Message Authentication with One-Way Hash Functions

Gene Tsudik
IBM Zürich Research Laboratory
CH-8803, Rüschlikon
Switzerland
gts@zurich.ibm.com

Abstract

Fast message integrity and authentication services are very important in today's high-speed network protocols. Current message authentication techniques are mostly encryption-based which is undesirable for several reasons. In this brief paper, we introduce encryption-free message authentication based entirely on the use of one-way hash functions. Two methods are presented and their strength is analyzed. The security of the proposed methods is based on the strength of the underlying one-way hash function.

Keywords: message authentication, data integrity, one-way hash functions, network protocols, communication security.

1 Introduction

Message authentication is a an important feature in many of today's network protocols. As network speeds increase, higher demands are made for processing speeds. However, encryption technology is still unable to match the bandwidth requirements of high-speed protocols in a cost-effective manner. For this reason, alternative approaches are being considered.

In recent years, some important advances were made in development of fast one-way hash functions. Two notable examples are Merkle's SNEFRU [7], Rivest's MD4 Message Digest algorithm [11] and the more recent (but slower) MD5 Message Digest Algorithm [12]. All of these hash functions are aimed at high-speed software implementations and all are currently in the public domain. MD4 is considered by many in the security community to be the most promising hash function (partly because the two-pass version of SNEFRU was *broken* by Biham and Shamir in 1990 [2]).

In this paper we explore some new applications of one-way hash functions. Hereafter, MD4 (MD5 is substantially similar) is used as the example function. The rest of the paper is organized as follows. Motivating factors for encryption-free message authentication are discussed in the next section. Section 3 reviews the key features of MD4. The proposed methods are described in Section 4 and subsequently analyzed in Section 5. A third, hybrid method is introduced in Section 6. The protocol costs are evaluated in Section 7 and potential applications are briefly discussed in Section 8.

2 Motivation

Until recently, fast one-way hash functions have been used primarily for maintaining message data integrity. In the realm of secure communications, message integrity alone does not provide sufficient defense against some attacks. While a strong one-way hash function protects against data modification, protection against *message substitution* requires that the message integrity value be irreproducible by anyone but the message originator.

Encryption-based integrity methods are able to provide this service, since one of the inputs to the integrity function is a secret key (e.g., DES). This is not the case with MD4 or any other non-trap-door hash function. Since MD4 implementations are readily available, anyone can produce an MD4 value of an arbitrary bogus message and insert $[msg, MD4(msg)]$ into the stream.

One simple solution is to have the sender encrypt (or sign) the message digest. In a public key cryptosystem, this can be done with the sender's private key. In a conventional cryptosystem, the sender can use the secret key shared amongst the appropriate parties.

This simple approach solves the message substitution problem, however, it has some drawbacks. Although, encryption usage is reduced to signing a short fixed-length message digest, there remain a few reasons for avoiding encryption altogether:

- Encryption software is quite slow
  The fastest available DES software implementation achieves data rates of ca. 700 Kbits/sec on a 16Mhz IBM PS2. An (unoptimized) MD4 implementation on the same hardware platform runs at 2.4 Mbits/sec.

An earlier version of this paper was presented at IEEE INFOCOM 92, May 4-8, Florence, Italy.

- Encryption hardware costs are non-negligible
  While software implementations tend to offer lower rates, their cost is typically low. Furthermore, many software implementations of DES as well as SNEFRU and MD4 have been placed in the public domain.

- Encryption hardware is optimized towards large data sizes
  Most hardware devices are optimized for bulk encryption. Encryption rates in the range of 45 Mbits/sec are not unheard of, yet performance tends to decrease drastically for small data units (e.g., $\leq 1Kbyte$). Typically, this is due to hardware initialization/invocation overhead (which is amortized for large units of data). Software implementations (MD4 being a case in point) tend to exhibit constant data rates irrespective of data sizes.

- Encryption algorithms are covered by patents
  Some encryption and signature algorithms (e.g., RSA [10]) are patented and must be licensed; with obvious cost implications.

- All encryption algorithms are subject to export control
  This is one of the chief arguments against the use of encryption for message authentication. For example, the United States Government places export restrictions on certain cryptographic *paraphernalia*. All DES implementations (hardware and software) fall into this category.

Drawing from the discussion above, an ideal scenario for countering both message substitution and message modification attacks would:

1. Avoid encryption altogether, and
2. Produce unforgeable integrity values

Initially, these two properties appear to be at odds, however, some innovative application of fast one-way hash functions may be able to reconcile them. After briefly touching upon the key features of MD4 in the next section, we turn to the description of the proposed protocols.

3 MD4 Summary

In brief, MD4 computes a 128-bit digest of an arbitrary length message. Its basic input unit is a 512-bit block. MD4 divides the input message into a number of 512-bit blocks (padding the message, if necessary) and appends an extra trailing block which contains the message length in its low-order 64 bits (the remainder of the trailing block contains all zeros).

It has been conjectured that MD4 has the following properties:

1. The difficulty of computing two distinct messages, M_1 and M_2 such that $MD4(M_1) = MD4(M_2)$ is on the order of 2^{64} operations.
2. Given a message digest, MD, the difficulty of computing a message, M such that $MD4(M) = MD$ is on the order of 2^{128} operations.

In this paper, the second property is of particular interest. It is based on the fact that MD4 maps 2^{512} 512-bit blocks into 2^{128} 128-bit digests. This implies that, on the average, $[2^{512}/2^{128} = 2^{384}]$ blocks are hashed into the same 128-bit digest. Therefore, given a message digest, it takes, on the average, 2^{128} trials before discovering one of the 2^{384} messages that maps into that digest.

4 Protocol Description

We suppose that two principals, A and B, would like to communicate over an insecure channel. Furthermore, there exist secure means of principal authentication [8, 16, 13]. At the time of session initiation, after mutual authentication is achieved, one of principals, say, A, generates a random 512-bit value, S_{AB}. A then communicates S_{AB} to B in secret (using encryption, if necessary).

When A has a message M to send to B, it computes $MD_M = MD4(M||S_{AB})$.[1] It then sends $[M, MD_M]$ to B. Since B possesses S_{AB}, it can re-compute $MD4(M||S_{AB})$ and verify MD_M. This is known as the *secret suffix* technique.

Alternatively, A and B can agree to compute MD_M as $MD4(S_{AB}||M)$ and use S_{AB} as a *secret prefix*. The secret prefix method was developed independently by the Internet Security and Privacy Working Group (SPWG) for use in Simple Network Management Protocol (SNMP) [6].[2]

5 Informal Analysis

Throughout the following discussion it is assumed that an intruder has the ability to record and analyze messages as well as insert fraudulent messages into the message stream. In order to mount a successful attack or effectively *break* the protocol, an intruder has to be able to produce a fraudulent message and an accompanying digest such that the $(message, digest)$ pair is accepted as genuine by any of the legitimate principals.

[1] $||$ denotes concatenation.
[2] S. Crocker, S. Kent, Private Communication, December 1990.

The following observations are based on the second property of the MD4 algorithm:[3]

- **Claim 1:** *Given a message digest, MD, and a message M_1, the difficulty of computing a message, M_2 such that $MD4(M_2||M_1) = MD$ is on the order of 2^{128} operations.*

- **Claim 2:** *Given a message digest, MD, and a message M_1, the difficulty of computing a message, M_2 such that $MD4(M_1||M_2) = MD$ is on the order of 2^{128} operations.*

Assuming these two claims (and the two conjectured properties) hold, the only venue left to the intruder is the *brute force* attack. For most purposes, attacks that take $\approx 2^{128}$ are considered *futile*, i.e., the system in question is secure.[4] Nevertheless, understanding the details of possible attacks is of interest. Two seemingly similar methods proposed below differ in their ability to withstand brute force attacks.

5.1 Definitions

- Given two 512-bit blocks, M_1 and M_2, we say that $M_1 \equiv M_2$ if $MD4(M_1) = MD4(M_2)$. The '$\equiv$' symbol denotes that M_1 and M_2 are *prefix-equivalent*. For a given 512-bit block M, let PEC_M denote the *prefix-equivalence class* of M.

- Two 512-bit blocks, $M_1 \doteq M_2$ if for all blocks M, $MD4(M||M_1) = MD4(M||M_2)$. The '$\doteq$' symbol denotes that M_1 and M_2 are *suffix-equivalent*. Let SEC_M denote the *suffix-equivalence class* of a 512-bit block M.

5.2 Secret Prefix

A naive intruder wishing to discover the secret prefix would first record a message M accompanied by its integrity value, $MD4(S_{AB}||M)$. He would then need to try on the average 2^{128} possibilities before discovering a prefix string $\hat{S}$ where $\hat{S} \equiv S_{AB}$.

One important observation that can be made by a more *astute* intruder is that the secret prefix, S_{AB}, is not really what is needed to *break* the protocol. In other words, the secret prefix is sufficient, yet not necessary, for a successful attack. Instead, to attain the goal of composing *genuine* messages, the intruder needs only the intermediate MD4 value of S_{AB}, i.e., $MD4(S_{AB})$. The length of $MD4(S_{AB})$ is 128 bits regardless of the length of S_{AB}. Nevertheless, the number of operations required to find $MD4(S_{AB})$ remains a hefty 2^{128}.

The subtle difference between the naive and the more *educated* brute force attacks is that, in the latter, the intruder is guaranteed to discover $MD4(S_{AB})$ (hence, successfully attacking the protocol) in at most 2^{128} operations. This is in contrast to the former where on the **average** 2^{128} operations are required. In summary, the shortcut makes the attack *deterministic* as opposed to *probabilistic*.

Another curious property of the secret prefix method is its vulnerability to *padding* attacks.[5] (A *padding* attack is successful when the intruder is able to either prepend or append extraneous data to an authentic message and pass the resulting message off as genuine.) An intruder can capture a message M along with its secret prefix-based check $MD4(S_{AB}||M)$. He can then append arbitrary data, $\hat{M}$, to the end of M and compute the digest of the resulting message, $(M||\hat{M})$ using $MD4(S_{AB}||M)$ as the initialization value. The legitimate receiver is then *fooled* into accepting the fraudulent message because the accompanying secret prefix value, $MD4(S_{AB}||M||\hat{M})$, is "correct". One safe-guarding measure is already implemented by MD4; the last 512-bit block always consists of 448 bits of zeroes followed by 64 bits that contain the message length. This, by itself, does not alleviate padding attacks, however, it makes the attacker's job more difficult by forcing him to include this trailing block of the genuine message as part of the "padded" bogus message.

The padding attack is possible for two reasons: 1) the attacker knows the underlying algorithm (MD4), and, 2) the integrity value is not protected. An obvious countermeasure for this type of attack is to require that the message length be included in the secret prefix calculation, e.g., as part of the first block. Some protocols already include length as part of the message header format (e.g., IP [9], SNMP [6]).

5.3 Secret Suffix

A brute force attack on the secret suffix variant is similar in that the intruder would need to try on the average 2^{128} possibilities before discovering an $\hat{S}$ such that $MD4(M||\hat{S}) = MD4(M||S_{AB})$. However, the probability that $\hat{S} = S_{AB}$ is only $1/2^{384}$ (since there are, on the average, 2^{384} 512-bit blocks which are hashed into the same 128-bit MD4 value). On the other hand, finding an $\hat{S}$ such that $\hat{S} \doteq S_{AB}$ is, for all purposes, equivalent to finding S_{AB}.

At this point in time, because of MD4's relatively recent introduction, and, hence, limited analysis thereof, little can be said with certainty about the average size

[3] See Section 1.

[4] For example, a brute force attack on DES requires only $\approx 2^{56}$ operations.

[5] This vulnerability was discovered by Dave Solo and Steve Kent.

of suffix-equivalence classes. One conjecture that can be made with some degree of certainty is that the size of an average suffix-equivalence class is no greater than that of an average prefix-equivalence class.

This does not imply, however, that it takes at most 2^{128} operations to find $\hat{S} \doteq S_{AB}$. Note that the intruder has to record multiple *genuine* messages and compute their respective digests at each step of the attack. Thus, the number of operations required to find $\hat{S}$ is $[N \times 2^{128}]$ where N is the size of the intruder's *message test pool.*

We also note that, in the suffix variant, there appears to be no counterpart to the "brute force" shortcut described in the previous section. Furthermore, secret suffix is secure with respect to padding attacks. This is because a message digest is computed with a secret suffix as the last input block. Without knowing the secret, the intruder can not, with any certainty, append (or prepend) to the message.

One drawback of the secret suffix is its susceptibility to the so-called *Birthday* attack [3]. In brief, this attack consists of the intruder generating a message trial pool of size R and recording a sample pool of genuine messages of size S. The probability of at least one message in a trial pool hashing to the same MD4 value as one of the messages in the sample pool is roughly expressed by $P = (R * S/N)$ where N is the range ($N = 2^{128}$ in our case). According to the *Birthday* paradox, there is a 50% chance of two messages hashing to the same MD4 value for $R = 2^{64}$ and $S = 1$. This feature forms the basis for the first property of MD4 (see Section 3).

The implication of this attack on the secret suffix method is as follows. The intruder can pre-compute 2^{64} sample messages along with the corresponding digests. Then, after recording a single genuine message (M) protected by the secret suffix, the intruder computes $MD4(M)$. The probability is high ($\geq 50\%$) that there exists a bogus message $\tilde{M}$ in the sample message pool such that $MD4(M) = MD4(\tilde{M})$. On the other hand, this attack requires $\approx 2^{64}$ operations along with 2^{64} storage space for the trial pool.[6]

In contrast, the prefix method is not susceptible to the Birthday attack. As above, the intruder can still discover (with high probability) a bogus message whose MD4 value matches that of a recorded genuine message. However, while:

- $MD4(M) = MD4(\tilde{M})$ $(M \neq \tilde{M})$ implies that $MD4(M||S_{AB}) = MD4(\tilde{M}||S_{AB})$

it is **not** the case that:

- $MD4(M) = MD4(\tilde{M})$ implies that $MD4(S_{AB}||M) = MD4(S_{AB}||\tilde{M})$

[6] However, the generation of the message trial pool can be done off-line; only the search has be carried out in real time. See [3] for a detailed discussion.

6 *Envelope* Method

At present time, a protocol requiring $\approx 2^{128}$ operations to break can be considered quite strong. Nevertheless, it is conceivable that a need for stronger protocols may arise in the future. For this reason, we consider a "hybrid" combination of the two methods described thus far.

In this third variant, message digest is computed by using a secret initialization vector in conjunction with a secret 512-bit suffix. In other words, the message is *enveloped* with a secret prefix and a secret suffix before its digest is computed.

The envelope method is significantly stronger than either prefix or suffix variants. The reasons for this are informally sketched below.

The most serious attack enables the intruder to compose and send arbitrary messages that are accepted as genuine by the receiving party. In order to succeed, the intruder has to discover $[P, S]$ (i.e., the secret prefix and the secret suffix) used by the two communicating parties or discover some other $[\hat{P}, \hat{S}]$ such that for all possible 512-bit blocks M, $MD4(P||M||S) = MD4(\hat{P}||M||\hat{S})$.

In order to assess the difficulty of mounting this attack, an important property of the envelope method needs to be observed:

• $\rightarrow$ **It is impossible for the intruder to discover P (or $\hat{P}$) without discovering the corresponding S (or $\hat{S}$).**

Proof sketch: Assume that the intruder has recorded a genuine message M and the accompanying digest, $MD = MD4(P||M||S)$. In the course of interating through the trial values of the secret prefix, suppose the intruder comes upon the "correct" value of P. In order to determine that the trial value is correct, the intruder has to perform a *test.* This test is possible only if:

- The intruder knows $MD4(P||M)$. Then, he can search for $\hat{P}$ such that $MD4(\hat{P}||M) = MD4(P||M)$. However, it is infeasible to obtain $MD4(P||M)$ from M and $MD4(P||M||S)$.

  OR

- The intruder must simultaneously select a trial value for the secret suffix. In other words, he must interate on $\hat{S}$ in *parallel* with iterating on $\hat{P}$.

The opposite assertion follows:

• $\rightarrow$ **It is impossible for the intruder to discover S (or $\hat{S}$) without discovering the corresponding P (or $\hat{P}$).**

Based on the above, we can conclude that it takes on the order of $\left[\sharp_P \times \sharp_S\right]$ operations in order to break the envelope method. (Where $\sharp_P$ and $\sharp_S$ denote the number of trials required to break the secret prefix and secre suffix methods, respectively.) Assuming that $\approx 2^{128}$ are required to break each of the prefix and suffix methods, the envelope method provides much greater protection with $\approx 2^{256}$ operations.

Another benefit of the envelope method is its resistance to both padding and Birthday attacks. Padding attack is made impossible because the secret suffix is used (see Section 5). It is less obvious why the Birthday attack is no longer feasible. As discussed in Section 5, this attack is possible with the secret suffix method because the intruder has the ability to compute the digest of a genuine recorded message. Having recorded a message M, the intruder can discover (possibly by precomputing a large number of messages) another, bogus message $\hat{M}$ such that $MD4(M) = MD4(\hat{M})$. Consequently, he can substitute M with $\hat{M}$ and succeed in subverting the receiving party into believing that $\hat{M}$ is genuine (because of the property that $MD4(M||S) = MD4(\hat{M}||S)$ for all suffixes S).

However, in the envelope method, substituting M with $\hat{M}$ does not have the desired effect. Although $MD4(M) = MD4(\hat{M})$, $P||M$ and $P||\hat{M}$ have a very low probability (2^{128}) of hashing into identical digests.

7 Cost

As evident from the above discussion, provided MD4's conjectured properties hold, both secret suffix and secret prefix methods provide strong protection against message substitution and tampering. Just as there are some subtle differences between the two variants with respect to their ability to resist hostile attacks, there are also differences in their respective costs.

The prefix variant costs close to nothing, since the intermediate MD4 value of the secret prefix can be pre-computed and used as an *initialization vector* for verifying message signatures. Furthermore, the concept of a *prefix* can be abandoned altogether in favor of a 128-bit *MD4 initialization vector*.

Secret suffix costs slightly more as no pre-computation can be done. The cost amounts to exactly one extra 512-bit block MD4 computation per message.

8 Applications

The secret suffix/prefix methods are particularly applicable to environments where high bandwidth and low delay requirements have, until now, made the implementation of certain security services prohibitively expensive, e.g., packetized voice and video applications. SNMP, as stated above, already implements the secret prefix method. Also, network-layer and routing protocols, e.g., SDNS SP3 [14], Inter-Domain Policy Routing (IDPR) [15] and Visa Protocol [4], can benefit substantially from inexpensive, encryption-free message authentication.

9 Conclusions

In summary, fast one-way hash functions such as MD4/5 can be used as a foundation for some relatively novel implementations of security services. In particular, simple and inexpensive secret prefix and secret suffix methods provide protection against message substitution attacks when used in conjunction with a strong one-way hash function (which itself protects against message modification).

There remain some outstanding issues. Most importantly, the methods presented in this paper are very much dependent on the *conjectured* properties of MD4. Since MD4 is still a relative newcomer, it has not yet been subjected to extensive analysis. Furthermore, the discussion in this paper has been largely informal; more rigorous scrutiny of the proposed methods is in order.

10 Acknowledgements

The author would like to thank Steve Kent, Deborah Estrin and Refik Molva for their comments on the earlier draft of this paper.

References

[1] Advanced Micro Devices, *AMD MOS Microprocessors and Peripherals Data Book.*, **Advanced Micro Devices, Inc., Sunnyvale, CA.**, 1987

[2] E. Biham, A. Shamir, *Differential Cryptanalysis of DES-like Cryptosystems*, **Proceedings of CRYPTO'90**, August 1990.

[3] D. Davies, W. Price, *Security For Computer Networks*, **New York, NY: Wiley**, 1984.

[4] D. Estrin, J. Mogul, G. Tsudik, *Visa Protocols for Controlling Inter-Organizational Datagram Flow*, **IEEE Journal on Selected Areas in Communications**, May 1989.

[5] *Federal Information Processing Standards*, National Bureau of Standards, Publication 46, 1977.

[6] J. Galvin, K. McCloghrie, J. Davin, *Secure Management of SNMP Networks* **Proceedings of IFIP Integrated Network Management Symposium**, April 1991.

[7] R. Merkle, *One-way Hash functions and DES*, **Proceedings of CRYPTO 89**, August 1989.

[8] R. Needham and M. Schroeder, *Using encryption for authentication in large networks of computers*, **Communications of the ACM**, December 1978.

[9] J. Postel, *Internet Protocol*, **RFC 791, SRI Network Information Center**, September 1981.

[10] R. Rivest, A. Shamir and L. Adelman, *A Method for Obtaining Digital Signatures and Public Key Cryptosystems*, **Communications of the ACM**, February 1978.

[11] R. Rivest, *The MD4 Message Digest Algorithm*, **Proceedings of CRYPTO'90**, August 1990.

[12] R. Rivest, *The MD5 Message Digest Algorithm*, **Internet DRAFT**, July 1991.

[13] J. Steiner, *The Kerberos Network Authentication Service Overview*, **MIT Project Athena RFC, Draft 1**, April 1989.

[14] SDNS Protocol and Signaling Working Group, SP3 Subgroup, *SDNS Secure Data Network System Security Protocol 3 (SP3)*, **SDN.301 Revision 1.5 1989-05-15**

[15] M. Steenstrup and Internet Open Routing Working Group, *Inter-Domain Policy Routing Specification*, **BBN Report Number 7346**, December 1990, BBN Corporation, Cambridge MA 02138.

[16] V. Voydock and S. Kent, *Security Mechanisms in High-level Network Protocols*, **ACM Computing Surveys**, June 1983.