Cryptanalysis of Message Authentication Codes[*]

B. Preneel[†]
Katholieke Universiteit Leuven
Department Electrical Engineering-ESAT/COSIC
K. Mercierlaan 94, B-3001 Heverlee, Belgium
`bart.preneel@esat.kuleuven.ac.be`

September 15, 2004

Abstract

This paper gives a survey of attacks on Message Authentication Codes (MACs). First it defines the required security properties. Next it describes generic forgery and key recovery attacks on MACs. Subsequently an overview is presented of most MAC constructions and on attacks on these algorithms. The MACs described include CBC-MAC and its variants, the MAC algorithms derived from cryptographic hash functions, and the ISO banking standard Message Authenticator Algorithm, also known as MAA.

1 Introduction

Digital signatures, introduced in 1976 by W. Diffie and M. Hellman [14], allow to establish in an irrefutable way the origin and content of digital information. They are gaining widespread attention for applications such as electronic commerce, which requires a worldwide public key infrastructure. Almost all practical digital signature schemes (such as RSA [44]) depend on the (believed) hardness of a number theoretic problem. The technical problem of designing digital signature schemes and implementing them is well understood, but for some applications they require still too much storage and computation.

Therefore many applications still use conventional Message Authentication Code (MAC) algorithms to provide data integrity and data origin authentication. MACs provide weaker guarantees than signatures, as they can only be used in a symmetric setting, where the parties trust each other. In technical terms, they do not provide non-repudiation of origin. Moreover, MACs rely on shared symmetric keys, the management of which is more costly and harder to scale than that of asymmetric key pairs. Banks have been using MACs since the late seventies [13] for message authentication. Recent applications in which MACs have been introduced include electronic purses (such as Proton and Mondex) and credit/debit applications (e.g., the EMV specifications). MACs are also being deployed for securing the Internet (e.g., IP security [3, 26]). For all these applications MACs are preferred over digital signatures because they are two to three orders of magnitude faster, and MAC results are 4. . . 16 bytes long compared to

[*]An earlier version of this paper has been published in *Information Security, Lecture Notes in Computer Science 1396*, E. Okamoto, G. Davida, and M. Mambo, Eds., Springer-Verlag, 1998, pp. 55-66.

[†]F.W.O. postdoctoral researcher, sponsored by the National Fund for Scientific Research – Flanders (Belgium).

Appeared in *Information Security, 1st International Workshop, ISW 1997*, Lecture Notes in Computer Science 1396, G. I. Davida, M. Mambo, and E. Okamoto (eds.), Springer-Verlag, pp. 55–65, 1997.

40. . . 128 bytes for signatures. On present day machines, software implementations of MACs can achieve speeds from 10. . . 100 Mbit/s and more, and MACs require very little resources on inexpensive 8-bit smart cards and on the currently deployed Point of Sale (POS) terminals.

In order to use a MAC, sender and receiver need to share a secret key K (a random bit-string of k bits – typical values for k are 56. . . 128). In order to protect a message, the sender computes the MAC corresponding the message, which is bit-string of m bits, and appends this string to the message (typical values for m are 32. . . 64). The MAC is a complex function of every bit of the message and the key. On receipt of the message, the receiver recomputes the MAC and verifies that it corresponds to the transmitted MAC value (see also Fig. 1).

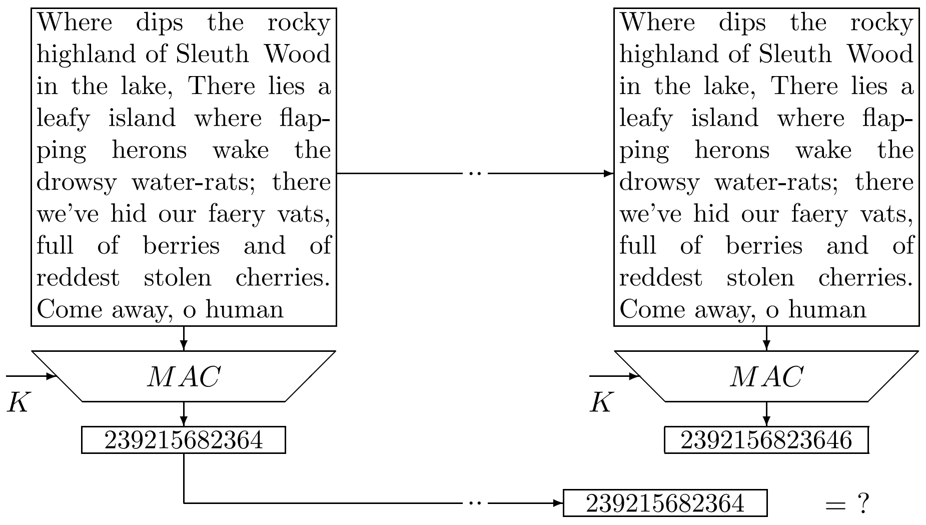
Where dips the rocky highland of Sleuth Wood in the lake, There lies a leafy island where flapping herons wake the drowsy water-rats; there we've hid our faery vats, full of berries and of reddest stolen cherries. Come away, o human
Where dips the rocky highland of Sleuth Wood in the lake, There lies a leafy island where flapping herons wake the drowsy water-rats; there we've hid our faery vats, full of berries and of reddest stolen cherries. Come away, o human
MAC
K
MAC
K
239215682364
2392156823646
239215682364
= ?

Figure 1: Using a Message Authentication Code for data integrity.

2 Security of MAC algorithms

An opponent who tries to deceive the receiver, does not know the secret key. It will be assumed that he knows the format of the messages, and the description of the MAC algorithm. His goal is to try to inject a fraudulent message and append a MAC value which will be accepted by the receiver. He can choose one of two attack strategies:

forgery attack: this attack consists of predicting the value of $\text{MAC}_K(x)$ for a message x without initial knowledge of K. If the adversary can do this for a single message, he is said to be capable of *existential forgery*. If the adversary is able to determine the MAC for a message of his choice, he is said to be capable of *selective forgery*. Ideally, existential forgery is computationally infeasible; a less demanding requirement is that only selective forgery is so. Practical attacks often require that a forgery is *verifiable*, i.e., that the forged MAC is known to be correct on beforehand with probability near 1.

key recovery attack: this attack consists of finding the key K itself from a number of message/MAC pairs. Such an attack is more powerful than forgery, since it allows for arbitrary selective forgeries. Ideally, any attack allowing key recovery requires about 2^k

operations (here k denotes the bit-length of K). Verification of such an attack requires k/m text-MAC pairs.

In addition, one requires that computing $\mathrm{MAC}_K(x)$ is easy, given MAC(), K, and an input x.

The attacks above can be further classified according to the type of control an adversary has over the device computing the MAC value. In a *chosen-text attack*, an adversary may request and receive MACs corresponding to a number of messages of his choice, before completing his (forgery or key recovery) attack. For forgery, the forged MAC must be on a message different than any for which a MAC was previously obtained. In an *adaptive* chosen-text attack, requests may depend on the outcome of previous requests.

Note that in certain environments, such as in wholesale banking applications, a chosen message attack is not a very realistic assumption: if an opponent can choose a single text and obtain the corresponding MAC, he can already make a substantial profit. However, it is better to be on the safe side, and to require resistance against chosen text attacks.

One can now write down a formal definition of security for a MAC, by stating that, after a number of chosen/known texts, a forgery with a certain probability requires a certain amount of time. See for example [7] (note that this definition does not distinguishes between chosen and known texts).

In the following, different attacks on MACs are considered: brute force key search, guessing of the MAC, a generic forgery attack, and attacks based on cryptanalytical weaknesses.

2.1 Brute force key search

A brute force key search needs a few known message-MAC pairs (about k/m, which is between 1 and 4 for most MAC algorithms). It is reasonable to assume that such a small number of message-MAC pairs is available. The opponent tries all the possible keys and checks whether they correspond to the given message-MAC pairs. Note that unlike for confidentiality protection, the opponent can only make use of the key if it is recovered within its active lifetime (which can be reasonably short). On the other hand, a single success during the lifetime of the system might be sufficient. This depends on a cost/benefit analysis, i.e., how much one loses as a consequence of a forgery.

The only way to preclude a key search is to choose a sufficiently large key. Currently finding a 56-bit key during a period of 1 year will require an investment of 50 000\$, and it can be done in a few months by using idle cycles on the Internet, as was demonstrated in the Spring of 1997. Moreover, one also has to take into account what is known as 'Moore's Law': the computing power for a given cost is multiplied by four every 3 years. This implies that if a system is deployed with an intended lifetime of 15 years, an extra security margin of about 10 bits is recommended. The important conclusion is that keys over 90 bits are sufficient for 15 years or more. More details on key lengths (but for confidentiality protection) can be found in the reports of M. Wiener [48] and of Blaze et al. [8].

2.2 Guessing of the MAC

A second very simple 'attack' is to choose an arbitrary fraudulent message, and to append a randomly chosen MAC value. Ideally, the probability that this MAC value is correct is equal to $1/2^m$, where m is the number of bits of the MAC value. This value should be multiplied with the expected profit corresponding to a fraudulent message, which results in the expected value of one trial. Repeated trials can increase this expected value, but note that in a good implementation, repeated MAC verification errors will result in a security alarm (the forgery is not verifiable). For most applications $m = 32 \ldots 64$ is sufficient to make this attack uneconomical.

2.3 Generic forgery attack

This attack exploits the fact most MAC algorithms consist of the iteration of a simple compression function. The MAC input x is padded to a multiple of the block size, and is then divided into t blocks denoted x_1 through x_t. The MAC involves a *compression function* f and an n-bit ($n \geq m$) *chaining variable* H_i between stage $i-1$ and stage i:

$$\begin{aligned} H_0 &= IV \\ H_i &= f(H_{i-1}, x_i), \qquad 1 \leq i \leq t \\ \mathrm{MAC}_K(x) &= g(H_t)\,. \end{aligned}$$

Here g denotes the *output transformation.* The secret key may be employed in the IV, in f, and/or in g.

For an input pair (x, x') with $\mathrm{MAC}_K(x) = g(H_t)$ and $\mathrm{MAC}_K(x') = g(H'_t)$, a collision is said to occur if $\mathrm{MAC}_K(x) = \mathrm{MAC}_K(x')$. This collision is called an *internal* collision if $H_t = H'_t$, and an *external* collision if $H_t \neq H'_t$ but $g(H_t) = g(H'_t)$.

In [37], B. Preneel and P.C. van Oorschot describe a general forgery attack that applies to all iterated MACs. Its feasibility depends on the bitsizes n of the chaining variable and m of the MAC result, the nature of the output transformation g, and the number s of common trailing blocks of the known texts ($s \geq 0$). The basic attack requires several known texts, but only a single chosen text. However, under certain conditions restrictions are imposed on the known texts; for example, if the message length is an input to the output transformation, all messages must have equal length. The attack starts with the following simple observation (e.g., [25, 37]):

Lemma 1 *An internal collision for an iterated MAC allows a verifiable MAC forgery, through a chosen-text attack requiring a single chosen text.* ∎

This follows since for an internal collision (x, x'), $\mathrm{MAC}_K(x \,\|\, y) = \mathrm{MAC}_K(x' \,\|\, y)$ for any single block y; thus a requested MAC on the chosen text $x \,\|\, y$ provides a forged MAC (the same) for $x' \,\|\, y$ (here $\|$ denotes concatenation). Note this assumes that the MAC algorithm is deterministic. Also, the forged message is of a special form, which may limit the practical impact of the attack.

The main question is to determine how one can find such an an internal collision for a given MAC algorithm. The answer is given by the following propositions [37, 40].

Proposition 1 *Let* MAC() *be an iterated MAC with n-bit chaining variable and m-bit result, and an output transformation g that is a permutation. An internal collision for h can be found using an expected number of $u = \sqrt{2} \cdot 2^{n/2}$ known text-MAC pairs of at least $t = 2$ blocks each.*

Proposition 2 *Let* MAC() *be an iterated MAC with n-bit chaining variable and m-bit result, and output transformation g which is a random function. An internal collision for h can be found using u known text-MAC pairs of at least $t = 2$ blocks each and v chosen texts of at* *least three blocks. The expected values for u and v are as follows: $u = \sqrt{2} \cdot 2^{n/2}$ and v is approximately*

$$2\left(2^{n-m}\left(1-\frac{1}{e}\right)+\left\lfloor\frac{n-1}{m-1}\right\rfloor\right) \approx 2^{n-m}\,. \tag{1}$$

Proposition 3 [1] *Let* MAC() *be an iterated MAC with n-bit chaining variable, m-bit result, a compression function f which behaves like a random function (for fixed x_i), and output transformation g. An internal collision for* MAC *can be found using u known text-MAC pairs, where each text has the same substring of $s \geq 0$ trailing blocks, and v chosen texts. The expected values for u and v are: $u = \sqrt{2/(s+1)} \cdot 2^{n/2}$; $v = 0$ if g is a permutation or $s+1 \geq 2^{n-m+6}$, and otherwise*

$$v \;\approx\; 2\left(\frac{2^{n-m}}{s+1}\cdot\left(1-\frac{1}{e}\right)+\left\lfloor\frac{n-m-\log_2(s+1)}{m-1}\right\rfloor+1\right)\,. \tag{2}$$

■

A simple way to preclude this attack is to append a sequence number at the *beginning* of every message and to make the MAC algorithm stateful. This means that the value of the sequence number is stored to ensure that each sequence number is used only once within the lifetime of the key. While this is not always practical, it has the additional advantage that it prevents replay attacks [13].

2.4 Weaknesses of the algorithm

The above attacks assume that no shortcuts exist to break the MAC algorithm (either for forgery or for key recovery). Since most existing MAC algorithms are not provably secure, it is recommended to use only well established algorithms which have been subjected to an independent evaluation. Even if this has been performed, a regular review of the algorithm based on progress in cryptanalysis is recommended.

3 Practical MAC algorithms

Compared to the number of block ciphers and hash functions, there relatively few MAC algorithms have been proposed. The main reason is that MACs have been derived from other primitives (initially from block ciphers and currently from hash functions), which reduces the need for dedicated proposals. This section reviews the known constructions.

3.1 Based on block ciphers

The most popular MAC algorithm is certainly CBC-MAC; it has been adopted by many standardization committees including ANSI and ISO/IEC [1, 2, 21, 22]. It is widely used with DES [16] as the underlying block cipher. CBC-MAC is an iterated MAC, with the following compression function:

$$H_i = E_K(H_{i-1} \oplus x_i)\,,\;\; 1 \leq i \leq t\,.$$

Here $E_K(x)$ denotes the encryption of x using the k-bit key K with an n-bit block cipher E and $H_0 = 0$. The MAC is then computed as $\text{MAC}_K(x) = g(H_t)$, where g is the output

[1]The details of this proposition have not been discussed in the lectures.

transformation. The mapping g is required to preclude the following simple forgeries (it is assumed that x, x', and y fall on block boundaries):

- given $\mathrm{MAC}(x)$, one knows that $\mathrm{MAC}_K(x \| (x \oplus \mathrm{MAC}_K(x))) = \mathrm{MAC}_K(x)$ (for a single block x);
- given $\mathrm{MAC}(x)$ and $\mathrm{MAC}(x')$ one knows that $\mathrm{MAC}_K(x \| (x' \oplus \mathrm{MAC}_K(x))) = \mathrm{MAC}_K(x')$ (for a single block x').
- given $\mathrm{MAC}(x)$, $\mathrm{MAC}(x \| y)$, and $\mathrm{MAC}(x')$, one knows that $\mathrm{MAC}(x' \| y') = \mathrm{MAC}(x \| y)$ if $y' = y \oplus \mathrm{MAC}(x) \oplus \mathrm{MAC}(x')$.

One approach is for g to select the leftmost m bits. However, L. Knudsen has shown that the simple attack can be extended to this case [27]; it requires then approximately $2 \cdot 2^{(n-m)/2}$ chosen texts and 2 known texts.

A widely used and better alternative is to replace the processing of the last block by a two-key triple encryption (with keys $K_1 = K$ and K_2); this is commonly known as the ANSI retail MAC, since it first appeared in [2]:

$$g(H_t) = E_{K_1}(D_{K_2}(H_t)) = E_{K_1}(D_{K_2}(E_{K_1}(x_t \oplus H_{t-1}))) .$$

Here D denotes decryption. This mapping requires little overhead, and has the additional advantage that it precludes an exhaustive search against the 56-bit DES key.

A second alternative is the use of a derived key K' (as opposed to a second independent key):

$$g(H_t) = E_{K'}(H_t) = E_{K'}(E_K(x_t \oplus H_{t-1})) .$$

This solution was proposed by the RIPE Consortium in [41] and has been included in IS 9797 [22].

All these variants are vulnerable to the forgery attack described in Sect. 2.3, which requires a single chosen message and about $2^{n/2}$ known messages (for DES this corresponds to 2^{32} known messages). For $m < n$, an additional 2^{m-n} chosen messages are required, which makes the attack less realistic.

For the ANSI retail MAC, one does not only obtain a forgery, but one can also recover the key in time $3 \cdot 2^k$ encryptions, compared to 2^{2k} encryptions for exhaustive search [39]. If DES is used, this implies that key recovery may become feasible. Another key recovery attack needs only a single known text, but requires about 2^k MAC verifications. Moreover, it reduces the effective MAC size from m to $\min(m, k)$.

The security of the ANSI retail MAC against key recovery attacks can be improved at no cost in performance by introducing a double DES encryption in the first and last iteration. This scheme was proposed by Knudsen and the author under the name MacDES: [28]:

$$H_1 = E_{K_2'}(E_{K_1}(X_1)) \quad \text{and} \quad g(H_t) = E_{K_2}(H_t) .$$

Here K_2' is derived from K_2.

In the light of these recent attacks, ISO/IEC SC27 has recently started the process to revise IS 9797 [22]. It is very likely that the 1999 edition will contain CBC-MAC, the ANSI retail MAC, and MacDES, as well as double variants that exor the result of two MAC calculations. This provides increased resistance against forgery attacks.

Bellare et al. provide in [7] a proof of security for CBC-MAC, i.e., they establish a lower bound to break the system under certain assumptions on the block cipher. It almost matches the upper bound of the attack of Sect. 2.3.

An alternative to CBC-MAC is RIPE-MAC, which adds a feedforward [41]:

$$H_i = E_K(H_{i-1} \oplus x_i) \oplus x_i\,, \quad 1 \leq i \leq t\,.$$

It has the advantage that the round function is harder to invert (even for someone who knows the secret key). An output transformation is needed as well.

XOR-MAC is another scheme based on a block cipher [6]. It is a randomized algorithm and its security can again be reduced to that of the block cipher. It has the advantage that it is parallellizable and that small modifications to the message (and to the MAC) can be made at very low cost. The use of random bits clearly helps to improve security, but it has a cost in practical implementations. Also, the performance is typically 25 to 50% slower than CBC-MAC.

Note that cryptanalysis of the underlying block cipher can often be extended to an attack on CBC-MAC, even if $m < n$, which implies that an attacker obtains less information than in conventional cryptanalysis on the ECB mode (see [32, 35] for the case of DES).

3.2 Based on cryptographic hash functions

The availability of fast dedicated hash functions (such as MD4 [42] and MD5 [43]) has resulted in several proposals for MAC algorithms based on these functions. As it became clear that these hash functions are weaker than intended, they are currently being replaced by RIPEMD-160 [15] and by SHA-1 [17].

The first proposed constructions were the secret prefix and secret suffix methods which can be described as follows: $\mathrm{MAC}_K(x) = h(K\|x)$, $\mathrm{MAC}_K(x) = h(x\|K)$. However, the first one allows for extension attacks, and the second one opens the possibility of off-line attacks (see [37] for a more detailed discussion).

The next proposal the secret envelope method, which can be described as $\mathrm{MAC}_K(x) = h(K_1\|x\|K_2)$ (for example Internet RFC 1828 [31]). For this method, Bellare et al. provide a security proof based on the assumption that the compression function of the hash function is pseudo-random [4]. While this is an interesting result, it should be pointed out that the compression function of most hash functions has not been evaluated with respect to this property. Also, it was shown in [38] that $2^{n/2}$ known texts does not only allow a forgery (cf. Sect. 2.3), but also a key recovery attack.

MDx-MAC extends the envelope method by also introducing secret key material into every iteration [37]. This makes the pseudo-randomness assumption more plausible. Moreover, it precludes the key recovery attack by extending the keys to complete blocks.

HMAC is yet another variant, which uses a nested construction (also with padded keys):

$$\mathrm{MAC}_K(x) = h(K_2\|h(x\|K_1))\,.$$

HMAC will be used for providing message authentication in the Internet Protocol [3, 26]. The security of HMAC is guaranteed if the hash function is collision resistant for a secret value H_0, and if the compression function itself is a secure MAC for 1 block (with the secret key in the H_i input and the message in the x_i input) [5]. While these assumptions are weaker, we believe that the the latter one still requires further validation for existing hash functions.

3.3 Dedicated MACs[2]

The most important dedicated MAC algorithm is certainly the Message Authenticator Algorithm (MAA). MAA was designed by D. Davies [11, 12] and became an ISO banking standard in 1987. Recently several weaknesses of MAA have been exposed in [36, 38]. The forgery attack of Sec. 2.3 can be optimized; it requires only 2^{24} messages of 1 Kbyte; a corresponding key recovery attack needs 2^{32} chosen texts consisting of a single message block. The number of off-line multiplications for this attack varies between 2^{44} for one key in 1000 to about 2^{51} for one key in 50. This should be compared to about $3 \cdot 2^{65}$ multiplications for an exhaustive key search. Finally, several classes of weak keys of MAA have been identified.

The DSA algorithm (Decimal Shift and Add, not to be confused with the Digital Signature Algorithm) was designed in 1980 by Sievi of the 'German Zentralstelle für das Chiffrierwesen,' and it is used as a message authenticator for banking applications in Germany [13]. Weaknesses of this algorithm have been identified in [19, 33]. The scheme by F. Cohen [9] and its improvement by Y. Huang and F. Cohen [20] proved susceptible to an adaptive chosen message attack [34]. Attacks were also developed [33] on the weaker versions of this algorithm that are implemented in the ASP integrity toolkit [10]. Several MAC algorithms exist that have not been published, such as the S.W.I.F.T. authenticator and Dataseal [30].

4 Concluding remarks

In the design and selection of algorithms for future applications, one should take into account that the additional computation required for a strong MAC algorithm with a large key is quite modest. It is suggested to use a minimum key length of at least 90 bits, but if the application permits it, a 128-bit key is recommended.

A second observation is that provably secure schemes (in the information theoretic sense) are becoming very attractive: recently major improvements have been found both in terms of speed of computation and the key size [18, 23, 24, 29, 45]. These constructions, which follow the model of Simmons [46] (who called them authentication codes or A-codes) and Wegman-Carter [47] require a one-time use of the key. They can be reduced to a potentially efficient computationally secure scheme by generating the keys using a cryptographically strong pseudo-random string generator. For the same security level against forgery attacks, the MAC will be about twice as long. Both elements indicate that this approach will mainly be suited for high speed applications with long messages.

References

[1] ANSI X9.9 (revised), *"Financial Institution Message Authentication (Wholesale),"* American Bankers Association, April 7, 1986.

[2] ANSI X9.19 *"Financial Institution Retail Message Authentication,"* American Bankers Association, August 13, 1986.

[3] R. Atkinson, "Security architecture for the Internet Protocol," Internet Request for Comments 1825, August 1995.

[4] M. Bellare, R. Canetti, H. Krawczyk, "Pseudorandom functions revisited: The cascade construction and its concrete security," *Proc. 37th Annual Symposium on the Foundations of Computer*

[2]This section has not been discussed in the lectures.

Science, IEEE, 1996, pp. 514–523.
Full version via `http://www-cse.ucsd.edu/users/mihir`.

[5] M. Bellare, R. Canetti, H. Krawczyk, “Keying hash functions for message authentication,” *Advances in Cryptology, Proceedings Crypto’96, LNCS 1109*, N. Koblitz, Ed., Springer-Verlag, 1996, pp. 1–15.
Full version: http:// www.research.ibm.com/security/.

[6] M. Bellare, R. Guérin, P. Rogaway, “XOR MACs: new methods for message authentication using block ciphers,” *Advances in Cryptology, Proceedings Crypto’95, LNCS 963*, D. Coppersmith, Ed., Springer-Verlag, 1995, pp. 15–28.

[7] M. Bellare, J. Kilian, P. Rogaway, “The security of cipher block chaining,” *Advances in Cryptology, Proceedings Crypto’94, LNCS 839*, Y. Desmedt, Ed., Springer-Verlag, 1994, pp. 341–358.

[8] M. Blaze, W. Diffie, R.L. Rivest, B. Schneier, T. Shimomura, E. Thompson, M. Wiener, “Minimal key lengths for symmetric ciphers to provide adequate commercial security. A Report by an Ad Hoc Group of Cryptographers and Computer Scientists,” January 1996.

[9] F. Cohen, “A cryptographic checksum for integrity protection,” *Computers & Security*, Vol. 6, No. 5, 1987, pp. 505–510.

[10] F. Cohen, *“The ASP integrity toolkit. Version 3.5,”* ASP Press, Pittsburgh (PA), 1991.

[11] D. Davies, “A message authenticator algorithm suitable for a mainframe computer,” *Advances in Cryptology, Proceedings Crypto’84, LNCS 196*, G.R. Blakley and D. Chaum, Eds., Springer-Verlag, 1985, pp. 393–400.

[12] D. Davies, D.O. Clayden, “The message authenticator algorithm (MAA) and its implementation,” *NPL Report DITC 109/88*, Feb. 1988.

[13] D. Davies, W. Price, *Security for Computer Networks*, 2nd ed., Wiley, 1989.

[14] W. Diffie, M.E. Hellman, “New directions in cryptography,” *IEEE Trans. on Information Theory*, Vol. IT–22, No. 6, 1976, pp. 644–654.

[15] H. Dobbertin, A. Bosselaers, B. Preneel, “RIPEMD-160: a strengthened version of RIPEMD,” *Fast Software Encryption, LNCS 1039*, D. Gollmann, Ed., Springer-Verlag, 1996, pp. 71–82.

[16] FIPS 46, *Data encryption standard,* NBS, U.S. Department of Commerce, Washington D.C., Jan. 1977.

[17] FIPS 180-1, *Secure hash standard,* NIST, US Department of Commerce, Washington D.C., April 1995.

[18] S. Halevi, H. Krawczyk, “MMH: Software message authentication in the Gbit/second rates,” *Fast Software Encryption, LNCS 1267*, E. Biham, Ed., Springer-Verlag, 1997, pp. 172–189.

[19] F. Heider, D. Kraus, M. Welschenbach, “Some preliminary remarks on the Decimal Shift and Add algorithm (DSA),” *Abstracts Eurocrypt’86, May 20–22, 1986, Linköping, Sweden*, p. 1.2. (Full paper available from the authors.)

[20] Y.J. Huang, F. Cohen, “Some weak points of one fast cryptographic checksum algorithm and its improvement,” *Computers & Security*, Vol. 7, No. 5, 1988, pp. 503–505.

[21] ISO 8731:1987, *Banking – approved algorithms for message authentication, Part 1, DEA, Part 2, Message Authentication Algorithm (MAA).*

[22] ISO/IEC 9797:1994, *Information technology - Data cryptographic techniques - Data integrity mechanisms using a cryptographic check function employing a block cipher algorithm.*

[23] T. Johansson, “Bucket hashing with a small key size,” *Advances in Cryptology, Proceedings Eurocrypt’97, LNCS 1233*, W. Fumy, Ed., Springer-Verlag, 1997, pp. 149–162.

[24] T. Johansson, G. Kabatianskii, B. Smeets, “On the relation between A-codes and codes correcting independent errors,” *Advances in Cryptology, Proceedings Eurocrypt’93, LNCS 765*, T. Helleseth, Ed., Springer-Verlag, 1994, pp. 1–11.

[25] B. Kaliski, M. Robshaw, “Message authentication with MD5,” *CryptoBytes (RSA Laboratories Technical Newsletter)*, Vol. 1, No. 1, Spring 1995, pp. 5–8.

[26] S. Kent, “Security architecture for the Internet Protocol,” Internet Draft, July 1997.

[27] L. Knudsen, "Chosen-text attack on CBC-MAC," *Electronics Letters*, Vol. 33, No. 1, 1997, pp. 48–49.

[28] L. Knudsen, B. Preneel, "MacDES: MAC algorithm based on DES," *Electronics Letters*, Vol. 34, No. 9, 1998, pp. 871–873.

[29] H. Krawczyk, "New hash functions for message authentication," *Advances in Cryptology, Proceedings Eurocrypt'95, LNCS 921*, L.C. Guillou and J.-J. Quisquater, Eds., Springer-Verlag, 1995, pp. 301–310.

[30] C. Lindén, H. Block, "Sealing electronic money in Sweden," *Computers & Security*, Vol. 1, No. 3, 1982, p. 226–230.

[31] P. Metzger, W. Simpson, "IP Authentication using Keyed MD5", Internet Request for Comments 1828, August 1995.

[32] K. Ohta, M. Matsui, "Differential attack on message authentication codes," *Advances in Cryptology, Proceedings Crypto'93, LNCS 773*, D. Stinson, Ed., Springer-Verlag, 1994, pp. 200–211.

[33] B. Preneel, "Analysis and design of cryptographic hash functions," *Doctoral Dissertation*, Katholieke Universiteit Leuven, 1993.

[34] B. Preneel, A. Bosselaers, R. Govaerts, J. Vandewalle, "Cryptanalysis of a fast cryptographic checksum algorithm," *Computers & Security*, Vol. 9, No. 3, 1990, pp. 257–262.

[35] B. Preneel, M. Nuttin, V. Rijmen, J. Buelens, "Cryptanalysis of the CFB mode of the DES with a reduced number of rounds," *Advances in Cryptology, Proceedings Crypto'93, LNCS 773*, D. Stinson, Ed., Springer-Verlag, 1994, pp. 212–223.

[36] B. Preneel, V. Rijmen, P.C. van Oorschot, "A security analysis of the Message Authenticator Algorithm (MAA)," *European Transactions on Telecommunications*, Vol. 8, No. 5, 1997, pp. 455–470.

[37] B. Preneel, P.C. van Oorschot, "MDx-MAC and building fast MACs from hash functions," *Advances in Cryptology, Proceedings Crypto'95, LNCS 963*, D. Coppersmith, Ed., Springer-Verlag, 1995, pp. 1–14.

[38] B. Preneel, P.C. van Oorschot, "On the security of two MAC algorithms," *Advances in Cryptology, Proceedings Eurocrypt'96, LNCS 1070*, U. Maurer, Ed., Springer-Verlag, 1996, pp. 19–32.

[39] B. Preneel, P.C. van Oorschot, "A key recovery attack on the ANSI X9.19 retail MAC," *Electronics Letters*, Vol. 32, No. 17, 1996, pp. 1568–1569.

[40] B. Preneel, P.C. van Oorschot, "On the security of iterated Message Authentication Codes," *IEEE Trans. on Information Theory*, Vol. IT–45, No. 1, 1999, pp. 188–199.

[41] RIPE, *"Integrity Primitives for Secure Information Systems. Final Report of RACE Integrity Primitives Evaluation (RIPE-RACE 1040)," LNCS 1007*, A. Bosselaers and B. Preneel, Eds., Springer-Verlag, 1995.

[42] R.L. Rivest, "The MD4 message digest algorithm," *Advances in Cryptology, Proceedings Crypto'90, LNCS 537*, S. Vanstone, Ed., Springer-Verlag, 1991, pp. 303–311.

[43] R.L. Rivest, "The MD5 message-digest algorithm," Request for Comments 1321, Internet Activities Board, Internet Privacy Task Force, April 1992.

[44] R.L. Rivest, A. Shamir, L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, Vol. 21, No. 2, 1978, pp. 120–126.

[45] P. Rogaway, "Bucket hashing and its application to fast message authentication," *Advances in Cryptology, Proceedings Crypto'95, LNCS 963*, D. Coppersmith, Ed., Springer-Verlag, 1995, pp. 29–42.

[46] G.J. Simmons, "A survey of information authentication," in *"Contemporary Cryptology: The Science of Information Integrity,"* G.J. Simmons, Ed., IEEE Press, 1991, pp. 381–419.

[47] M.N. Wegman, J.L. Carter, "New hash functions and their use in authentication and set equality," *Journal of Computer and System Sciences*, Vol. 22, No. 3, 1981, pp. 265–279.

[48] M.J. Wiener, "Efficient DES key search," *Technical Report TR-244*, School of Computer Science, Carleton University, Ottawa, Canada, May 1994. Presented at the rump session of Crypto'93.