Appears in *RSA Laboratories' CryptoBytes*, Vol. 2, No. 1, Spring 1996.

Message Authentication using Hash Functions— The HMAC Construction

Mihir Bellare[*] Ran Canetti[†] Hugo Krawczyk[‡]

There has recently been a lot of interest in the subject of authenticating information using cryptographic hash functions like MD5 and SHA, particularly for Internet security protocols. We report on our HMAC construction [1] which seems to be gaining acceptance as a solution.

Introduction

Two parties communicating across an insecure channel need a method by which any attempt to modify the information sent by one to the other, or fake its origin, is detected. Most commonly such a mechanism is based on a shared key between the parties, and in this setting is usually called a MAC, or Message Authentication Code. (Other terms include Integrity Check Value or Cryptographic Checksum). The sender appends to the data D an *authentication tag* computed as a function of the data and the shared key. At reception, the receiver recomputes the authentication tag on the received message using the shared key, and accepts the data as valid only if this value matches the tag attached to the received message.

The most common approach is to construct MACs from block ciphers like DES. Of such constructions the most popular is the CBC MAC. (Its security is analyzed in [4, 12]). More recently, however, people have suggested that MACs might be constructed from cryptographic hash functions like MD5 and SHA. There are several good reasons to attempt this: In software these hash functions are significantly faster than DES; library code is widely and freely available; and there are no export restrictions on hash functions.

Thus people seem agreed that hash function based constructions of MACs are worth having. The more difficult question is how best to do it. Hash functions were not originally designed for message authentication. (One of many difficulties is that they are not even keyed primitives, i.e., do not accommodate naturally the notion of a secret key). Several constructions were proposed prior to HMAC, but they lacked a convincing security analysis.

The HMAC construction is intended to fill this gap. It has a performance which is essentially that of the underlying hash function. It uses the hash function in a black box way so that it can be implemented with available code, and also replacement of the hash function is easy should need of such a replacement arise due to security or performance reasons. Its main advantage, however, is that it can be proven secure provided the underlying hash function has some reasonable cryptographic strengths. The security features can be summarized like this: if HMAC fails to be a secure MAC, it means there are sufficient weaknesses in the underlying hash function that it needs to be dropped not only from this particular usage but also from a wide range of other popular usages to which it is now subject.

[*]Department of Computer Science & Engineering, Mail Code 0114, University of California at San Diego, 9500 Gilman Drive, La Jolla, CA 92093. Email: `mihir@cs.ucsd.edu`. `http://www-cse.ucsd.edu/users/mihir`.

[†]Laboratory for Computer Science, 545 Technology Square, Cambridge, MA 02139. Email: `canetti@theory.lcs.mit.edu`. Supported by a post-doctoral grant from the Rothschild Foundation.

[‡]IBM T.J. Watson Research Center, PO Box 704, Yorktown Heights, New York 10598. Email: `hugo@watson.ibm.com`.

Several articles in the literature survey existing constructions, their properties, and some of their weaknesses, so we will not try to do this again here. In particular the reader is referred to Tsudik [17], who provides one of the earliest works on the subject; Kaliski and Robshaw who, in the first CryptoBytes [8], compare various possible constructions; updates appearing in succeeding issues of CryptoBytes; and Preneel and van Oorschot [12, 13], who present a detailed description of the effect of birthday attacks on "iterated constructions" and also a new construction called MDx-MAC.

We now move on to discuss the HMAC construction, status, and rationale. For a complete description, implementation guidelines, and detailed analysis we refer the reader to [1, 9].

HMAC

Let H be the hash function. For simplicity of description we may assume H to be MD5 or SHA-1; however the construction and analysis can be applied to other functions as well (see below). H takes inputs of any length and produces l-bit output ($l = 128$ for MD5 and $l = 160$ for SHA-1). Let Text denote the data to which the MAC function is to be applied and let K be the message authentication secret key shared by the two parties. (It should not be larger than 64 bytes, the size of a hashing block, and, if shorter, zeros are appended to bring its length to exactly 64 bytes.) We further define two fixed and different 64 byte strings ipad and opad as follows (the "i" and "o" are mnemonics for inner and outer):

ipad = the byte 0x36 repeated 64 times

opad = the byte 0x5C repeated 64 times.

The function HMAC takes the key K and Text, and produces $\mathrm{HMAC}_K(\mathsf{Text})$ =

$$H(K \oplus \mathsf{opad}, H(K \oplus \mathsf{ipad}, \mathsf{Text})) .$$

Namely,

(1) Append zeros to the end of K to create a 64 byte string

(2) XOR (bitwise exclusive-OR) the 64 byte string computed in step (1) with ipad

(3) Append the data stream Text to the 64 byte string resulting from step (2)

(4) Apply H to the stream generated in step (3)

(5) XOR (bitwise exclusive-OR) the 64 byte string computed in step (1) with opad

(6) Append the H result from step (4) to the 64 byte string resulting from step (5)

(7) Apply H to the stream generated in step (6) and output the result

The recommended length of the key is at least l bits. A longer key does not add significantly to the security of the function, although it may be advisable if the randomness of the key is considered weak.

HMAC optionally allows truncation of the final output say to 80 bits.

As a result we get a simple and efficient construction. The overall cost for authenticating a stream Text is close to that of hashing that stream, especially as Text gets large. Furthermore, the hashing of the padded keys can be precomputed for even improved efficiency.

Note HMAC uses the hash function H as a black box. No modifications to the code for H are required to implement HMAC. This makes it easy to use library code for H, and also makes it easy to replace a particular hash function, such as MD5, with another, such as SHA, should the need to do this arise.

HMAC was recently chosen as the mandatory-to-implement authentication transform for the Internet security protocols being designed by the IPSEC working group of the IETF (it replaces as a mandatory transform the one described in [10]). For this purpose HMAC is described in the Internet Draft [9], and in an upcoming RFC. Other Internet protocols are adopting HMAC as well (e.g., s-http [14], SSL [7]).

The rationale

We now briefly explain some of the rationale used in [1] to justify the HMAC construction.

As we indicated above, hash functions were not originally designed to be used for message authentication. In particular they are not keyed primitives, and it is not clear how best to "key" them. Thus, one ought to be quite careful in using hash functions to build MACs.

The standard approach to security evaluation is to look for attacks on a candidate MAC construction. When practical attacks can be found, their effect is certainly conclusive: the construction must be dropped. The difficulty is when attacks are not yet found. Should one adopt the construction? Not clear, because attacks might be found in the future.

The maxim that guided the HMAC construction was that *an absence of attacks today does not im-*

ply security for the future. A better way must be found to justify the security of a construction before adopting it.

You can't make good wine from bad grapes: if no strengths are assumed of the hash function, we can't hope to justify any construction based on it. Accordingly it is appropriate to make some assumptions on the strength of the hash function.

A well justified MAC construction, in our view, is one under which the security of the MAC can be related as closely as possible to the (assumed) security properties of the underlying hash function.

The assumptions on the security of the hash function should not be too strong, since after all not enough confidence has been gathered in current candidates (like MD5 or SHA). In fact, the weaker the assumed security properties of the hash function, the stronger the resultant MAC construction is.

We make assumptions that reflect the more standard existing usages of the hash function. The properties we require are mainly collision-freeness and some limited "unpredictability." What is shown is that if the hash function function has these properties the MAC is secure; the *only* way the MAC could fail is if the hash function fails.

In fact the assumptions we make are in many ways *weaker* than standard ones. In particular we require only a weak form of collision-resistance. Thus it is possible that H is broken as a hash function (for example collisions are found) and yet HMAC based on H survives.

A closer look

Security of the MAC means security against forgery. The MAC is considered broken if an attacker, not having the key K, can find some text Text together with its correct MAC value $\mathrm{HMAC}_K(\mathsf{Text})$. The attacker is assumed able to gather some number of example pair of texts and their valid MACs by observing the traffic between the sender and the receiver. Indeed the adversary is even allowed a chosen message attack under which she can influence the choice of messages for which the sender computes MACs. Following [4, 3] we quantify security in terms of the probability of successful forgery under such attacks.

The analysis of [1] applies to hash functions of the iterated type, a class that includes MD5 and SHA, and consists of hash functions built by iterating applications of a compression function f according to the procedure of Merkle [11] and Damgård [5]. (In this construction a l-bit initial variable IV is fixed, and the output of H on text x is computed by breaking x into 512 bit blocks and hashing in stages using f, in a simple way that the reader can find described in many places, e.g. [8].)

Roughly what [1] say is that an attacker who can forge the HMAC function can, with the same effort (time and collected information), break the underlying hash function in one of the following ways:

(1) The attacker finds collisions in the hash function even when the IV is random and secret, or

(2) The attacker is able to compute an output of the compression function even with an IV that is random, secret and unknown to the attacker. (That is, the attacker is successful in forging with respect to the application of the compression function secretly keyed and viewed as a MAC on fixed length messages.)

The feasibility of any of these attacks would contradict some of our basic assumptions about the cryptographic strength of these hash functions. Success in the first of the above attacks means success in finding collisions, the prevention of which is the main design goal of cryptographic hash functions, and thus can be assumed hard to do. But in fact, even more is true: success in the first attack above is even *harder* than finding collisions in the hash function, because collisions when the IV is secret (as is the case here) is far more difficult than finding collisions in the plain (fixed IV) hash function. This is because the former requires interaction with the legitimate user of the function (in order to generate pairs of input/outputs from the function), and disallows the parallelism of traditional birthday attacks. Thus, even if the hash function is not collision-free in the traditional sense, our schemes could be secure.

Some "randomness" of hash functions is assumed in their usage for key generation and as pseudorandom generators. (For example the designers of SHA suggested that SHA be used for this purpose [6].) Randomness of the function is also used as a design methodology towards achieving collision-resistance. The success of the second attack above would imply that these randomness properties of the hash functions are very poor.

The analyses in [1] used to establish the above are *exact* (no asymptotics involved), consider *generic* rather than particular attacks, and establish a *tight* relationship between the securities.

Resistance to known attacks

As shown in [12, 2], birthday attacks, that are the basis to finding collisions in cryptographic hash functions, can be applied to attack also keyed MAC schemes based on iterated functions (including also CBC-MAC, and other schemes). These attacks apply to most (or all) of the proposed hash-based constructions of MACs. In particular, they constitute the best known forgery attacks against the HMAC construction. Consideration of these attacks is important since they strongly improve on naive exhaustive search attacks. However, their practical relevance against these functions is negligible given the typical hash lengths like 128 or 160. Indeed, these attacks require the collection of the MAC value (for a given key) on about $2^{l/2}$ messages (where l is the length of the hash output). For values of $l \geq 128$ the attack becomes totally infeasible. In contrast to the birthday attack on key-less hash functions, the new attacks require interaction with the key owner to produce the MAC values on a huge number of messages, and then allow for no parallelization. For example, when using MD5 such an attack would require the authentication of 2^{64} blocks (or 2^{73} bits) of data using the same key. On a 1 Gbit/sec communication link, one would need 250,000 years to process all the data required by such an attack. This is in sharp contrast to birthday attacks on key-less hash functions which allow for far more efficient and close-to-realistic attacks [18].

References

[1] M. Bellare, R. Canetti and H. Krawczyk. Keying hash functions for message authentication. *Advances in Cryptology – Crypto 96 Proceedings*, Lecture Notes in Computer Science Vol. ??, N. Koblitz ed., Springer-Verlag, 1996.

[2] M. Bellare, R. Canetti and H. Krawczyk. Pseudorandom functions revisited: The cascade construction. Manuscript, April 1996.

[3] M. Bellare, R. Guérin and P. Rogaway. XOR MACs: New methods for message authentication using finite pseudorandom functions. *Advances in Cryptology – Crypto 95 Proceedings*, Lecture Notes in Computer Science Vol. 963, D. Coppersmith ed., Springer-Verlag, 1995.

[4] M. Bellare, J. Kilian and P. Rogaway. The security of cipher block chaining. *Advances in Cryptology – Crypto 94 Proceedings*, Lecture Notes in Computer Science Vol. 839, Y. Desmedt ed., Springer-Verlag, 1994.

[5] I. Damgård. A design principle for hash functions. *Advances in Cryptology – Crypto 89 Proceedings*, Lecture Notes in Computer Science Vol. 435, G. Brassard ed., Springer-Verlag, 1989.

[6] National Institute for Standards and Technology. Digital Signature Standard (DSS). Federal Register, Vol. 56, No. 169, August, 1991

[7] A.O. Freier, P. Karlton, and P. C. Kocher. The SSL Protocol – Version 3.0. Internet draft draft-freier-ssl-version3-01.txt, March 1996.

[8] B. Kaliski and M. Robshaw. Message Authentication with MD5. *RSA Labs' CryptoBytes*, Vol. 1 No. 1, Spring 1995.

[9] H. Krawczyk, M. Bellare and R. Canetti. HMAC-MD5: Keyed-MD5 for Message Authentication. Internet draft draft-ietf-ipsec-hmac-md5-txt.00, March 1996.

[10] P. Metzger and W. Simpson. IP Authentication using Keyed MD5", IETF Network Working Group, RFC 1828, August 1995.

[11] R. Merkle. One way hash functions and DES. *Advances in Cryptology – Crypto 89 Proceedings*, Lecture Notes in Computer Science Vol. 435, G. Brassard ed., Springer-Verlag, 1989. (Based on unpublished paper from 1979 and his Ph. D thesis, Stanford, 1979).

[12] B. Preneel and P. van Oorschot. MD-x MAC and building fast MACs from hash functions. *Advances in Cryptology – Crypto 95 Proceedings*, Lecture Notes in Computer Science Vol. 963, D. Coppersmith ed., Springer-Verlag, 1995.

[13] B. Preneel and P. van Oorschot. On the security of two MAC algorithms. *Advances in Cryptology – Eurocrypt 96 Proceedings*, Lecture Notes in Computer Science Vol. ??, U. Maurer ed., Springer-Verlag, 1996.

[14] E. Rescorla and A. Schiffman. The Secure HyperText Transfer Protocol. Internet draft draft-ietf-wts-shttp-01.txt, February 1996.

[15] R. Rivest. The MD5 message-digest algorithm. IETF Network Working Group, RFC 1321, April 1992.

[16] FIPS 180-1. Secure Hash Standard. Federal Information Processing Standard (FIPS), Publication 180-1, National Institute of Standards and Technology, US Department of Commerce, Washington D.C., April 1995.

[17] G. Tsudik. Message authentication with one-way hash functions. *Proceedings of Infocom 92.*

[18] 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.