Network Working Group
Request for Comments: 2104
Category: Informational

H. Krawczyk
IBM
M. Bellare
UCSD
R. Canetti
IBM

February 1997

Проверка подлинности сообщений HMAC

HMAC: Keyed-Hashing for Message Authentication

Статус документа

Этот документ содержит информацию для сообщества Internet и не задает каких-либо стандартов Internet. Документ можно распространять без ограничений.

Тезисы

Этот документ описывает механизм HMAC для проверки подлинности (аутентификации) сообщений с использованием криптографических функций хэширования. HMAC может применяться с любой итеративной криптографической хэш-функцией (например, MD5, SHA-1) в комбинации с разделяемым секретным ключом. Криптостойкость HMAC зависит от свойств используемой хэш-функции.

1. Введение

Обеспечение способа проверки целостности информации, передаваемой или сохраняемой в ненадежной среде, является настоятельной необходимостью в мире открытых вычислений и коммуникаций. Механизмы, обеспечивающие проверку целостности на базе секретного ключа обычно называют кодами MAC[1]. Обычно такие коды применяются между двумя сторонами, знающими общий секретный ключ для валиадации данных, передаваемых между этими сторонами. В этом документе представлен такой механизм MAC, основанный на криптографических хэш-функциях. Этот механизм, названный HMAC, основан на работе [BCK1], в которой представлена конструкция и выполнен криптографический анализ. Эта работа упомянута по причине наличия в ней подробного обоснования и анализа защищенности HMAC, а также сравнение с другими методами хэширования на основе ключей.

HMAC может применяться с любой итеративной криптографической хэш-функцией. Примерами таких функций могут служить MD5 и SHA-1. HMAC также использует секретный ключ для расчета и проверки значений, используемых при аутентификации сообщения. Основные цели предлагаемой конструкции перечислены ниже.

- применение без изменения механизма разных доступных хэш-функций (в частности, хэш-функции с открытым кодом и широким распространением);
- сохранение производительности исходной хэш-функции без существенного снижения;
- простота использования и обслуживания ключей;
- понятный криптоанализ стойкости механизма аутентификации на основе разумных предположений об используемой хэш-функции;
- простота смены хэш-функции в случаях создания или необходимости применения боле быстрой или более защищенной функции.

Этот документ служит спецификацией механизма HMAC использующего обычную криптографическую хэш-функцию (H). Для конкретной реализации HMAC нужно выбрать определенную функцию хэширования. Современные претенденты на эту роль включают SHA-1 [SHA], MD5 [MD5], RIPEMD-128/160 [RIPEMD]. Такие реализации HMAC будут обозначаться HMAC-SHA1, HMAC-MD5, HMAC-RIPEMD и т. п.

> Примечание. На момент подготовки этого документа MD5 и SHA-1 были наиболее распространенными криптографическими хэш-функциями. Недавно была продемонстрирована уязвимость MD5 к атакам с поиском коллизий [Dobb]. Эта атака и другие известные уязвимости MD5 не отменяют применения MD5 для HMAC, указанного в этом документе (см. [Dobb]), однако функция SHA-1 представляется более криптостойкой. На данный момент MD5 может рассматриваться для использования в HMAC для приложений, где важна непревзойденная производительность MD5. В любом случае разработчикам и пользователям следует принимать во внимание возможности криптоаналитических разработок, касающихся применяемых хэш-функций и связанного с этим риска необходимости замены используемой функции (см. обсуждение вопросов безопасности в разделе 6).

[1] Message authentication code — код аутентификации сообщения.

2. Определение HMAC

Определение HMAC требует криптографической хэш функции, которую мы обозначим H, и секретный ключ K. Предполагается, что криптографическая функция H хэширует данные путем итеративного применения базовой функции сжатия к блокам данных. Обозначим размер такого блока в байта буквой B (B=для всех упомянутых выше примеров хэш-функций), а L — выходной размер хэш функции в байтах (L=16 для MD5, L=20 для SHA-1). Ключ аутентификации K может иметь любой размер вплоть до размера блока хэш-функции (B). Приложения, использующие ключи размером больше B, будут сначала хэшировать такой ключ с помощью H, а затем использовать результат размера L в формате строки байтов, как реальный ключ для HMAC. Во всех случаях рекомендуется использовать ключи K размером не менее L байтов (выходной размер хэш-функции). Дополнительные сведения о ключах приведены в разделе 3.

Мы определяем две фиксированных строки ipad и opad (i — вход, o — выход) следующим образом:

```
ipad = повторенный B раз байт 0x36
opad = повторенный B раз байт 0x5C
```

Для расчета значения HMAC для строки text выполняется операция

```
H(K XOR opad, H(K XOR ipad, text))
```

Ниже приведен подробный список выполняемых операций.

(1) в конце K добавляются нули до создания строки размером B байтов (например, если K имеет размер 20 байтов, а B=64, в конце строки K добавляется 44 байта со значением 0x00);

(2) выполняется операция XOR (побитовое «исключающее-ИЛИ) для строки размера B, полученной на этапе (1), и ipad;

(3) к полученной на этапе (2) строке размера B в конце добавляется поток входной информации text;

(4) к полученному на этапе (3) результату применяется функция H;

(5) выполняется операция XOR (побитовое «исключающее-ИЛИ) для строки размера B, полученной на этапе (1), и opad;

(6) результат H, полученные на этапе (4) добавляется в конец строки размера B, полученной на этапе (5);

(7) к полученному на этапе (6) результату применяется функция H.

Для иллюстрации этого процесса в Приложении дан пример кода, использующего хэш-функцию MD5.

3. Ключи

Ключи для HMAC могут иметь любой размер (для ключей размером больше B байтов предварительно применяется функция H). Однако использовать ключи размером меньше L байтов настоятельно не рекомендуется, поскольку это будет существенно снижать криптостойкость функции. Ключи размером более L подходят и увеличение размера ведет к существенному росту криптостойкости функции (более длинные ключи желательно применять в тех случаях, когда уровень случайности ключа представляется низким).

Ключи должны выбираться с использованием случайных значений (или применением криптографически сильного генератора псевдослучайных чисел со случайной затравкой) и периодически обновляться. (Современные атаки не определяют конкретных требований к срокам смена ключей, поскольку такие атаки практически не осуществимы. Однако периодическая смена ключей является основой практического обеспечения безопасности и помогает преодолеть возможную слабость функций и ключей, а также снизить ущерб в случае раскрытия ключа.)

4. Примечание для разработчиков

Механизм HMAC определен таким образом, что не требуется вносить изменений в код используемой хэш-функции H. В частности, используется функция H с предопределенным начальным вектором инициализации IV (фиксированное значение, задаваемое каждой итеративной хэш-функцией для инициализации ее функции сжатия). Однако при необходимости можно увеличить эффективность механизма путем (возможного) изменения кода H с целью поддержки изменяемых значений IV.

Идея состоит в том, что промежуточные результаты функции сжатия B-байтовых блоков (K XOR ipad) и (K XOR opad) можно заранее рассчитать только один раз в момент генерации ключа K или до первого его использования. Эти промежуточные результаты сохраняются и позднее используются для инициализации IV функции H каждый раз, когда требуется аутентификация сообщения. Этот метод позволяет при аутентификации каждого сообщения опустить применение функции сжатия из состава H к двум B-байтовым блокам (т. е., K XOR ipad и K XOR opad). Такая экономия может оказаться существенной при аутентификации небольших объемов данных. Подчеркнем, что промежуточные значения требуют такого же отношения и защиты, как секретные ключи.

Выбор для реализации HMAC описанного выше метода определяется разработчиком и не влияет на интероперабельность.

5. Отсечка вывода

Общепринятой практикой применения кодов аутентификации сообщений является отсечка вывода MAC с использованием лишь части битов (например, [MM, ANSI]). Preneel и van Oorschot [Pv] показали некоторые аналитические преимущества отсечки выхода основанных на хэшировании функций MAC. Результаты в этом смысле не являются абсолютными в плане общего повышение уровня защиты в результате отсечки. Имеются как позитивные (снижение объема информации, доступной для атакующего), так и негативные (атакующему нужно угадать меньше битов) эффекты. Приложения HMAC могут выбрать отсечку результата HMAC путем вывода лишь t старших (левых) битов HMAC (расчет выполняется обычным путем, как описано в разделе 2, а кнонечный результат отсекается до размера t битов). Рекомендуется делать размер вывода t не меньше половины выходного размера хэш-функции (атаки

birthday[1]) и не меньше 80 битов (приемлемая нижняя граница числа битов, которые потребуется угадать злоумышленнику). Предлагается обозначать реализации HMAC с хэш-функцией H и отсечкой вывода t, как HMAC-H-t. Например, HMAC-SHA1-80 будет указывать реализацию HMAC с использованием хэш-функции SHA-1 и отсечкой вывода до 80 битов (если параметр t не указан — например, HMAC-MD5 — предполагается вывод без отсечки).

6. Безопасность

Защищенность представленного здесь механизма аутентификации сообщений зависит от криптографических свойств хэш-функции H - устойчивости к поиску коллизий (для случая, когда начальное значение случайно и хранится в секрете, а вывод функции не доступен атакующему в явном виде) и возможностей аутентификации сообщений функции сжатия в составе H в случаях применения к одному блоку (в HMAC эти блоки частично не известны атакующему, поскольку они содержат результат внутреннего расчета H и, в частности, не могут быть полностью выбраны атакующим).

Эти свойства (а реально более строгие) обычно предполагаются у хэш-функций, используемых для HMAC. В частности, хэш-функция, не обладающая указанными выше свойствами, не подойдет для большинства (возможно, для всех) криптографических приложений, включая дополнительные схемы аутентификации сообщений на основе таких функций (полный анализ и обоснование функции HMAC можно найти в [BCK1]).

С учетом ограниченности данных для оценки криптостойкости рассматриваемых в качестве кандидатов хэш-функций, важно отметить два свойства HMAC и применения этого механизма для аутентификации сообщений.

1. Конструкция механизм не зависит от деталей конкретной хэш-функции H и эту функцию можно заменить любой другой защищенной (итеративной) криптографической хэш-функцией.
2. Аутентификация сообщений, в отличие от их шифрования, имеет переходный эффект. Публикация сведений о взломе схемы проверки подлинности сообщений ведет к необходимости замены этой схемы, но не оказывает никакого влияния на выполненную ранее аутентификацию. А вот зашифрованная сегодня информация может быть расшифрована в будущем при взломе использованного алгоритма шифрования.

Наиболее сильная атака на HMAC основана на частоте коллизий для хэш-функции H (birthday attack) [PV,BCK2] и практически не реализуема для сколь-нибудь серьезных хэш-функций.

Например, для хэш-функции типа MD5 с выходным размером L=16 байтов (128 битов) атакующему потребуется собрать корректные теги аутентификации сообщений (с тем же секретным ключом K) примерно для 2^{64} известных нешифрованных текстов. Это потребует обработки не менее 2^{64} с помощью H, что на практике не представляется реальным (при размере блока 64 байта это потребует 250 000 лет непрерывных передачи по каналу производительностью 1 Гбит/с без смены ключа K в течение всего времени). Такая атака станет реальной только при обнаружении серьезных проблем с коллизиями функции H (например, обнаруженной после 2^{30} сообщений). Такое событие потребует незамедлительной замены функции H (гораздо сильнее это воздействовало бы на традиционные применения функции H в контексте цифровых подписей, сертификатов открытых ключей и т. п.).

> Примечание. Описанная атака сильно отличается от обычных атак с поиском коллизий на криптографические функции без секретного ключа, где можно использовать 2^{64} параллельных (!) операций для поиска коллизий. Эта атака уже представляется возможной на практике [VW], тогда как birthday-атаки на HMAC совершенно не реальны на практике (в приведенном выше примере использование функции с выходом 160 битов потребует уже 2^{80} блоков вместо 2^{64}).

Корректная реализация описанной выше конструкции, выбор случайных (или криптографических псевдослучайных) ключей, защищенный механизм обмена ключами, частая смена ключей и обеспечение их секретности являются важными компонентами защиты механизма контроля целостности, обеспечиваемого HMAC.

Приложение — пример кода

Для наглядности ниже представлен образец кода, реализующего HMAC-MD5 и некоторые тестовые векторы (пример основан на коде MD5, описанном в [MD5]).

```
/*
** Функция - hmac_md5
*/

void
hmac_md5(text, text_len, key, key_len, digest)
unsigned char*  text;                /* указатель на поток данных */
int             text_len;            /* размер потока данных */
unsigned char*  key;                 /* указатель на ключ аутентификации */
int             key_len;             /* размер ключа аутентификации */
caddr_t         digest;              /* дайджест для заполнения (результат функции) */

{
        MD5_CTX context;
        unsigned char k_ipad[65];    /* внутреннее заполнение -  key XOR ipad */
        unsigned char k_opad[65];    /* внешнее заполнение -  key XOR opad */
        unsigned char tk[16];
        int i;
        /* сброс ключа размером более 64 байтов в key=MD5(key) */
        if (key_len > 64) {
                MD5_CTX      tctx;

                MD5Init(&tctx);
                MD5Update(&tctx, key, key_len);
```

[1]См. описание такие атак в Википедии. *Прим. перев.*

```
                MD5Final(tk, &tctx);

                key = tk;
                key_len = 16;
        }

        /* преобразование HMAC_MD5 имеет вид:
         *
         * MD5(K XOR opad, MD5(K XOR ipad, text))
         *
         * где K — ключ размером n байтов
         * ipad — байт 0x36, повторенный 64 раза
         * opad — байт 0x5c, повторенный 64 раза
         * text — защищаемые данные
         */

        /* начинаем с сохранения ключа в заполнении */
        bzero( k_ipad, sizeof k_ipad);
        bzero( k_opad, sizeof k_opad);
        bcopy( key, k_ipad, key_len);
        bcopy( key, k_opad, key_len);

        /* выполняем операцию XOR для значений ipad и opad */
        for (i=0; i<64; i++) {
                k_ipad[i] ^= 0x36;
                k_opad[i] ^= 0x5c;
        }
        /* расчет внутреннего значения MD5 */
        MD5Init(&context);                   /* инициализация содержимого для 1-го прохода */
        MD5Update(&context, k_ipad, 64)      /* начало внутреннего заполнения */
        MD5Update(&context, text, text_len); /* текст дейтаграммы */
        MD5Final(digest, &context);          /* завершение  1-го прохода */
        /* расчет внешнего значения MD5 */
        MD5Init(&context);                   /* инициализация содержимого для 2-го прохода */
        MD5Update(&context, k_opad, 64);     /* начало внешнего заполнения */
        MD5Update(&context, digest, 16);     /* используем результат 1-го хэширования */
        MD5Final(digest, &context);          /* завершение  2-го прохода */
}
```

Тестовые векторы (\0 в конце символьной строки не указан).

```
  key =         0x0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b0b
  key_len =     16 байтов
  data =        "Hi There"
  data_len =    8  байтов
  digest =      0x9294727a3638bb1c13f48ef8158bfc9d

  key =         "Jefe"
  data =        "what do ya want for nothing?"
  data_len =    28 байтов
  digest =      0x750c783e6ab0b503eaa86e310a5db738

  key =         0xAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
  key_len       16 байтов
  data =        0xDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD
                DDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD
  data_len =    50 байтов
  digest =      0x56be34521d144c88dbb8c733f0e8b3f6
```

Благодарности

Pau-Chen Cheng, Jeff Kraemer и Michael Oehler представили полезные комментарии к предварительным вариантам документа и выполнили первые проверки интероперабельности для этой спецификации. Jeff и Pau-Chen любезно предоставили образец кода и тестовые векторы, приведенные в Приложении. Burt Kaliski, Bart Preneel, Matt Robshaw, Adi Shamir и Paul van Oorschot представили полезные комментарии и предложения в процессе обсуждения конструкции HMAC.

Литература

[ANSI] ANSI X9.9, "American National Standard for Financial Institution Message Authentication (Wholesale)," American Bankers Association, 1981. Revised 1986.

[Atk] Atkinson, R., "IP Authentication Header", RFC 1826, August 1995.

[BCK1] M. Bellare, R. Canetti, and H. Krawczyk, "Keyed Hash Functions and Message Authentication", Proceedings of Crypto'96, LNCS 1109, pp. 1-15. (http://www.research.ibm.com/security/keyed-md5.html)

[BCK2] M. Bellare, R. Canetti, and H. Krawczyk, "Pseudorandom Functions Revisited: The Cascade Construction", Proceedings of FOCS'96.

[Dobb] H. Dobbertin, "The Status of MD5 After a Recent Attack", RSA Labs' CryptoBytes, Vol. 2 No. 2, Summer 1996. http://www.rsa.com/rsalabs/pubs/cryptobytes.html

[PV] B. Preneel and P. van Oorschot, "Building fast MACs from hash functions", Advances in Cryptology -- CRYPTO'95 Proceedings, Lecture Notes in Computer Science, Springer-Verlag Vol.963, 1995, pp. 1-14.

[MD5] Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, April 1992.

[MM] Meyer, S. and Matyas, S.M., Cryptography, New York Wiley, 1982.

[RIPEMD] H. Dobbertin, A. Bosselaers, and B. Preneel, "RIPEMD-160: A strengthened version of RIPEMD", Fast Software Encryption, LNCS Vol 1039, pp. 71-82. ftp://ftp.esat.kuleuven.ac.be/pub/COSIC/bosselae/ripemd/.

[SHA] NIST, FIPS PUB 180-1: Secure Hash Standard, April 1995.

[Tsu] G. Tsudik, "Message authentication with one-way hash functions", In Proceedings of Infocom'92, May 1992. (Also in "Access Control and Policy Enforcement in Internetworks", Ph.D. Dissertation, Computer Science Department, University of Southern California, April 1991.)

[VW] P. van Oorschot and M. Wiener, "Parallel Collision Search with Applications to Hash Functions and Discrete Logarithms", Proceedings of the 2nd ACM Conf. Computer and Communications Security, Fairfax, VA, November 1994.

Адреса авторов

Hugo Krawczyk

IBM T.J. Watson Research Center

P.O.Box 704

Yorktown Heights, NY 10598

EMail: hugo@watson.ibm.com

Mihir Bellare

Dept of Computer Science and Engineering

Mail Code 0114

University of California at San Diego

9500 Gilman Drive

La Jolla, CA 92093

EMail: mihir@cs.ucsd.edu

Ran Canetti

IBM T.J. Watson Research Center

P.O.Box 704

Yorktown Heights, NY 10598

EMail: canetti@watson.ibm.com

Перевод на русский язык

Николай Малых

nmalykh@gmail.com