итерации секретными ключами K_{L1}, K_{L2}, K_{L3_1}, K_{L3_2}, соответственно.

Функция ключевого хеширования первого уровня (Level 1) предназначена для формирования хеш-кода $Y_{L1} = Hash_{L1}(K_{L1}, M)$ по введенному информационному сообщению M, подлежащему аутентификации и представленному в виде массива-строки размерностью от одного до Y бит (2^{64} байт).

Функция ключевого хеширования второго уровня (Level 2), осуществляет формирование хеш-кода $Y_{L2} = Hash_{L2}(K_{L2}, Y_{L1})$. Если длина сообщения M равна или менее 1024 байт этот уровень хеширования не выполняется, вычисляется сразу хеш-код третьего уровня:

$$Y_{L3_i} = Y_{L3} = Hash_{L3}\left(K_{L3_1}, K_{L3_2}, Hash_{L1}(K_{L1}, M)\right).$$

Функция ключевого хеширования третьего уровня (Level 3) предназначена для формирования хеш-кода $Y_{L3} = Hash_{L3}(K_{L3_1}, K_{L3_2}, Y_{L2})$.

Вычисление соответствующих хеш-кодов Y_{L1}, Y_{L2} и Y_{L3} на каждом уровне реализуется под управлением собственного ключа: первый уровень управляется ключом K_{L1}, второй - K_{L2} и третий двумя ключами K_{L3_1} и K_{L3_2}, соответственно.

Ключевые последовательности K_{L1}, K_{L2}, K_{L3_1}, K_{L3_2} формируются по введенному секретному ключу reverse длины Keylen байт с использованием специальной функции $KDF(K, Index, Numbyte)$ (Key-Derivation Function – KDF), где Index и Numbyte представляют собой два целых положительных числа, не превосходящих 2^{64}:

$$K_{L1} = KDF(K, Index, Numbyte),\ Index = 1,$$
$$Numbyte = 1024 + 16(It - 1);$$
$$K_{L2} = KDF(K, Index, Numbyte),\ Index = 2,$$
$$Numbyte = 24It;$$
$$K_{L3_1} = KDF(K, Index, Numbyte),\ Index = 3,$$
$$Numbyte = 64It;$$
$$K_{L3_2} = KDF(K, Index, Numbyte),\ Index = 4,$$
$$Numbyte = 4It.$$

Таким образом, для каждой 2^x-ой итерации ($i = 1, 2, ..., It$) используется 4 ключа: K_{L1}, K_{L2}, K_{L3_1}, K_{L3_2} по 1024, 24, 64 и 4 байта соответственно (см. рис. 1).

1.2.1. Первый уровень хеширования выполняет разбиение массива-строки M размерности до 2^{64} байт на блоки M_i по 1024 байт с последующим преобразованием каждого блока функцией $NH(K_{L1}, M_i)$. Полученные результаты $Hash_{L1_i} = NH(K_{L1}, M_i)$ конкатенируются (объединяются) в строку $Y_{L1} = Hash_{L1}(K_{L1}, M)$, которая короче информационной последовательности в 128 раз. Эта строка и является результатом хеширования первого уровня:

$$Y_{L1} = Hash_{L1}(K_{L1}, M) =$$
$$= NH(K_{L1}, M_0) \| NH(K_{L1}, M_1) \| ... \| NH(K_{L1}, M_{n-1}),$$

где $n = \left[\frac{Length(M)}{1024}\right]$, $[x]$ – целая часть числа x; $Length(M)$ – байтовая длина информационного сообщения M.

Значение функции $Hash_{L1_i} = NH(K_{L1}, M_i)$ вычисляется по следующему правилу. Информационный блок M_i разбивается на четырехбайтовые подблоки так, что

$$M_i = M_{i_1} \| M_{i_2} \| ... \| M_{i_t},$$

где $t = \left[\frac{Length(M_i)}{4}\right]$. В данном случае $t = \left[\frac{1024}{4}\right] = 256$.

Аналогичным образом ключевая последовательность K_{L1} представляется в виде последовательностей четырехбайтовых подблоков:

$$K_{L1} = K_{L1_1} \| K_{L1_2} \| ... \| K_{L1_t}.$$

После чего (принимая начальное состояние $Hash_{L1_i} = 0$) для всех $j = 1, 9, 17..., t-7$ выполняются следующие операции:

$$Hash_{L1_i} = Hash_{L1_i} +_{64} ((M_{i_{j+0}} +_{32} K_{L1_{j+0}}) \times$$
$$\times_{64} (M_{i_{j+4}} +_{32} K_{L1_{j+4}}));$$
$$Hash_{L1_i} = Hash_{L1_i} +_{64} ((M_{i_{j+1}} +_{32} K_{L1_{j+1}}) \times$$
$$\times_{64} (M_{i_{j+5}} +_{32} K_{L1_{j+5}}));$$
$$Hash_{L1_i} = Hash_{L1_i} +_{64} ((M_{i_{j+2}} +_{32} K_{L1_{j+2}}) \times$$
$$\times_{64} (M_{i_{j+6}} +_{32} K_{L1_{j+6}}));$$

$$\mathrm{Hash}_{\mathrm{L1_i}} = \mathrm{Hash}_{\mathrm{L1_i}} +_{64} ((\mathrm{M}_{\mathrm{i_{j+3}}} +_{32} \mathrm{K}_{\mathrm{L1_{j+3}}}) \times \times_{64} (\mathrm{M}_{\mathrm{i_{j+7}}} +_{32} \mathrm{K}_{\mathrm{L1_{j+7}}})),$$

где $+_{64}$, $+_{32}$ – операции сложения по модулю 2^{64} и 2^{32}, соответственно; $\times_{64}$ – операция умножения по модулю 2^{64}.

В результате вычислений формируется восьмибайтное значение $\mathrm{Hash}_{\mathrm{L1_i}}$, упрощенную схему такого формирования представим на рис. 2 [1 – 4].

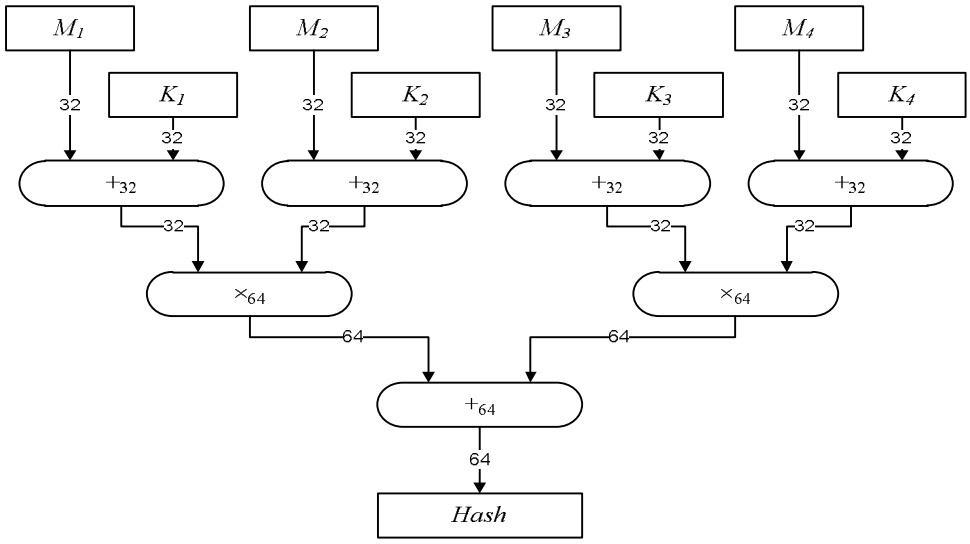

Рис. 2. Упрощенная схема формирования хеш-образа $\mathrm{Hash}_{\mathrm{L1_i}} = \mathrm{NH}(\mathrm{K}_{\mathrm{L1}}, \mathrm{M}_{\mathrm{i}})$

Рассмотренная функция ключевого хеширования NH принадлежит к классу универсальных хеширующих функций [1-4].

1.2.2. Второй уровень хеширования использует полиномиальное ключевое хеширование Poly, подробно рассмотренное в работах [14, 15]. Результатом работы этого уровня есть вычисление хеш-кода $\mathrm{Y}_{\mathrm{L2}} = \mathrm{Hash}_{\mathrm{L2}}(\mathrm{K}_{\mathrm{L2}}, \mathrm{Y}_{\mathrm{L1}}) = \mathrm{Poly}(\mathrm{Wordbits}, \mathrm{Maxwordrange}, \mathrm{k}, \mathrm{M_P})$, т.е. на вход хеширования второго уровня подается строка $\mathrm{Y}_{\mathrm{L1}} = \mathrm{Hash}_{\mathrm{L1}}(\mathrm{K}_{\mathrm{L1}}, \mathrm{M})$.

В качестве исходных данных функция полиномиального хеширования использует: $\mathrm{Wordbits} \in [64, 128]$; Maxwordrange – положительное целое число, меньшее 2^{Wordbits}; k – зависящее от ключа K_{L2} целое число из диапазона $[0, ..., \mathrm{prime(Wordbits)} - 1]$, $\mathrm{prime(x)}$ – наибольшее простое число, меньшее 2^{x}; $\mathrm{M_P} = \mathrm{Y}_{\mathrm{L1}} = \mathrm{Hash}_{\mathrm{L1}}(\mathrm{K}_{\mathrm{L1}}, \mathrm{M})$ – данные, подлежащие полиномиальному хешированию.

По спецификации алгоритма UMAC в качестве $\mathrm{prime(x)}$ используются следующие константы:

$$\mathrm{prime}(36) = 2^{36} - 5, \mathrm{prime}(64) = 2^{64} - 59,$$

$$\mathrm{prime}(128) = 2^{128} - 159.$$

Битовую длину $\mathrm{M_P}$ обозначим $\mathrm{Bytelength}(\mathrm{M_P})$. В зависимости от длины $\mathrm{M_P}$ используются следующие особенности в реализации второго уровня хеширования:

– если длина поступивших данных $\mathrm{M_P}$ не превосходит 2^{17} байт, тогда полиномиальное хеширование Poly выполняется с параметрами $\mathrm{Wordbits} = 64$; $\mathrm{Maxwordrange} = 2^{64} - 2^{32}$; $\mathrm{k} = \mathrm{k64}$ – строка, образованная первыми восемью байтами ключа K_{L2} и специальной восьмибайтной маской;

– если длина поступивших данных $\mathrm{M_P}$ превосходит 2^{17} байт (но не превосходит 2^{64} байт), тогда первые 2^{17} байт данных обрабатываются функцией полиномиального хеширования $\mathrm{Poly}(64, 2^{64} - 2^{32}, \mathrm{k64}, \mathrm{M_P})$, а оставшиеся байты данных обрабатываются функцией Poly с параметрами $\mathrm{Wordbits} = 128$;

$\mathrm{Maxwordrange} = 2^{128} - 2^{96}$; $\mathrm{k} = \mathrm{k128}$ – строка, образованная последними 16 байтами ключа K_{L2} и специальной 16 байтной маской.

Хешируемые данные $\mathrm{M_P}$ разбиваются на блоки по $\mathrm{Wordbytes} = \mathrm{Wordbits}/8$ байт:

$$\mathrm{M_P} = \mathrm{M_{P_1}} \| \mathrm{M_{P_2}} \| ... \| \mathrm{M_{P_n}},$$

где $\mathrm{n} = \mathrm{Bytelength(M_P)} / \mathrm{Wordbytes}$. Результатом хеширования является значение полиномиальной функции

$$\mathrm{Y_{L2}} = \left(\mathrm{M_{P_n}} + \mathrm{kM_{P_{n-1}}} + ... + \mathrm{k^{n-1}M_{P_1}} + \mathrm{k^n}\right) \mathrm{mod(p)},$$

которое вычисляется итеративной процедурой (для всех $\mathrm{i} = 1, 2, ..., \mathrm{n}$):

$$\mathrm{Poly_i} = \left(\mathrm{kPoly_{i-1}} + \mathrm{M_{P_i}}\right) \mathrm{mod(p)}, \ \mathrm{Poly_0} = 1,$$

$$p = \text{prime}(\text{Wordbits})$$

с помощью схемы Горнера

$$M_{P_n} + kM_{P_{n-1}} + ... + k^{n-1}M_{P_1} + k^n = \\ = (((k + M_{P_1})k + M_{P_2})k + ... + M_{P_{n-1}})k + M_{P_n}.$$

Вычисленное хеш-значение $Y_{L2} = \text{Poly}_n$ является целым числом из диапазона $[0,..., \text{prime}(\text{Wordbits}) - 1]$.

В работах [1 – 4] показано, что рассмотренная функция полиномиального ключевого хеширования $\text{Poly}(\text{Wordbits}, \text{Maxwordrange}, k, M_P)$ принадлежит к классу универсальных хеширующих функций.

1.2.3. Третий уровень хеширования $\text{Hash}_{L3}\left(K_{L3_1}, K_{L3_2}, Y_{L2}\right)$ выполняется над результатом полиномиального хеширования и преобразует поданные на его вход данные длины до 16 байт в хеш-код Y фиксированной длины 32 бита.

В качестве исходных данных третьего уровня хеширования выступают две ключевых последовательности K_{L3_1} и K_{L3_2} длины 64 и 4 байта соответственно, а также входная 16 байтная последовательность Y_{L2}.

Хешируемые данные Y_{L2} и ключевая последовательность K_{L3_1} равномерно разбиваются на восемь блоков, каждый из которых представляется как целое число Y_{L2_i} и $K_{L3_{1i}}$, $i = 1, 2, ..., 8$.

Хеш-значение Y_{L3} вычисляется следующим образом:

$$Y_{L3} = \left(\left(\left(\sum_{i=1}^{m} Y_{L2_i} K_{L3_{1i}}\right) \bmod(\text{prime}(36))\right) \bmod(2^{32})\right) \text{xor}(K_{L3_2}),$$

где $(x)\text{xor}(y)$ – операция “исключающего ИЛИ” над значениями x и y.

Функция ключевого хеширования $Y_{L3} = \text{Hash}_{L3}\left(K_{L3_1}, K_{L3_2}, Y_{L2}\right)$ принадлежит к классу универсальных хеширующих функций [1 – 4].

1.3. Схема формирования ключей (KDF: Key-Derivation Function). Специальная функция $\text{KDF}(K, \text{Index}, \text{Numbyte})$ предназначена для формирования последовательностей псевдо-случайных бит данных, которые используются на различных уровнях формирования кодов подлинности сообщений как ключевые данные соответствующих функций хеширования.

В качестве исходных данных функции генерации ключевых псевдослучайных последовательностей используется секретных ключ K длины Keylen байт и два положительных целых числа Index и Numbyte, значение которых не превосходит 2^{64}.

Для формирования псевдослучайных ключевых последовательностей используется блочный симметричный шифр. Обозначим процедуру шифрования блока данных T длины Blocklen байт с использованием секретного ключа K длины Keylen байт в виде некоторой функции $\text{Enchiper}(K, T)$. Тогда процедуру формирования псевдослучайной ключевой последовательности $K' = \text{KDF}(K, \text{Index}, \text{Numbyte})$ можно представить в виде следующего итеративного (для всех $i = 1, 2, ..., n$) преобразования:

$$T_i = \text{Index} \| i;\ K'_i = \text{Enchiper}(K, T_i);$$

$$K' = K'_1 \| K'_2 \| ... \| K'_n,$$

где $n = \left\lceil \frac{\text{Numbyte}}{\text{Blocklen}} \right\rceil$, $\lceil x \rceil$ – целая часть числа x; $a \| b$ – конкатенация (присоединение) строк a и b.

Сформированная последовательность псевдослучайных ключевых бит данных K' имеет длину Numbyte байт, кратную длине блока Blocklen байт.

1.4. Схема формирования псевдослучайной подложки (PDF: Pad-Derivation Function). Функция $\text{PDF}(K, \text{Nonce}, \text{Taglen})$ предназначена для формирования псевдослучайной подложки Pad, используемой на заключительном этапе формирования кода подлинности сообщения.

В качестве исходных данных используется секретных ключ K длины Keylen байт и неповторяющееся (для всех вводимых информационных сообщений M) восьмибайтное число Nonce, а также целое число Taglen, задающее размер (длину в байтах) формируемого кода подлинности Tag.

Процедура формирования псевдослучайной подложки $\text{Pad} = \text{PDF}(K, \text{Nonce}, \text{Taglen})$ состоит в формировании подключа

$$K' = \text{KDF}(K, \text{Index}, \text{Numbyte});\ \text{Index} = 0;$$

$$\text{Numbyte} = \text{Keylen},$$

с использованием рассмотренной выше процедуры формирования последовательностей псевдослучайных ключевых бит и шифрования значения Nonce на сформированном подключе K', т.е.:

$$\text{Pad} = \text{PDF}(K, \text{Nonce}, \text{Taglen}) = \\ = \text{Enchiper}\left(\text{KDF}(K, 0, \text{Keylen}), \text{Nonce}\right).$$

Процедура формирования псевдослучайной подложки Pad построена так, что результирующее значение Pad имеет длину Taglen байт вне зависимости от значений Blocklen и Nonce.

Таким образом, рассмотренная схема формирования кодов подлинности сообщений UMAC использует многоуровневую конструкцию универсального хеширования $\text{Hash}(K, M, \text{Taglen})$ и процедуру формирования псевдослучайной подложки

Pad. Применение универсального хеширования позволяет обеспечить равновероятность формирования хеш-образов для всего множества используемых ключевых данных, на чем и базируется доказательство безопасности алгоритма [1 – 4]. Формирование псевдослучайной подложки криптографически стойким алгоритмом (например, с использованием блочного симметричного шифра AES) обеспечивает криптостойкость алгоритма UMAC на уровне стойкости применяемого криптоалгоритма [5]. Следовательно, рассмотренная схема формирования UMAC обладает потенциально высокими показателями эффективности.

В тоже время на сегодняшний день не исследованы коллизионные свойства алгоритма UMAC после применения завершающей процедуры наложения на формируемые хеш-коды $\mathrm{Y} = \mathrm{Hash}(\mathrm{K}, \mathrm{M}, \mathrm{Taglen})$ псевдослучайных подложек $\mathrm{Pad} = \mathrm{PDF}(\mathrm{K}, \mathrm{Nonce}, \mathrm{Taglen})$. Ниже показано, что результирующие коды подлинности сообщений $\mathrm{Tag} = \mathrm{UMAC}(\mathrm{K}, \mathrm{M}, \mathrm{Nonce}, \mathrm{Taglen}) = \mathrm{Y} \oplus \mathrm{Pad}$ формируются не равновероятно для всего множества используемых ключевых данных. Следовательно, алгоритм формирования UMAC после применения последнего слоя наложения псевдослучайных подложек теряет свойство «универсальности» хеширования, его коллизионные свойства существенно ухудшаются.

2. Разработка уменьшенной модели формирования кодов подлинности сообщений UMAC (мини-UMAC). В основе предлагаемой методики исследования коллизионных свойств кодов аутентификации сообщений UMAC лежит использование уменьшенных моделей отдельных слоев используемых преобразований и оценка распределения коллизий (столкновений) формируемых образов (кодов).

Схема формирования кодов аутентификации сообщений UMAC использует в своей структуре несколько слоев преобразования, в том числе блочный симметричных шифр (рекомендован к использованию шифр AES). Разрабатываемая уменьшенная модель UMAC должна включать соответствующие слои преобразования с сохранением их алгебраической структуры при выполнении масштабирования до мини-версии. Естественным представляется исследовать коллизионные характеристики формируемых образов (кодов) на каждом из слоев преобразования, в том числе формируемых с помощью блочного симметричного шифра псевдослучайных подложек Pad, проанализировать их влияние на коллизионные свойства кодов аутентификации сообщений уменьшенной модели UMAC.

Выше было показано, что схема формирования кодов UMAC состоит из следующих слоев:

– трехуровневое универсальное хеширование для формирования хеш-кодов

$$\mathrm{Y} = \mathrm{Hash}(\mathrm{K}, \mathrm{M}, \mathrm{Taglen});$$

– криптографическое преобразования с использованием блочного симметричного шифра для формирования псевдослучайной подложки

$$\mathrm{Pad} = \mathrm{PDF}(\mathrm{K}, \mathrm{Nonce}, \mathrm{Taglen});$$

– заключительное преобразование для формирования кодов аутентификации сообщений

$$\mathrm{Tag} = \mathrm{UMAC}(\mathrm{K}, \mathrm{M}, \mathrm{Nonce}, \mathrm{Taglen}) = \mathrm{Y} \oplus \mathrm{Pad}.$$

Рассмотрим каждый слой схемы формирования кодов аутентификации сообщений UMAC на предмет их масштабирования.

2.1. Мини-версию трехуровневого универсального хеширования построим без изменения структуры алгебраических преобразований простым уменьшением размерности блоков обрабатываемых данных в восемь раз.

Соответствующая длина хеш-кода $\mathrm{Y}_{\mathrm{min}\,i}$ уменьшенной модели первого слоя будет кратна 4 битам, его значение сформируем посредством объединения (конкатенации) четырех последовательностей $\mathrm{Y}_{\mathrm{min}\,i\mathrm{L3}_i}$

$$\mathrm{Y}_{\mathrm{min}\,i} = \mathrm{Y}_{\mathrm{min}\,i\mathrm{L3}_1} \| \mathrm{Y}_{\mathrm{min}\,i\mathrm{L3}_2} \| \mathrm{Y}_{\mathrm{min}\,i\mathrm{L3}_3} \| \mathrm{Y}_{\mathrm{min}\,i\mathrm{L3}_4},$$

где $\mathrm{Y}_{\mathrm{min}\,i\mathrm{L3}_i}$ – результат многоуровневого хеширования сообщения уменьшенной модели первого слоя мини-UMAC.

Рассмотрим процесс формирования одного блока $\mathrm{Y}_{\mathrm{min}\,i\mathrm{L3}_i}$ (второй уровень хеширования в уменьшенной модели выполнять не будем):

$$\mathrm{Y}_{\mathrm{min}\,i\mathrm{L3}_i} = \mathrm{Y}_{\mathrm{min}\,i\mathrm{L3}} = \mathrm{Hash}_{\mathrm{min}\,i\mathrm{L3}}\left(\mathrm{K}_{\mathrm{min}\,i\mathrm{L3}_1}, \mathrm{K}_{\mathrm{min}\,i\mathrm{L3}_2}, \mathrm{Hash}_{\mathrm{min}\,i\mathrm{L1}}\left(\mathrm{K}_{\mathrm{min}\,i\mathrm{L1}}, \mathrm{M}_{\mathrm{min}\,i}\right)\right),$$

где $\mathrm{K}_{\mathrm{min}\,i\mathrm{L1}}$, $\mathrm{K}_{\mathrm{min}\,i\mathrm{L3}_1}$, $\mathrm{K}_{\mathrm{min}\,i\mathrm{L3}_2}$ – ключевые последовательности мини-UMAC; $\mathrm{Hash}_{\mathrm{min}\,i\mathrm{L1}}$ и $\mathrm{Hash}_{\mathrm{min}\,i\mathrm{L3}}$ – уменьшенные версии хеширования первого и третьего уровней соответственно.

На первом уровне массив-строка $\mathrm{M}_{\mathrm{min}\,i}$ размерности 32 бита преобразуется функцией $\mathrm{NH}(\mathrm{K}_{\mathrm{L1}}, \mathrm{M}_i)$. Эта строка и является результатом хеширования первого уровня:

$$\mathrm{Y}_{\mathrm{min}\,i\mathrm{L1}} = \mathrm{NH}_{\mathrm{min}\,i}\left(\mathrm{K}_{\mathrm{min}\,i\mathrm{L1}}, \mathrm{M}_{\mathrm{min}\,i}\right).$$

Значение функции $\mathrm{NH}_{\mathrm{min}\,i}\left(\mathrm{K}_{\mathrm{min}\,i\mathrm{L1}}, \mathrm{M}_{\mathrm{min}\,i}\right)$ вычисляется по следующему правилу. Информационный блок $\mathrm{M}_{\mathrm{min}\,i}$ разбивается на восемь четырехбитовых подблоков

$$\mathrm{M}_{\mathrm{min}\,i} = \mathrm{M}_{\mathrm{min}\,i_1} \| \mathrm{M}_{\mathrm{min}\,i_2} \| \ldots \| \mathrm{M}_{\mathrm{min}\,i_8}.$$

Аналогичным образом ключевая последовательность K_{L1} представляется в виде последова-

тельностей из восьми четырехбитовых подблоков:

$$K_{miniL1} = K_{miniL1_1} \| K_{miniL1_2} \| ... \| K_{miniL1_8}.$$

После чего (принимая начальное состояние $Hash_{L1} = 0$) выполняются следующие операции:

$$Hash_{miniL1} = Hash_{miniL1} +$$
$$+_8((M_{mini_0} +_4 K_{miniL1_0}) \times_8 (M_{mini_4} +_4 K_{miniL1_4}));$$
$$Hash_{miniL1} = Hash_{miniL1} +$$
$$+_8((M_{mini_1} +_4 K_{miniL1_1}) \times_8 (M_{mini_5} +_4 K_{miniL1_5}));$$
$$Hash_{miniL1} = Hash_{miniL1} +$$
$$+_8((M_{mini_2} +_4 K_{miniL1_2}) \times_8 (M_{mini_6} +_4 K_{miniL1_6}));$$
$$Hash_{miniL1} = Hash_{miniL1} +$$
$$+_8((M_{mini_3} +_4 K_{miniL1_3}) \times_8 (M_{mini_7} +_4 K_{miniL1_7})),$$

где $+_8$, $+_4$ – операции сложения по модулю 2^8 и 2^4, соответственно; $\times_8$ – операция умножения по модулю 2^8.

В результате вычислений формируется восьмибитное значение

$$Y_{miniL1} = Hash_{miniL1}.$$

Третий уровень хеширования преобразует поданные на его вход восьмибитные данные Y_{miniL1} в хеш-код Y_{miniL3} длины 4 бита. В качестве ключевых последовательностей выступают K_{miniL3_1} и K_{miniL3_2} длины 16 и 4 бита соответственно.

Хешируемые данные $Hash_{miniL1}$ и ключевая последовательность K_{miniL3_1} равномерно разбиваются на четыре блока, каждый из которых представляется как целое число Y_{miniL2_i} и $K_{miniL3_{1i}}$, $i = 1,2,...,4$.

Хеш-значение Y_{miniL3} вычисляется следующим образом:

$$Y_{miniL3} = \left(\left(\left(\sum_{i=1}^{4} Y_{miniL2_i} K_{miniL3_{1i}}\right) \bmod(17)\right) \bmod(2^4)\right) \text{xor}(K_{miniL3_2}),$$

где (x)xor(y) – операция “исключающего ИЛИ” над значениями x и y.

2.2. Мини-версия блочного симметричного шифра AES подробно рассмотрена в работах [8 – 13]. Кратко изложим одну из версий мини-шифра и обоснуем ее использование для формирования псевдослучайной подложки в мини-UMAC.

В соответствии [8 – 10] размер блока открытого текста мини-шифра равен 16 бит, которые обозначим четырьмя шестнадцатеричными числами h_0, h_1, h_2, h_3. Отметим, что h_0 состоит из первых четырех бит входного потока. Однако когда h_0 рассматривается как шестнадцатеричная цифра, первый бит рассматривается как бит высшего порядка. Например, входной блок 1000 1100 0111 0001 будет представлен $h_0 = 8$, $h_1 = c$, $h_2 = 7$, $h_3 = 1$.

Размер ключа также равен 16 бит. Обозначим его как 4 шестнадцатеричных числа k_0, k_1, k_2, k_3.

Шаги шифра применяются к состоянию - массиву 2×2 шестнадцатеричных цифр. Однако для рассматриваемой ниже операции $\tilde{\sigma}$ состояние будет представлено как массив 8×2 бит, т.е. каждая шестнадцатеричная цифра будет рассматривается как столбец 4 бит с битом высшего порядка сверху.

Входной блок загружается в состояние отображением h_0, h_1, h_2, h_3 в $\begin{bmatrix} h_0 & h_2 \\ h_1 & h_3 \end{bmatrix}$. Например, входной блок 1000 1100 0111 0001 будет загружен как $\begin{bmatrix} 8 & 7 \\ c & 1 \end{bmatrix}$, где матрица 8×2 будет $\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 1 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}$.

Baby-Rijndael включает несколько идентичных по структуре раундов (по умолчанию их 4). Перед шифрованием входной блок загружается в состояние, как описано выше и рассчитываются раундовые ключи. Шифрование имеет общую структуру: $E(a) = r_4 \circ r_3 \circ r_2 \circ r_1 \circ (a \oplus k_0)$, где a обозначает состояние, k_0, k_1, k_2, k_3, k_4 – раундовые ключи и $r_i(a) = (t \cdot \tilde{\sigma}(S(a))) \oplus k_i$, за исключением r_4, где пропущено умножение на t. В конце шифра состояние сгружается в 16-битный блок в таком же порядке, в котором он загружался.

Теперь опишем отдельные компоненты шифра.

SubBytes: Операция S есть выборочная таблица, которая применяется к каждой 16-ричной цифре состояния:

$$\begin{bmatrix} h_0 & h_2 \\ h_1 & h_3 \end{bmatrix} \xrightarrow{S} \begin{bmatrix} S(h_0) & S(h_2) \\ S(h_1) & S(h_3) \end{bmatrix},$$

где функция S задается следующей табл. 1.

Таблица 1

Выборочная таблица, реализующая S-блок Baby-Rijndael

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
S(x)	a	4	3	b	8	e	2	c	5	7	6	f	0	1	9	d

ShiftRows: Операция $\tilde{\sigma}$ просто меняет входы во второй строке состояния:

$$\begin{bmatrix} h_0 & h_2 \\ h_1 & h_3 \end{bmatrix} \xrightarrow{\tilde{\sigma}} \begin{bmatrix} h_0 & h_2 \\ h_3 & h_1 \end{bmatrix}.$$

MixColumns: Матрица t является следующей 8×8 матрицей бит:

$$\begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}.$$

Для этого преобразования состояние рассматривается как 8×2 битовая матрица. Состояние умножается слева на t, используя матричное умножение по модулю 2: $a = ta$.

KeySchedule: В начале шифра и в конце каждого раунда состояние побитно складывается (т.е. по модулю 2) с раундовым ключом. Столбцы раундовых ключей определены рекурсивно следующим образом:

$$w_0 = \begin{pmatrix} k_0 \\ k_1 \end{pmatrix},\ w_1 = \begin{pmatrix} k_2 \\ k_3 \end{pmatrix};$$

$$w_{2i} = w_{2i-2} \oplus S(\mathrm{reverse}(w_{2i-2})) \oplus r_i;$$

$$w_{2i+1} = w_{2i-1} \oplus w_{2i}$$

для всех $i = 1,2,3,4$, где $r_i = \begin{pmatrix} 2^{i-1} \\ 0 \end{pmatrix}$, а функция reverse заменяет два входа в столбец. Функция S та же, что и описанная выше.

Следует заметить, что все сложения выполняются побитно по модулю 2. Наконец, для $i = 1,2,3,4$ раундовый ключ k_i есть матрица, чьи столбцы есть w_{2i} и w_{2i+1}.

Использование рассмотренной уменьшенной модели блочного симметричного шифра AES позволяет провести экспериментальные исследования коллизионных свойств формируемых псевдослучайных подложек по всему множеству секретных ключей. Так, псевдослучайная подложка Pad_{mini} мини-UMAC формируется посредством шифрования неповторяющегося для каждого информационного сообщения M_{mini} числа Nonce. Результирующее значение Pad_{mini} имеет длину 16 бит, так же, как и соответствующая длина хеш-кода Y_{mini}.

2.3. Мини-версия заключительного преобразования для формирования кодов аутентификации сообщений мини-UMAC состоит в поразрядном суммировании по модулю 2 значений Y_{mini} и Pad_{mini}: $Tag_{mini} = Y_{mini} \oplus Pad_{mini}$.

Таким образом, масштабирование применяемых преобразований на соответствующих слоях схемы формирования кодов аутентификации сообщений, позволяет построить уменьшенную модель UMAC, экспериментально исследовать коллизионные свойства формируемых образов (кодов). На рис. 3. схематично изображен процесс масштабирования при разработке мини-версии UMAC.

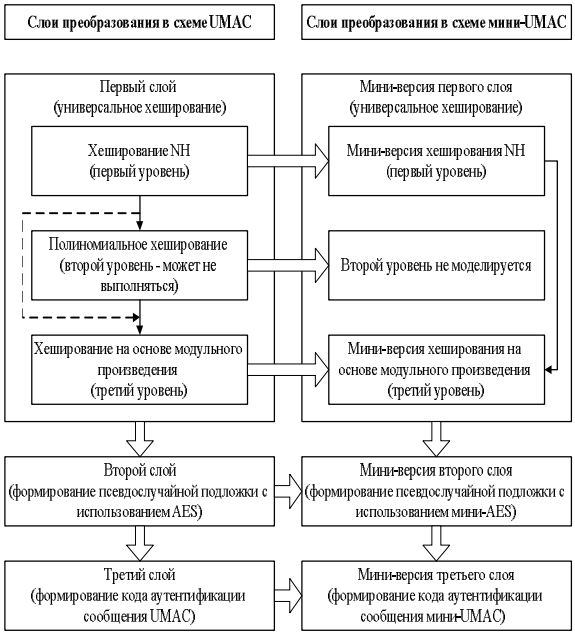

Рис. 3. Схема масштабирования UMAC

Коэффициент масштабирования при разработке мини-модели UMAC выбран таким образом, чтобы длина формируемых хеш-кодов Y, псевдослучайных подложек Pad и кодов аутентификации сообщений $Tag = Y \oplus Pad$ была равна длине блока мини-версии блочного симметричного шифра AES [8 – 13], т.е. 16 битам. Выбор такого коэффициента масштабирования позволяет с одной стороны сохранить алгебраическую структуру основных преобразований алгоритма UMAC, в том числе и входящего в его схему алгоритма AES, с другой стороны это дает возможность провести экспериментальные исследования с использованием методов статистической проверки гипотез и математической статистики. Таким образом, использование уменьшенных моделей отдельных слоев преобразований UMAC позволяет экспериментально исследовать коллизионные свойства формируемых кодов аутентификации сообщений.

Выводы

Проведенные исследования показали, что математический аппарат универсального хеширования позволяет строить эффективные схемы формирования кодов аутентификации сообщений. Современные многослойные конструкции позволяют обеспечить высокую эффективность MAC-кодов при обеспечении целостности и аутентичности информации.

Для исследования коллизионных свойств кодов аутентификации сообщений UMAC разработана модель (мини-UMAC), в основе построения которой лежит масштабирование отдельных слоев применяемых преобразований с сохранением их алгебраической структуры. ***Перспективным направлением дальнейших исследований*** является экспериментальная оценка коллизионных свойств UMAC-кодов с использованием разработанной уменьшенной модели, т.е. эмпирическая оценка числа правил хеширования, при которых существует коллизия, а также обоснование на основе анализа опытных данных конкретных предложений по совершенствованию механизмов обеспечения целостности и аутентичности данных в современных информационных системах.

Список литературы

1. UMAC: Fast and provably secure message authentication / J. Black, S. Halevi, H. Krawczyk, T. Krovetz, P. Rogaway // Advances in Cryptology - CRYPTO '99, LNCS. – Springer-Verlag, 1999. – Vol. 1666. – P. 216-233.

2. Krovetz T. Fast universal hashing with small keys and no preprocessing [Электронный ресурс] / T. Krovetz, P. Rogaway // Work in progress, 2000. – Режим доступа к ресурсу: http://www.cs.ucdavis.edu/~rogaway/umac.

3. UMAC -Message authentication code using universal hashing [Электронный ресурс] / T. Krovetz, J. Black, S. Halevi, A. Hevia, H. Krawczyk, and P. Rogaway. // IETF Internet Draft. – Режим доступа к ресурсу: www.cs.ucdavis.edu/~rogaway/umac, 2000.

4. Krovetz T. UMAC -Message authentication code using universal hashing [Электронный ресурс] / T. Krovetz // IETF Internet Draft. – Режим доступа к ресурсу: www.cs.ucdavis.edu/~rogaway/umac, 2004.

5. Final report of European project number IST-1999-12324, named New European Schemes for Signatures, Integrity, and Encryption, April 19, 2004 – Version 0.15 (beta), Springer-Verlag.

6. Krovetz T. UMAC – Message authentication code using universal hashing [Электронный ресурс] / T. Krovetz. 2006. – Режим доступа к ресурсу: http://www.cs.ucdavis.edu/~rogaway/umac.

7. Krovetz T. Software-Optimized Universal Hashing and Message Authentication. Dissertation submitted in partial satisfaction of the requirements for the degree of doctor of philosophy / T. Krovetz. – University Of California Davis. September 2000. – 269 p.

8. A Description of Baby Rijndael // ISU CprE/Math 533; NTU ST765-U. – 2003.

9. Raphael Chung-Wei Phan. Mini Advanced Encryption Standard (Mini-AES): A testbed for Cryptanalysis Students / Raphael Chung-Wei Phan // Cryptologia, XXVI(4). – October 2002. – P. 283-306.

10. Raphael Chung-Wei Phan. Impossible Differential Cryptanalysis of Mini-AES / Raphael Chung-Wei Phan // Cryptologia, Vol. XXVII, No. 4, October 2003.

11. Исследование дифференциальных свойств мини-шифров Baby-ADE и Baby-AES / В.И. Долгов, А.А. Кузнецов, Р.В. Сергиенко, О.И. Олешко // Прикладная радиоэлектроника. – Х.: ХНУРЭ, 2009. – Т. 8, № 3. – С. 252-257.

12. Долгов В.И. Исследование криптографических свойств нелинейных узлов замены уменьшенных версий некоторых шифров / В.И. Долгов, А.А. Кузнецов, И.В. Лисицкая, Р.В. Сергиенко, О.И. Олешко // Прикладная радиоэлектроника. – Х.: ХНУРЭ, 2009. – Т. 8, № 3. – С. 268-277.

13. Исследование дифференциальных свойств блочно-симметричных шифров. / Л.С. Сорока, А.А. Кузнецов, И.В. Московченко, С.А. Исаев // Системи обробки інформації: зб. наук. пр. – Х.: ХУПС, 2010. – Вип. 6 (87). – С. 286-294.

14. Carter J.L. Universal classes of hash functions / J.L. Carter, M.N. Wegman // Computer and System Scince. – 1979. – № 18. – P. 143-154.

15. Wegman M.N. New hash functions and their use in authentication and set equality / M.N.Wegman, J.L. Carter // Computer and System Scince. – 1981. – № 22. – P. 265-279.

Поступила в редколлегию 20.04.2011

Рецензент: д-р техн. наук, проф. Ю.В. Стасєв, Харківський університет Повітряних Сил ім. І. Кожедуба, Харків.

МОДЕЛЬ ФОРМУВАННЯ КОДІВ АУТЕНТИФІКАЦІЇ ПОВІДОМЛЕНЬ З ВИКОРИСТАННЯМ УНІВЕРСАЛЬНИХ ГЕШУЮЧИХ ФУНКЦІЙ

О.О. Кузнецов, О.Г. Король, В.В. Босько

Досліджуються методи й моделі формування кодів автентичності повідомлень (MAC – Message Authentication Code), у тому числі з використанням універсальних гешуючих функцій (UMAC – Message Authentication Code using Universal Hashing). За допомогою масштабування застосовуваних перетворень розробляється зменшена модель UMAC (міні-UMAC) для дослідження колізійних властивостей формованих кодів автентичності повідомлень.

Ключові слова: *MAC-коди, універсальні геш-функції, модель UMAC, колізійні властивості.*

MODEL OF FORMING OF CODES OF AUTHENTIFICATION OF MESSAGES WITH THE USE OF UNIVERSAL HASH FUNCTIONS

A.A. Kuznetsov, O.G. Korol', V.V. Bos'ko

They are researched methods and models of the shaping the codes to authentifications of the messages (the MAC - Message Authentication Code), including with use universal hash function (UMAC - Message Authentication Code using Universal Hashing). By means of scaling the applicable transformations is developed reduced model UMAC (mini-umac) for study conflict characteristic of the formed codes to authentifications of the messages.

Keywords: *COLOR-codes, universal hash-functions, model UMAC, conflict characteristic/*