Message Authentication with MD5 [*]

Burt Kaliski and Matt Robshaw
RSA Laboratories
100 Marine Parkway, Suite 500
Redwood City, CA 94065 USA
burt@rsa.com matt@rsa.com

Message authentication is playing an important role in a variety of applications, especially those related to the Internet protocols and network management, where undetected manipulation of messages can have disastrous effects.

There is no shortage of good message authentication codes, beginning with DES-MAC, as defined in FIPS PUB 113 [7]. However, message authentication codes based on encryption functions such as DES, which were designed for hardware implementation, may be somewhat limited in performance for software, and there is also the question of U.S. export restrictions on encryption functions.

In standards efforts such as the Simple Network Management Protocol [5] and proposals for Internet Protocol security, a more practical solution seemed to be to base the authentication codes not on DES but on hash functions designed for fast software implementation which are widely available without restriction, such as the MD5 message-digest algorithm [9].

But how to do it? Hash functions are intended to resist inversion —finding a message with a given hash value — and collision — finding two messages with the same hash value. Message authentication codes, on the other hand, are intended to resist forgery — computing a message authentication code without knowledge of a secret key. Building a message authentication code on an encryption function thus seems a logical choice (and the security relationship has been recently settled — in work by Mihir Bellare, Joe Kilian and Phillip Rogaway [3]). Building one on a hash function, however, is not as simple, because the hash function doesn't have a key.

A hash function can provide message authentication in a most satisfying manner when combined with a digital signature algorithm, which does have a key. But typical digital

[*] Copyright © 1995 RSA Laboratories, a division of RSA Data Security, Inc. All rights reserved. RSA Data Security, Inc. part number 003-903060-100-000-000. Reprinted from *CryptoBytes*, 1(1), 5-8, Spring 1995.

signature schemes have some performance overhead, which while acceptable for the periodic setup of communications sessions, is often too large on a message-by-message basis. Thus, the focus is on message authentication based on a shared secret key, which is ideally integrated into the hash function in some manner.

As an illustration of the challenges, consider the "prefix" approach where the message authentication code is computed simply as the hash of the concatenation of the key and the message, where the key comes first and which we denote as MD5 (*k* . *m*).

MD5 follows the Damgård/Merkle [4,6] iterative structure, where the hash is computed by repeated application of a compression function to successive blocks of the message. (See Figure 1.) For MD5, the compression function takes two inputs — a 128-bit chaining value and a 512-bit message block — and produces as output a new 128-bit chaining value, which is input to the next iteration of the compression function. The message to be hashed is first padded to a multiple of 512 bits, and then divided into a sequence of 512-bit message blocks. Then the compression function is repeatedly applied, starting with an initial chaining value and the first message block, and continuing with each new chaining value and successive message blocks. After the last message block has been processed, the final chaining value is output as the hash of the message.

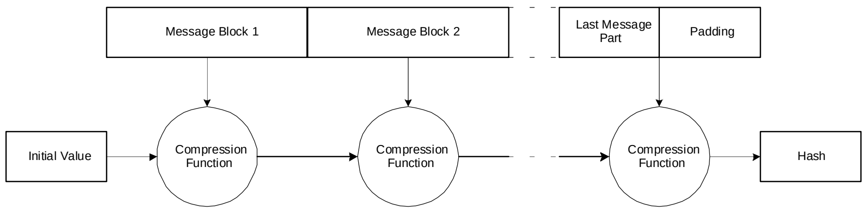

Figure 1. Damgård/Merkle iterative structure for hash functions.

Because of the iterative design, it is possible, from only the hash of a message, to compute the hash of longer messages that start with the initial message and include the padding required for the initial message to reach a multiple of 512 bits. Applying this to the prefix approach, it follows that from MD5 (*k* . *m*), one can compute MD5 (*k* . *m*') for any *m*' that starts with *m* . *p*, where *p* is the padding on *k* . *m*. In other words, from the message authentication code of *m*, one can forge the message authentication code of *m* . *p* . x for any x, without even knowing the key *k*, and without breaking MD5 in any sense. This is called a "message extension" or "padding" attack (see [10]).

Other hash functions with an iterative design, such as NIST's Secure Hash Algorithm [8], are also vulnerable to the message extension attack, and similar attacks can also be mounted on tree-structured designs.

(Note also that if only part of the hash were output, say only 64 bits, this attack would not be possible; however, this is not a completely satisfying solution because of other concerns raised below. In SNMP, the message extension attack is not a problem because messages are a fixed length. Another way to avoid the attack is to include an explicit length field at the beginning of the message.)

Because of the message extension attack on the prefix approach, the “suffix” approach, MD5 (m . k), would seem to be preferred. But another problem arises: the key may be vulnerable to cryptanalysis, depending on the properties of the compression function. This is because the message authentication code is a function of known values and the key, assuming the key is passed entirely to the last iteration of the compression function. (The known values are the next-to-last chaining value, which by assumption depends only on the message; the last part of the message; and the padding.)

An opponent who sees the message authentication codes for many messages thus sees the result of applying the compression function to many different known values and the same key, which may reveal information about the key. While our analysis suggests MD5's compression function is unlikely to reveal information about the key, other hash functions may not fare as well, and so we prefer a more robust design.

The prefix approach is also affected by these issues, but only when the message is very short and there is only a single iteration of the compression function.

Recommendations

In joint work with Mihir Bellare and Hugo Krawczyk of IBM, we have considered a number of approaches to message authentication with MD5, settling on three which we recommended to the Internet Protocol security (IPSEC) working group:

1. MD5 (k_1 . MD5 (k_2 . m)), where k_1 and k_2 are independent 128-bit keys

2. MD5 (k . p . m . k), where k is a 128-bit key and p is 384 bits of padding

3. MD5 (k . MD5 (k . m)), where k is a 128-bit key

The first and third approaches (see Figure 2) are similar, and solve the message extension attack on the prefix approach by the outer application of MD5, which conceals the chaining value which is needed for the attack. The outer MD5 also solves the concerns of cryptanalysis of the suffix approach, because the message authentication code is a function of the unknown secret key and other varying values, which are unknown. These approaches also approximate certain “provably secure” constructions developed by Bellare, Ran Canetti and Krawczyk [1].

(As a disclaimer, we can imagine hash functions for which this construction still doesn't solve the cryptanalytic problems because information from the inner application leaks to the outer one, but this seems more of a pathological case.)

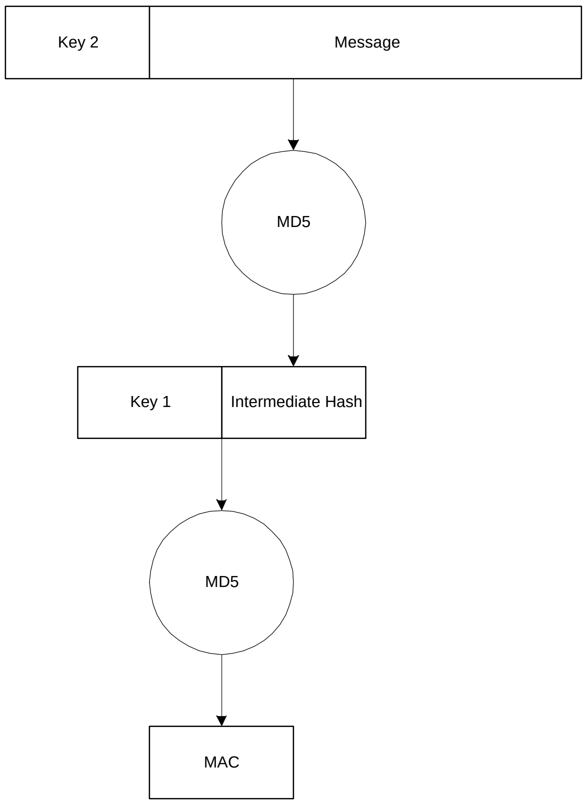

Figure 2. A recommended approach to message authentication with MD5. Here, the keys are each 128 bits long. They may be the same, although different keys are preferable.

The third approach may be more vulnerable to attack than the first since there is only one key and so any information revealed from the outer application of the hash function compromises security, but we know of no such attack on MD5.

Although the third approach has a shorter key size than the first, the first could also be implemented with a 128-bit key, without any apparent loss in security. For instance, the keys k_1 and k_2 could be derived from a single 128-bit key k as k_1 = MD5 (k . α) and k_2 = MD5 (k . β), where α and β are distinct constants.

The second approach (see Figure 3) is somewhat like triple encryption, where the first and third keys are the same (the second key is the message). The padding on the key at the beginning ensures that overall, there are at least two iterations of the compression function. Message extension in the prefix approach is solved by the key at the end, and the cryptanalysis of the suffix approach is solved by the key at the beginning. (Without the padding, very short messages might be vulnerable.)

It remains to be seen which, if any, of these three approaches is adopted.

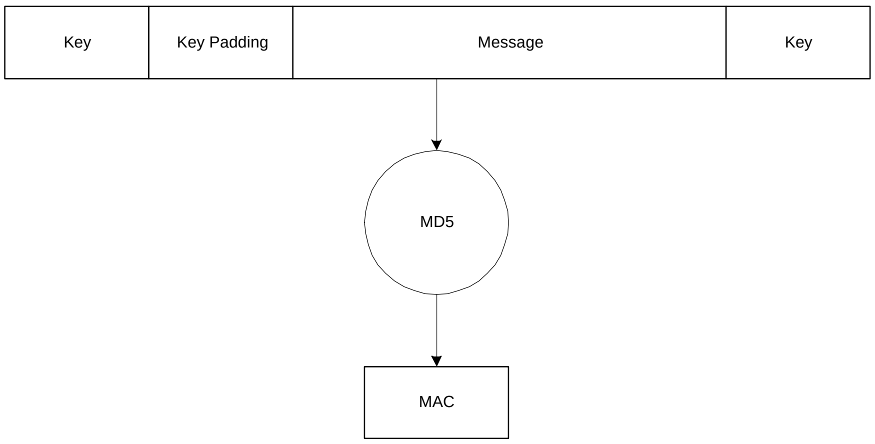

Figure 3. Another recommended approach to message authentication with MD5. Here, the key is 128 bits long and the padding is 384 bits long.

Interestingly, one of the approaches we had been previously promoting is not among the three we recommended to the IPSEC working group, based on our concerns about key exposure. That approach, MD5 (k . MD5 (m)), had the advantages that the inner MD5 is applied to the message in the familiar way — as a hash function — and the outer MD5 is applied to a fixed-length value, thereby avoiding message extension. However, since MD5 (m) is known, the door is open for possible cryptanalysis of the outer MD5 to recover the key k. While we once again do not have an attack that recovers the key, we felt as a general design principle that the key should be better concealed.

(Another concern with this approach, observed by some, is that collisions in MD5 — two messages with the same hash — result in collisions in the message authentication code. We do not consider this an intrinsic problem with this option, since MD5 is designed to resist collisions, at least to a certain level of difficulty. Nevertheless, we have no objection if the design of a message authentication code raises that level even further.)

Yet another approach we considered was MD5(MD5 (k . m)), which again applies MD5 in a familiar way. However, in terms of "provability" under certain assumptions it is less attractive than the three we recommended. (This does not mean that the approach is insecure, simply that the assumptions required for it to be secure are more complicated.)

As the IBM team has pointed out to us, all of the approaches are vulnerable to a chosen message attack involving about 2^{64} chosen messages. This general attack exploits the iterative structure of the message authentication code and applies to MACs based on encryption functions as well. The basic idea is that if two messages a_i . b and a_j . b have the same MAC, then it is possible that the "collision" occured before b was processed, so that for any c, a_i . c and a_j . c have the same MAC. Having found two messages a_i . b and a_j . b with the same MAC, the opponent asks for the MAC of a_i . c for some c, thereby obtaining (fraudulently) the MAC of a_j . c.

As chosen message attacks go, 2^{64} is quite a large number, and we know of no general way to extend the attack to known messages, except when the known messages are all the same length and end with the same suffix. Full details are given in [1].

Starting over

So far, our research has focused on adapting an existing hash function to message authentication, which is a practical solution, since MD5 is already trusted, and software for MD5 is widely available. For the long term, designing a message authentication code from scratch is perhaps a better solution.

Mihir Bellare, Roch Guérin and Phillip Rogaway [2] describe techniques for such message authentication that are "provably secure," under certain assumptions about underlying functions. Their techniques are also highly parallelizable, a feature that the iterative approach lacks by definition.

Bellare *et al*'s techniques assume the existence of a pseudorandom function, which takes two inputs, a key and a message block, and produces one output. By assumption, if the key input is fixed and unknown, it is difficult to distinguish the pseudorandom function on the message block from a truly random one in any reasonable amount of time. (This is similar to the idea that it is difficult to find collisions for a hash function — although it is possible because they exist, the amount of time required is large.)

The message authentication code is computed by combining, perhaps by bit-wise exclusive-or, the outputs of the pseudorandom function applied to the blocks of the message. To maintain the ordering of the different blocks, each block is tagged with its position in the message. A random block is also included for technical reasons.

Bellare *et al* show that if an opponent can forge message authentication codes, even with the opportunity to request message authentication codes on many different messages, then the opponent can also distinguish the pseudorandom function from a truly random one. Thus, under the assumption that it is difficult to distinguish the pseudorandom function from a truly random one, the message authentication code is secure.

The independent processing of the message blocks leads to the parallelizability of this approach.

It seems that many of the concerns about designing a message authentication code from a hash function are a consequence of the fact that the key is processed only once, or maybe twice. As a result, the key is isolated, and information about it can be obtained, or other parts of the message can be manipulated independent of the key. By contrast, in message authentication codes based on encryption functions, such as DES-MAC, the key is processed at every step. In Bellare *et al* 's techniques, the key is processed at every step.

We expect that MD5's compression function or a variant of it may be a suitable pseudorandom function for Bellare *et al*'s techniques, something which further research will determine.

References

[1] M. Bellare, R. Canetti and H. Krawczyk. *Keying MD5 - Message authentication via iterated pseudorandomness*. In preparation.

[2] Mihir Bellare, Roch Guérin and Phillip Rogaway. *XOR MACs: New methods for message authentication using block ciphers*. Accepted to Crypto’95.

[3] Mihir Bellare, Joe Kilian and Phillip Rogaway. The security of cipher block chaining. In Yvo G. Desmedt, editor, *Advances in Cryptology — CRYPTO ‘94*, volume 839 of *Lecture Notes in Computer Science*, pages 341–358. Springer-Verlag, New York, 1994.

[4] I.B. Damgård. A design principle for hash functions. In G. Brassard, editor, *Advances in Cryptology: Proceedings of CRYPTO '89,* volume 435 of *Lecture Notes in Computer Science,* pages 416–427. Springer-Verlag, New York, 1990.

[5] J. Galvin and K. McCloghrie. *RFC 1446: Security Protocols for version 2 of the Simple Network Management Protocol (SNMPv2).* Trusted Information Systems and Hughes LAN Systems, April 1993.

[6] R. Merkle. One way hash functions and DES. In G. Brassard, editor, *Advances in Cryptology: Proceedings of CRYPTO '89*, volume 435 of *Lecture Notes in Computer Science*, pages 428–446. Springer-Verlag, New York, 1990.

[7] National Institute of Standards and Technology (formerly National Bureau of Standards). *FIPS PUB 113: Computer Data Authentication.* May 30, 1985.

[8] National Institute of Standards and Technology. *FIPS PUB 180: Secure Hash Standard (SHS).* May 11, 1993.

[9] R. Rivest. *RFC 1321: The MD5 Message-Digest Algorithm.* RSA Data Security, Inc., April 1992.

[10] Gene Tsudik. Message authentication with one-way hash functions. *ACM Computer Communications Review*, 22(5):29–38, 1992.