Conf. Globecom-94. – 1994. – San Francisco. – CA. – November. – P. 339-343. ***4.*** *Barbulescu A., Pietrobon S. Turbo Codes: a tutorial on a new class of powerful error correcting coding schemes. Part I: Code Structures and Interleaver Design // University of South Australia, 1998. – October. – P. 1 –22.* ***5.*** *Berrou C., Glavieux A.,Thitimajshima P. Near Shannon Limit Error-Correcting Coding and Decoding: Turbo-Codes // Proc. Int. Conf. On Commun., ICC-93. – Geneva, 1993. – May. – P. 1064 – 1070.* ***6.*** *Hokfelt J., Edfors O., Maseng T. Interleaver Design for turbo codes based on the perfomance of iterative decoding // P. 1 – 5.* ***7.*** *Банкет В. Л., Прокопов С. Д., Постовой А. Г., Топорков Ф. В. Экспериментальное исследование процедур кодирования / декодирования турбокодов // Зв'язок. – 2004. – № 6. – С. 57-58.* ***8.*** *Ливенцев С. П., Алексеев Д. А., Зайцев С. В. Анализ характеристик перемежителей, используемых в турбокодах // Зв'язок. – 2005. – № 3. – С. 57-61.* ***9.*** *Qi J. Turbo code in IS-2000 code division multiple access communications under fading // Wichita State University. – 1999. P. 9-16.* ***10.*** *Woodard J., Hanzo L. Comparative Study of Turbo Decoding Techniques: An Overview // IEEE Transactions on Vehicular Technoligy, Vol. 49, No. 6, 2000. - November. P. 2208 – 2232.* ***11.*** *Robertson P., Villebrun E., Hoeher P. A comparison of optimal and sub-optimal MAP decoding algorithms operating in the log domain // in Proc. Int. Conf. on Commun., ICC-95. – 1995. – June. – P. 1009-1013.*

УДК 681.3.06

ВЫСОКОСКОРОСТНОЙ UMAC АЛГОРИТМ

Геннадий Халимов
Харковский национальний университет радиоэлектроники

Анотація: **Розглядається високошвидкісний UMAC алгоритм, характеристики, параметри та оцінка таємності.**
Suumary: **The high-speed UMAC algorithm, characteristics, parameters and an estimation of privacy are considered.**
Ключевые слова: **UMAC алгоритм, универсальные хеш-функции.**

I Введение

UMAC алгоритм, известный в модификациях UMAC (1999) [1] и UMAC (2000) [2], представлен проекту NESSIE как программно ориентированный для реализации на современных операционных платформах и обеспечивает чрезвычайно высокую скорость вычислений. Разработчики UMAC преследовали две главные цели:

- ✓ высокую скорость вычислений;
- ✓ доказуемую секретность.

Решение этих задач оказалось возможным на основе применения композиционной схемы с многократным универсальным хешированием и криптографическим вычислением тега аутентификации.

Задачей данной статьи является анализ основных характеристик UMAC кодов, изложение алгоритма построения и оценка секретности. С этой целью в разделе 2 приводятся основные определения универсальных хеш-функций UMAC реализации. В разделе 3 рассматриваются версии алгоритма UMAC (1999) и UMAC (2000). В разделе 4 оценена секретность UMAC схемы.

II Определения универсальных хеш-функций UMAC реализации

Алгоритм UMAC является композиционной схемой универсального хеширования. Коды подлинности сообщений, основанные на универсальном хешировании, используют определение семейства хеш-функций [3].

Определение 1. $H=\{h: D\to R\}$ есть семейство хэш-функций с общей областью определения D и конечным диапазоном значений R.

MAC коды являются отображением $H: K\times D\to R$, где $h = H_K: D\to R$ – функция, определенная для каждого $K \in K$ с заданным распределением на H. Основные универсальные семейства хэш-функций имеют следующие представления.

1. H является универсальным семейством хэш-функций $(\varepsilon\text{-}U)$, если для всех $x\neq y\in D$,

$$Pr_{h\in H}[h(x)=h(y)]=\varepsilon,\ \varepsilon=1/|R|.$$

2. H является Δ-универсальным семейством хэш-функций $(\varepsilon\text{-}\Delta U)$, если для всех $x\neq y\in D$ и всех $a\in R$,

$$Pr_{h\in H}[h(x)-h(y)=a]=\varepsilon,\ \varepsilon=1/|R|.$$

3. H является почти Δ-универсальным семейством хэш-функций (ε-AΔU), если для всех $x \neq y \in D$ и всех $a \in R$,

$$Pr_{h \in H}[h(x)-h(y)=a] \leq \varepsilon.$$

4. H является строго универсальным семейством хэш-функций (ε-SU), если для всех $x \neq y \in D$ и всех $a,b \in R$,

$$Pr_{h \in H}[h(x)=a,\ h(y)=b] = \varepsilon,\ \varepsilon = 1/|R|^2.$$

5. H является почти строго универсальным семейством хэш-функций (ε-ASU), если для всех $x \neq y \in D$ и всех $a,b \in R$,

$$Pr_{h \in H}[h(x)=a,\ h(y)=b] \leq \varepsilon.$$

Отправной точкой для построения хеш функций в алгоритме UMAC является полилинейная модулярная MMH^+ конструкция (multilinear modular hashing) Картера-Вегмана.

Определение 2. Пусть p – простое число и k целое число, $k>0$. Семейство функций MMH^+ есть отображение $Z_p^k \rightarrow Z_p$, $h_x(m)=\sum_{i=1}^{k} m_i x_i \bmod p$ для $x=(x_1, x_2, \ldots, x_k)$, $m=(m_1, m_2, \ldots, m_k)$, x_i, $m_i \in Z_p$.

Семейство MMH^+ является Δ-универсальным [4].

Модификация полилинейной модулярной схемы с учётом особенности 32-й разрядной арифметики приводит к семейству функций MMH_{32} [4].

Определение 3. Пусть $p=2^{32}+15$ и $k=32$. Семейство функций MMH_{32} есть отображение $(\{0,1\}^{32})^k \rightarrow \{0,1\}^{32}$, $h_x(m)=[[[\sum_{i=1}^{k} m_i x_i] \bmod 2^{64}] \bmod (2^{32}+15)] \bmod 2^{32}$ для $x=(x_1, x_2, \ldots, x_k)$, $m=(m_1, m_2, \ldots, m_k)$.

Семейство MMH_{32} является ε-AΔU, $\varepsilon = 6 \cdot 2^{-32}$ [4].

Дальнейшее улучшение MMH конструкции достигается за счет применения нелинейной схемы NMH_{32}, уменьшающей в два раза число умножений.

Определение 4. Пусть $p=2^{32}+15$ и $k=32$. Семейство функций NMH_{32} (nonlinear modular hashing) есть отображение $(\{0,1\}^{32})^k \rightarrow \{0,1\}^{32}$, $h_x(m)=[[[\sum_{i=1}^{k/2} (x_{2i-1}+_{32}m_{2i-1})(x_{2i}+_{32}m_{2i})] \bmod 2^{64}] \bmod (2^{32}+15)] \bmod 2^{32}$, для $x=(x_1, x_2, \ldots, x_k)$, $m=(m_1, m_2, \ldots, m_k)$, где символ $+_{32}$ обозначает сложение по модулю 2^{32}.

Семейство NMH_{32} является ε-AΔU, где $\varepsilon = 2^{-32}$ [4].

UMAC код использует NH хеширование, которое является усовершенствованием алгоритмов NMH и MMH.

Определение 5. Полилинейное NH хеширование определяется выражением

$$h_x(m)=\sum_{i=1}^{k/2} (x_{2i-1}+_w m_{2i-1})(x_{2i}+_w m_{2i}) \bmod 2^{2w}, \quad (1)$$

где $x=(x_1, x_2, \ldots, x_k)$, $m=(m_1, m_2, \ldots, m_k)$, x_i, $m_i \in (0;\ldots; 2^w-1)$, символ $+_w$ обозначает сложение по модулю 2^w и $w=16, 32$.

Вычисление NH в кольцах $Z/2^w$ и $Z/2^{2w}$ является эффективным для машинной арифметики. Следующая теорема определяет вероятность коллизии для NH хеширования [2].

Теорема 1. Для четного $k \geq 2$ и $w \geq 1$, NH является 2^{-w} – AU семейством хеш-функций.

При вычислении NH функции (1) строки в выражении интерпретируются как последовательности целых чисел без знака. На практике арифметика чисел осуществляется с использованием знака, то есть x_i, $m_i \in (-2^{w-1};\ldots; 2^{w-1}-1)$. Это в два раза увеличивает вероятность коллизии и NH в знаковой версии является 2^{-w+1} – AU семейством хеш-функций.

NH семейство хеш-функций очень быстро вычисляется и длина ключа равна размеру сообщения. Ограничение ключевого размера приводит к увеличению числа хеш-значений из-за многократного хеширования нескольких блоков данных. Хеш код после NH хеширования имеет переменную длину. Чтобы сделать NH функцию практически более полезной, необходимо включить её в композиционную

схему с функцией, которая использует короткий ключ и производит выход фиксированной длины. В качестве такой функции можно использовать быструю полиномиальную функцию универсального хеширования, скалярную функцию вариационно-универсального хеширования или криптографическую хеш-функцию. Все эти возможности используются в спецификации алгоритма UMAC.

Быстрое полиномиальное универсальное хеширование.

Определение 6. Пусть p – простое число и k целое число, $k>0$. Полиномиальное хеширование PolyCW определяется выражением

$$h_x(m)=\sum_{i=0}^{k} m_i x^i \bmod p, \qquad (2)$$

где $x\in Z_p$, $m=(m_1, m_2, \ldots, m_k)$, $m_i\in Z_p$.

PolyCW хеширование (polynomial Carter-Wegman hashing) – одно из наиболее известных универсальных семейств хеш-функций [5]. Главное свойство PolyCW функции определяется фундаментальной теоремой алгебры, которая определяет, что многочлен, отличный от нуля степени не меньше k, имеет не меньше k корней.

Теорема 2. Полиномиальное семейство хеш функций PolyCW является ε-U универсальным и вероятность коллизии $\Pr_{h\in H}\{h_x(a)=h_x(b) \bmod p\}=k/p$.

Вычисление PolyCW хеш-функции можно осуществить по итерационной схеме Горнера с одной операцией умножения и сложения в конечном поле на каждом шаге. Важным аспектом практической реализации, который учитывает 32-й разрядную машинную арифметику, является выбор простого поля $Z_{p(w)}$, где $p(w)$ – w-битовое простое число. Арифметика $GF(2^w)$ является менее удобной для современных микропроцессоров, хотя обеспечивает легкое разделение хешируемых битовых строк на w-битовые подстроки. Лучший результат достигается в арифметике $Z_{p(w)}$ при как можно большом простом числе $p(w)\le 2^w$. В табл. 1 представлены простые числа, используемые в UMAC.

Таблица 1 – Простые числа для UMAC алгоритма

w	p(w) (decimal)	p(w) (hexadecimal)
19	2^{19} -1	0x0007FFFF
32	2^{32} -5	0xFFFFFFFB
36	2^{36} -5	0x0000000F FFFFFFFB
64	2^{64} -59	0xFFFFFFFF FFFFFFC5
128	2^{128} -159	0xFFFFFFFF FFFFFFFF FFFFFFFF FFFFFF61

Схема Горнера предполагает вычисление $y\leftarrow xy+m \bmod p(w)$. Обычное применение ассемблерной операции mod p(32) потребует 12.4 cpb (циклов процессора на байт) Pentium II. UMAC использует более эффективный алгоритм Poly32 для вычисления $y\leftarrow xy+m \bmod p(32)$, $p(32)= 2^{32}-5$. Используя представление $xy=a2^{32}+b$ и учитывая, что $a2^{32}=5a$ в $Z_{p(32)}$, получим $y=xy+m \bmod p(32)=5a+b+m \bmod p(32)$, что потребует одного умножения и два сложения на 32 разрядных регистрах и нескольких инструкций, контролирующих выход результата вычислений за пределы диапазона представления чисел в $Z_{p(32)}$. Данная реализация использует 8 строк ассемблерной программы и достигает производительности 3.69 cpb Pentium II. Расплатой за высокую скорость вычислений является уменьшение ключевого пространства до величины $2^{32}/5=2^{29}$ и увеличение вероятности коллизии до $k2^{-29}$. Практическая схема алгоритма Poly32 включает ещё одно дополнительное преобразование. Векторы чисел с 32 битами, которые имеют элементы вне $Z_{p(w)}$, преобразуются в вектор, который их не имеет, с помощью так называемого двойного представления. Это приводит к удвоению размера данных и соответственно к увеличению вероятности коллизии до $k2^{-28}$ и несколько снижает скорость вычислений до значения 3.86 cpb Pentium II.

Аналогичная реализация Poly64 хеш-функции для $p(64)= 2^{64}-59$ имеет вероятность коллизии $k2^{-49}$, использует 40 строк ассемблерной программы и имеет пиковую производительность 6.86 cpb Pentium II.

Линейное возрастание вероятности коллизии для полиномиальной хеш-функции ограничивает размер хешируемого сообщения (см. теорему 2). Чем больше размер ключевого пространства, тем большее количество слов можно хешировать до достижения допустимой вероятности коллизии. Мощность множества ключей определяется значением простого числа p(w), определяющего поле $Z_{p(w)}$. При увеличении p(w), возрастают временные затраты на вычисление многочленов в $Z_{p(w)}$. При движении p(w) от 2^{32} к 2^{64} временные затраты увеличиваются в два раза.

Для оптимизации временных затрат в UMAC используется RPHash (ramped polynomial hashing)

алгоритм, который является композиционной полиномиальной схемой RP хеширования и объединяет три алгоритма Poly32, 64 и 128, которые при хешировании последовательно используются по мере возрастания длины сообщения. Согласно RPHash, если сообщение M короче 2^{17} бит, то результат хеша - Poly32(M). Если M больше 2^{17} бит, хеш рассчитывается как Poly64(Poly32(M_1)|| M_2), $M=M_1|| M_2$. Для случая M более длинного, чем 2^{39} бит по аналогичной схеме используется Poly128.

Вероятность коллизии в RPHash схеме хеширования ограничивается значением максимума вероятности коллизии на шагах Poly32, 64 или 128.

Скалярные вариационно-универсальные хеш-функции.

Семейство вариационных универсальных хеш-функций впервые было представлено в [6].

Определение 7. Пусть $\varepsilon \in R^+$ положительное число. Зафиксируем множества *D* и *R*. Конечный набор хеш функций $H=\{h:D\to R\}$ называется ε-вариационным универсальным (ε-VU) (variational-universal) если для некоторых $m,m' \in D$, всех a, c, $c' \in R$, и для всех функций $f: R\to\{0,1\}$,

1 $\Pr_{h\in H}\{h(m)=c\}=1/|R|$;

2 $\Pr_{h\in H}[f(h(m))=1 \mid h(m')=c'] - \Pr_{y\in B}[f(y)=1] \le \varepsilon$.

Вариационный универсальный класс устанавливает требование к псевдослучайности символов хеш кода. Второе условие можно интерпретировать следующим образом. Алгоритм f является пользовательским приложением, которое использует результат хеширования. Наблюдатель изучает один из двух сценариев: y, где y – случайная строка в диапазоне хэш-функции, или h(m), где h – случайная хэш-функция, такая, что h(m')=c'. Наблюдатель не может различить, какой из этих двух сценариев имеет место в действительности с вероятностью, превышающей ε.

Интерес к ε-VU хеш-функциям определяется, прежде всего, тем, что вычислительно они могут быть более эффективными по сравнению с SU классами. SU семейства хэша имеют тенденцию к интенсивным вычислениям и часто требуют длинные ключи в специальных приложениях, в то время как недавнее исследование программно-эффективных AU хэш-функций привело к семействам, которые обрабатывают сообщения с очень низкой скоростью – 0.5 cpb [1, 4, 7].

Применение композиционных схем является основным методом построения VU хеш-функций [6].

Определение 8. Пусть $H=\{h: A\to B\}$ и $G = \{g: B\to C\}$ есть два семейства функций. Композиционное семейство функций $G\circ H = \{f: A\to C\}$ определяется через $f= g\circ h$, для всех $g\in G$ и всех $h\in H$.

Практическая конструкция VU хеш семейства определяется следующей теоремой [6].

Теорема 3. Если $H^{au}=\{h: A\to B\}$ является ε-AU, и $H^{su}=\{g: B\to C\}$ является SU, тогда $H^{su}H^{au}$ является $(\varepsilon-\varepsilon/|C|)$ – VU семейством хеш-функций.

В UMAC конструкции объединены быстрые полиномиальные алгоритмы Poly32, 64, 128 AU хеширования в RPHash схеме с медленной совершенной SU хеш-функцией скалярного произведения IPHASH (inner product hash).

Хеш-функция скалярного произведения IPHASH вычисляется следующим образом:

1 $Y=\sum_{i=1}^{n} m_i k_i \bmod p(w)$,

2 $Y=Y+_w T$,

3 $Y=Y[n_1 \ldots n_2]$,

где $k=(k_1,k_2,\ldots,k_n)$, $k_i \in Z_{p(w)}$, $m=m_1|| m_2|| \ldots || m_n$, $m\in\{0,1\}^u$, $|m_i|=u/n$, $T\in\{0,1\}^w$, символ $+_w$ обозначает сложение по модулю 2^w, $n_1<n_2\le w$.

На первом шаге вычисляется скалярное произведение слов сообщения m и ключа k с приведением по модулю простого числа p(w), ближайшего по величине к 2^w. По второй и третьей инструкции результат складывается по модулю 2^w с ключевым словом T с последующей выборкой хеш результата из области $n_1 \div n_2$ бит вектора Y.

IPHASH алгоритм в проекте UMAC представлен в двух модификациях:

– IPHASH16 с параметрами u=128, n=8, w=19, $p(w)=2^{19}-1$, $n_1=4$, $n_2=19$;

– IPHASH32 с параметрами u=192, n=6, w=36, $p(w)=2^{36}-5$, $n_1=5$, $n_2=36$.

Криптографические хеш-функции.

Известны два подхода к построению криптографических хеш-функций:

1 с применением блочных шифров в режиме CBC-MAC (ISO/IEC 9797-1);

2 на основе безключевых хэш-функций HMAC (ISO/IEC 9797-2).

Секретность этих конструкций может быть доказана при условии, что основной блочный шифр или ключевая хэш-функция являются псевдослучайными.

Спецификация UMAC не устанавливает криптопримитивы для схем CBC-MAC и HMAC. Основным

критерием выбора является защищенность и скорость вычисления криптографических хеш-функций. Были рассмотрены алгоритмы RC6 и SHA-1. Семейство функций CBCMAC-RC6 выполняется не быстрее чем 15 cpb, что является самым скоростным среди блочных шифров. HMAC на основе SHA-1 ограничивается значением скорости 13.1 cpb. Алгоритм MD5 в HMAC конструкции имеет производительность 5.3 cpb, но менее предпочтительный из-за обнаруженных недостатков секретности [8].

Криптографическая хеш-функция использует вычисление криптографического примитива на каждом байте сообщения. Применение RC6 или SHA-1 ограничивает скорость вычислений MAC кода значением 13 cpb.

III UMAC: описание алгоритма

UMAC алгоритм является программно ориентированным для процессоров, которые хорошо поддерживают умножение с 32 разрядными целыми числами и обеспечивает гигабитные скорости вычислений. UMAC впервые был представлен в 1999 году и с изменениями в 2000.

UMAC (1999) вычисляет MAC код путем предварительного сжатия сообщения с установленным отношением, используя NH универсальное семейство хэш-функции. Алгоритм имеет параметры: размер блока, размер слова, псевдослучайный генератор PRG, который используется для вычисления необходимого ключевого материала из ключа пользователя, и псевдослучайную функцию PRF для обработки сжатого сообщения и показателя новизны. Вычисление аутентификационного тега состоит из следующих шагов:

Генерирование подключа:

PRG: $Key \rightarrow K = K_1K_2,...,K_{1024}$, где $\forall K_i$ является 32 битовым словом, и $Key \rightarrow A$ длиной 512бит.

Вычисление хеш значения Msg в HM = $NHX_{Key}(Msg)$:

Len=|Msg| mod 4096, |Len|=16 бит.

В конец Msg добавляются минимальное число из 0 бит, так чтобы |Msg| было делимым на 64.

$Msg= Msg_1||Msg_2|| \_ \_ \_ ||Msg_t$, где каждый Msg_i – 1024 слово за исключением Msg_t, которое лежит между 2 и 1024 словами.

$HM=NH_K(Msg_1) || NH_K(Msg_2)|| \_ \_ \_ ||NH_K(Msg_t)||Len$.

Вычисление аутентификационного тега:

$Teg=HMAC\text{-}SHA1_A$ (Показатель новизны||HM).

NH хеширование вычисляется для сообщения, предварительно разбитого на блоки фиксированной длины (за исключением последнего блока, который может быть короче). Для блока из 1024 слов по 32 бита, можно получить значение хеша в 64 бит, что определяет коэффициент сжатия 512. Показатель новизны вместе с хеш-значениями всех блоков и информацией о длине объединяются в одну строку. Результирующая строка обрабатывается псевдослучайной функцией PRF для получения аутентификационного тега. В качестве PRF функции применяется одна из криптографических хеш-функции в режиме CBC-MAC или HMAC.

Шаг, связанный с генерированием подключа, является типичным для многих современных шифров и для длительных сеансов не оказывает существенного влияния на производительность алгоритма.

Универсальное хеширование для рассмотренного алгоритма имеет вероятность коллизии 2^{-32}. Понижение вероятности коллизии до значения 2^{-64} возможно за счет связывания результатов двойного хеширования сообщения с независимыми ключами. Важная оптимизация в UMAC состоит в применении теплицевой конструкции к формированию ключей; один ключ является производным из другого с добавлением нескольких новых слов.

Чтобы получить хорошее сжатие с более коротким подключом, можно использовать хеширование с двумя уровнями (2L). Если ключ хеша длины n_1 дает отношение сжатия λ_1 и ключ хеша длины n_2 дает отношению сжатия λ_2, то, используя двухуровневое хеширование, получим отношение сжатия $\lambda_1\lambda_2$ с ключевым размером n_1+n_2.

Алгоритм UMAC и спецификации были разработаны так, чтобы обеспечить параллельные вычисления в SIMD архитектуре. SIMD архитектура обеспечивается регистрами, которые в некоторых инструкциях могут обращаться с малоразмерными словами как векторами. Одна из самых быстрых реализаций UMAC использует MMX инструкции Pentium, которые обращаются с 64 битовым регистром как с четырехмерным вектором 16 битовых слов.

В зависимости от установки начальных параметров алгоритма UMAC изучены пять схем: UMAC-STD-30, UMAC-STD-60, UMAC-MMX-15, UMAC-MMX-30 и UMAC-MMX-60.

UMAC-STD-30 и UMAC-STD-60 используют слова размерности w = 32 бит. Они реализуют 2L хеширование с коэффициентом сжатия 32 и с последующим коэффициентом сжатия 16. Подключ K содержит приблизительно 400 Байтов. Используют HMAC-SHA1 в качестве основной PRF функции.

Арифметика вычислений знаковая. Отличие между UMAC-STD-30 и UMAC-STD-60 состоит в вероятности коллизии, соответственно равной 2^{-30} и 2^{-60}.

UMAC-MMX-15, UMAC-MMX-30 и UMAC-MMX-60 хорошо приспособлены для SIMD-параллелизма. Они используют слова с w=16 бит. Хеширование выполняется с одним уровнем и ключом, равным приблизительно 4кбт, что дает тот же коэффициент сжатия, как и схема 2L. Криптографическая функция используется в режиме CBC-MAC с блочным шифром RC6 [34]. Арифметика вычислений знаковая. Различие между UMAC-MMX-15, UMAC-MMX-30 и UMAC-MMX-60 состоит в максимальной вероятности подделки: 2^{-15}, 2^{-30} и 2^{-60}, соответственно.

Результаты испытаний схем UMAC (1999) с оценкой скорости вычислений представлены в табл. 2 [2].

Таблица 2 – Пиковые скорости вычисления UMAC, измеренные в Гбит/сек (циклы/байт)

	Pentium II	PowerPC	Альфа
UMAC-STD-60	1.49 (1.93)	1.81 (1.58)	1.03 (2.78)
UMAC-STD-30	2.79 (1.03)	2.28 (1.26)	1.79 (1.60)
UMAC-MMX-60	2.94 (0.98)	4.21 (0.66)	0.287 (10.0)
UMAC-MMX-30	5.66 (0.51)	7.20 (0.39)	0.571 (5.02)
UMAC-MMX-15	8.47 (0.33)	10.5 (0.27)	0.981 (2.85)
CBC-MAC-RC6	0.162 (17.7)	0.210 (13.7)	0.068 (42.5)
HMAC-SHA1	0.227 (12.6)	0.228 (12.6)	0.117 (24.5)

Максимальный коэффициент сжатия достигается на сообщениях 4 Кбт. UMAC выполняется лучше всего на длинных сообщениях, потому что хэш-функция является тогда более эффективной из-за сокращения количества вычислений, приходящихся на PRF. Некоторый выигрыш в скорости появляется уже при длинах сообщения в пару сотен байт.

Повышение скорости вычислений особенно при маленьких сообщениях было достигнуто UMAC (2000). В UMAC (2000) предложено две схемы: UMAC32 (без SIMD параллелизма) и UMAC16 (с SIMD параллелизмом), что позволило достигнуть производительности в три раза большей, чем в первоначальных версиях UMAC-STD и UMAC-MMX (см. табл. 3).

Высокая скорость обеспечивается за счет того, что тег аутентификации вычисляется по схеме: результат хеширования $\oplus$ PRF(показатель новизны), который посылается получателю вместе с сообщением и показателем новизны.

Таблица 3 – Производительность алгоритмов UMAC для различных длин хешируемых данных (циклы/байт)

	43 Бт	256 Бт	1500 Бт	256 КБт
UMAC32	16.3	3.8	2.1	1.9
UMAC-STD	52.9	12.3	3.8	1.9
UMAC16	14.0	2.7	1.2	1.0
UMAC-MMX	35.9	4.5	1.7	1.0

UMAC16, аналогично UMAC32, использует три семейства хеш-функций: NH, RPHash и IPHASH.

NH хеширует блоки по 2КБт, производя хеш-результат в 32 бита, что соответствует коэффициенту сжатия 512. Вероятность коллизии не превышает 2^{-15}. Результат передается к RP уровню хеширования, который вычисляет выходную строку длины 128 бит. RP семейство использует три простых поля с 32 битным, с 64 битным и с 128 битным простыми модулями, соответственно. Длина сообщения ограничена максимальным значением 2^{64} бит, и доказано, что этот слой добавляет около 2^{-19} к вероятности коллизии. Если аутентифицируемое сообщение короткое, то RP слой пропускается для оптимизации скорости вычисления. IP уровень сворачивает входную последовательность с 128 битами к выходной с 16 битами, поддерживая вероятность коллизии почти 2^{-15}. Конструкция с тремя уровнями повторяется неоднократно, с независимыми ключами, увеличивая длину аутентификационного тега и уменьшая шанс подделки MAC кода. Заданное по умолчанию число – четыре раза, и конкатенация 16 битовых слов вычисляет MAC код с 64 битами с вероятностью подделки 2^{-60}. Главное отличие в UHASH32 состоит в том, что используются слова с 32 битами и вычисления повторяются по схеме с тремя слоями только дважды (значение по умолчанию).

Преимущество по сравнению с предыдущей UMAC (1999) версией состоит в том, что использование криптографического примитива (проект NESSIE рекомендует AES шифр) минимизировано, что в

результате оказывается более эффективным на коротких сообщениях, и это дает дополнительную гибкость для верификации. Можно выбирать, сколько из параллельных вычислений использовать в MAC коде, учитывая временные затраты и уровень гарантий.

IV Анализ безопасности UMAC алгоритма

UMAC код аутентификации сообщения основан на семействах универсальных хэш-функций и предлагает доказуемую безопасность в том смысле, что имеются теоретические границы коллизии для хешируемой части при вычислении MAC кода, так что безопасность в конечном счете зависит от криптографического примитива, используемого для зашифрования показателя новизны.

Примитив, заявленный в спецификации UMAC кода - AES (Rijndael) блочный шифр, является достаточно надежным. Имеется дополнительное преимущество, которое состоит в том, что шифрование выполняется на коротком показателе новизны. Не были найдены недостатки в доказательстве безопасности UMAC [2]. Для уровня, использующего NH универсальное семейство хэш-функции, первое положение доказательства безопасности состоит в том, что NH является 2^{-W}– почти универсальным. Это означает, что вероятность коллизии не больше, чем 2^{-W} для строк равной длины и слов длиной в w бит. Это соответствует использованию NH на одном блоке установленной длины сообщения. Второе положение в доказательстве безопасности позволяет расширить этот результат на NH алгоритм, работающий на любом числе строк, в соответствии с параметрами в спецификации UMAC. Вероятность коллизии после NH уровня в UHASH16 схеме равна 2^{-15} , что больше чем 2^{-16} из-за знаковой арифметики.

UHASH16 и UHASH32 имеют два дополнительных уровня хеширования, использующие RP и IP универсальные семейства хэш-функций, и они повторяют схему с тремя слоями четыре раза и соответственно два раза. Доказано, что семейство UHASH16 является 4-связанным ($2^{-15}+2^{-18}+2^{-28}$) почти универсальным, а семейство UHASH32 является 2-связанным ($2^{-31}+2^{-33}$) почти универсальным, что гарантирует вероятность коллизии 2^{-60}.

V Выводы

UMAC (1999, 2000) реализации имеют самую высокую скорость вычислений для MAC примитивов, представленных проекту NESSIE. Алгоритм UMAC основан на применения композиционной схемы с многократным универсальным хешированием и криптографическим вычислением тега аутентификации, гарантирует вероятность коллизии 2^{-60} и по итоговым показателям является одним из лучших.

*Литература: **1**. Black, J., Halevi, S., Krawczyk, H., Krovetz, T., and Rogaway, P. UMAC: Fast and secure message authentication. In Advances in Cryptology 'CRYPTO '99 (1999), vol. 1666 of Lecture Notes in Computer Science, Springer-Verlag, pp. 216-233. **2.** T. Krovetz, J. Black, S. Halevi, A. Hevia, H. Krawczyk, and P. Rogaway, "UMAC." Primitive submitted to NESSIE, Sept. 2000. [p. 5, 157, 160] **3.** B. Preneel, "Cryptographic primitives for information authentication state of the art." in State of the Art in Applied Cryptography (B. Preneel and V. Rijmen, eds.), no. 1528 in Lecture Notes in Computer Science, pp. 50-105, Springer-Verlag, 1998. **4.** Halevi, S., and Krawczyk, H. MMH: Software message authentication in the Gbit/second rates. In Proceedings of the 4th Workshop on Fast Software Encryption (1997), vol. 1267, Springer-Verlag, pp. 172-189. **5.** Carter, L., and Wegman, M. Universal classes of hash functions. J. of Computerand System Sciences 18 (1979), 143-154. **6.** Krovetz, T., and Rogaway, P. Variationally universal hashing. In preparation, 2000. **7.** Rogaway, P. Bucket hashing and its application to fast message authentication. In Advances in Cryptology 'CRYPTO '95 (1995), vol. 963 of Lecture Notes in Computer Science, Springer-Verlag, pp. 313-328. **8.** Bosselaers, A., Govaerts, R., and Vandewalle, J. Fast hashing on the Pentium. In Advances in Cryptology 'CRYPTO '96 (1996), vol. 1109 of Lecture Notes in Computer Science, Springer-Verlag, pp. 298-312. Updated timing at http://www.esat.kuleuven.ac.be/ bosselae/fast.html.*

УДК 004.056 : 004.424.47

ПІДВИЩЕННЯ СТІЙКОСТІ ФУНКЦІЙ ХЕШУВАННЯ В СХЕМАХ АВТЕНТИФІКАЦІЇ

Олександр Іщенко, Юрій Яремчук

Вінницький національний технічний університет

Анотація: **Розглянуто проблеми, пов'язані з використанням функцій хешування в схемах автентифікації. Запропоновано метод усунення атаки на подовження/скорочення повідомлення.**