Open access • Book Chapter • DOI:10.1007/BFB0024473

Hash Functions and MAC Algorithms Based on Block Ciphers — Source link

Bart Preneel

Institutions: Katholieke Universiteit Leuven

Published on: 17 Dec 1997 - Lecture Notes in Computer Science (Springer, Berlin, Heidelberg)

Topics: Security of cryptographic hash functions, MDC-2, Hash function, Key Wrap and Cryptographic primitive

Hash functions and MAC algorithms based on block ciphers

B. Preneel[⋆]

Katholieke Universiteit Leuven, Department Electrical Engineering-ESAT,
Kardinaal Mercierlaan 94, B–3001 Heverlee, Belgium
bart.preneel@esat.kuleuven.ac.be, www.esat.kuleuven.ac.be/∼preneel

Abstract. This paper reviews constructions of hash functions and MAC algorithms based on block ciphers. It discusses the main requirements for these cryptographic primitives, motivates these constructions, and presents the state of the art of both attacks and security proofs.

1 Introduction

Hash functions and MAC algorithms are important tools to protect information integrity. Together with digital signature schemes, they play an important role for securing our electronic interactions, in applications such as electronic cash and electronic commerce. This paper discusses how to reduce the construction of these cryptographic primitives to that of another primitive, namely a block cipher. The focus is on the relation between the security and performance of the primitives.

Section 2 contains (informal) definitions of hash functions and MAC algorithms. Next a general model for these functions is presented in Sect. 3. Section 4 lists brute force attacks on hash functions, and overviews constructions for hash functions based on block ciphers. This is followed by a similar treatment of MAC algorithms in Sect. 5. Finally some concluding remarks are made.

2 Definitions

2.1 Hash functions

Hash functions were introduced in cryptography by W. Diffie and M. Hellman together with the concept of digital signatures. The basic idea is to sign a short 'digest' or 'fingerprint' of the message rather than the message itself. Hash functions are used in a more general context to replace the protection of the integrity of a large amount of data (in principle of arbitrary size) by a short string of fixed length (m bits), the hash result. Applications require that the evaluation of the hash function is 'efficient'. Moreover, the hash function should be publicly known,

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

Appeared in *Cryptography and Coding, 6th IMA International Conference*, Lecture Notes in Computer Science 1355, M. Darnell (ed.), Springer-Verlag, pp. 270–282.

and should not require any secret parameter (these hash functions have also been called MDCs or Manipulation Detection Codes). For cryptographic applications, one imposes three security requirements, that can be stated informally as follows:

preimage resistance: for essentially all outputs, it is 'computationally infeasible' to find any input hashing to that output;
2nd-preimage resistance: it is 'computationally infeasible' to find a second (distinct) input hashing to the same output as any given input;
collision resistance: it is 'computationally infeasible' to find two colliding inputs, i.e., x and $x' \neq x$ with $h(x) = h(x')$.

A hash function that is preimage resistant and 2nd-preimage resistant is called *one-way*; a hash function that satisfies the three security properties is called *collision resistant.* Collision resistance may be required for digital signatures to preclude repudiation by the sender: if she can find a collision pair (x, x'), she can later claim that she has signed x' rather than x. Not all applications need collision resistance; hash functions that are only one-way are more efficient in terms of computation and storage (cf. Sect. 4.1).

Hash functions have also been used to construct new digital signature schemes (where the hash function is an integral part of the design). Other applications include the commitment to information without revealing it, the protection of pass-phrases, tick payments, key derivation, and key agreement. It should be pointed out that hash functions have often been used in applications that require new security properties, for which they have not been evaluated.

2.2 MAC algorithms

The banking world used MACs already in the seventies. A MAC is a hash function that takes as input a second parameter, the secret key. The sender of a message appends the MAC to the information; the receiver recomputes the MAC and compares it to the transmitted version. He accepts the information as authentic only if both values are equal.

The goal is that an eavesdropper, who can modify the message, does not know how the update the MAC accordingly. Hence the main security property for a MAC is that for someone who does not know the key, it should be computationally infeasible to predict the MAC value for a given message. Unlike digital signatures, MAC algorithms can only be used between mutually trusting parties, but they are faster to compute and require less storage than digital signatures.

The attack model is as follows: an opponent can choose a number of inputs x_i, and obtain the corresponding MAC value $h_K(x_i)$ (his choice of x_i might depend on the outcome of previous queries, i.e., the attack may be adaptive). Next he has to come up with an input x ($\neq x_i$, $\forall i$) and the value $h_K(x)$, which has to be correct with probability significantly larger than $1/2^m$, with m the number of bits in the MAC result. If the opponent succeeds in finding such a value, he is said to be capable of an *existential forgery*. If the opponent can choose the value of x, he is said to be capable of a *selective forgery*. If the success probability of the attack is close to 1, the forgery is called *verifiable*. An alternative strategy is to try to recover the secret key K from a number of message/MAC pairs. A *key recovery* is more devastating than a forgery, since it allows for arbitrary selective forgeries.

A MAC is said to be secure if it is 'computationally infeasible' to perform an existential forgery under an adaptive chosen text attack. Note that in many applications only known text attacks are feasible. For example, in a wholesale banking application one could gain a substantial profit if one could choose a single message and obtain its MAC. Nevertheless, it seems prudent to work with a strong definition.

3 General model

Almost all hash functions are iterative processes, which repeatedly apply a simple compression function f. Therefore they are called *iterated* hash functions. The input x is padded to a multiple of the block size, and is divided into t blocks denoted x_1 through x_t. It is then processed block by block. The intermediate result is stored in an n-bit ($n \geq m$) *chaining variable* denoted with H_i:

$$H_0 = IV \qquad\qquad H_i = f(H_{i-1}, x_i), \quad 1 \leq i \leq t \qquad\qquad h(x) = g(H_t)\,.$$

Here g denotes the *output transformation*.

The goal of the designer is to derive the security properties of the hash function h from those of the compression function f. In order to exclude trivial attacks, it is important to fix the value of IV and to precode the message. The simplest way of coding is to append an extra block at the end with the length of the input in bits. R. Merkle and I. Damgård showed that under these conditions, collision resistance of f is sufficient for collision resistant of h [6, 23]. Note that it is not a necessary condition, i.e., collisions for the compression function do not necessarily lead to collisions for the hash function. X. Lai and J. Massey proved a similar property for (2nd) preimage resistance [20]; in this case the converse holds (in a weaker form).

Similar to hash functions, MACs are often built using a compression function. The output transformation g is in this case often more important. The secret key may be introduced in the IV, in the compression function f, and/or in the output transformation g.

4 Hash functions

This section first discusses generic attacks on hash functions, i.e., attacks that depend only on the size of the hash result, and not on the internal structure of the hash function. Then constructions of hash functions based on block ciphers are reviewed.

Appeared in *Cryptography and Coding, 6th IMA International Conference*, Lecture Notes in Computer Science 1355, M. Darnell (ed.), Springer-Verlag, pp. 270–282.

4.1 Generic attacks

Two brute force attacks on hash functions can be identified.

(2nd) preimage attack. In order to find (2nd) preimages, one can pick an arbitrary input and evaluate h. The success probability of a single trial is $1/2^m$, with m the number of bits in the hash result, which implies that on average 2^{m-1} attempts are required. This attack can be applied off-line and in parallel. If it is sufficient to find a (2nd) preimage for one out of t given values, the success probability is increased with a factor of t. This can be avoided by parameterising the hash function for each input. This attack can be precluded by choosing values of m between 64 bits (marginally secure) to 128 bits (security for 20 years or more even against a determined opponent).

collision attack. Finding collisions is much easier than finding preimages: one needs only about $\sqrt{2^m} = 2^{m/2}$ evaluations of the hash function (as this results in about 2^m pairs of hash values) [36]. This phenomenon is related to the 'birthday paradox,' which states that in a group of 23 people, the odds are 1:2 that there are two people with the same birthday. Note that it is easy to impose that the colliding message are restricted to a small set with a prescribed format, and that one can implement this attack in parallel with minimal storage requirements [34]. Collision resistance requires that m is between 128 bits (marginally secure) and 160 bits (15 years or more).

4.2 Constructions

Hash functions based on block ciphers have been popular in part for historical reasons, as designers tried to use the Data Encryption Standard (DES, [9]) also for hashing. This reduces the design and evaluation effort, and results in compact implementations. It also allows to transfer the trust in DES (or in any other block cipher) to a hash function. This is quite important since many hash functions have been broken.

However, this approach has some disadvantages. First, the use of a block cipher may create additional *export problems*. Moreover, the use of a block cipher in this application requires *stronger properties* from the block cipher. Indeed, it might be that the block cipher has certain properties that do not affect its security level for encryption, but create serious problems in hashing modes. Examples are the (semi-)weak keys of DES [7, 25] and the quasi weak keys and weak hash keys [15]. Another problem is that differential cryptanalysis can be adopted to this setting; for DES this has been explored in [32].

This construction is only *efficient* if the key scheduling of the block cipher is not too slow, as every iteration typically requires a key change. Note that current dedicated hash functions (such as RIPEMD-160 [8] and SHA-1 [10]) run at about 50 Mbit/s on a 90 MHz Pentium. DES achieves 17 Mbit/s on the same machine; other block ciphers are up to twice as fast. However, the performance including key schedule is about 2 to 10 times slower, and some constructions need more than one encryption to process a single block. As a consequence, dedicated hash

Appeared in *Cryptography and Coding, 6th IMA International Conference*, Lecture Notes in Computer Science 1355, M. Darnell (ed.), Springer-Verlag, pp. 270–282.

functions are for the time being 5. . . 10 times faster than hash functions based on block ciphers with a comparable security level.

In the rest of this section (except for Sect. 4.6) it will be assumed that the block cipher has block length and key length of m bits. The *hash rate* of a hash function based on an m-bit block cipher is defined as the number of m-bit message blocks hashed per encryption.

4.3 Single block length hash functions

For these hash functions the size of the hash result is equal to the block size of the block cipher. All these schemes have rate 1. The first cryptographic hash function was probably the construction proposed by M.O. Rabin [31]: $H_i = E_{x_i}(H_{i-1})$. Here $E_K(x)$ denotes the encryption of plaintext x using the key K. The fact that this hash function is not collision resistant when used with DES ($m = 64$) was pointed out by G. Yuval in [36]. R. Merkle showed that finding a (2nd) preimage requires only $2^{m/2}$ operations (using a meet-in-the-middle attack).

The first secure construction was the hash function of Matyas, Meyer, and Oseas: $H_i = E_{H_{i-1}}(x_i) \oplus x_i$. This scheme has been included in ISO/IEC 10118–2 [14], with an additional mapping from the ciphertext space to the key space (cf. Sect. 4.4 for an example). Its dual is known as the Davies-Meyer scheme, although it probably should be attributed to S.M. Matyas and C.H. Meyer: $H_i = E_{x_i}(H_{i-1}) \oplus H_{i-1}$. A variant of the two schemes was proposed by the author (in '89) and by Miyaguchi et al. [24]: $H_i = E_{H_{i-1}}(x_i) \oplus x_i \oplus H_{i-1}$.

In [27], the author has analyzed all 64 schemes of this general form, and came to the conclusion that 12 of these are secure. They can be constructed from the Matyas-Meyer-Oseas scheme and the Preneel-Miyaguchi scheme by applying an affine transformation to the inputs. It is conjectured that for these schemes no shortcut attacks exist, which implies that a collision attack requires about $2^{m/2}$ operations and a (2nd) preimage attack about 2^m operations. However, since most block ciphers have a block length of $m = 64$ bits, collisions can be found in only 2^{32} operations. Therefore hash functions with a larger hash result are needed.

4.4 Double block length hash functions

The aim of *double block length* hash functions is to achieve a higher security level against collision attacks. Ideally a collision attack on such a hash function should require 2^m operations, and a (2nd) preimage attack 2^{2m} operations. An important class of proposals is of the following form:

$$H_i^1 = E_{A_i^1}(B_i^1) \oplus C_i^1$$
$$H_i^2 = E_{A_i^2}(B_i^2) \oplus C_i^2 \ ,$$

where A_i^1, B_i^1, and C_i^1 are binary linear combinations of H_{i-1}^1, H_{i-1}^2, x_i^1, and x_i^2 and where A_i^2, B_i^2, and C_i^2 are binary linear combinations of H_{i-1}^1, H_{i-1}^2,

Appeared in *Cryptography and Coding, 6th IMA International Conference*, Lecture Notes in Computer Science 1355, M. Darnell (ed.), Springer-Verlag, pp. 270–282.

x_i^1, x_i^2, and H_i^1. The hash result is equal to the concatenation of H_t^1 and H_t^2. Several hash functions in this class have been published as individual proposals between '89 and '93. First it was shown by Hohl et al. that the security level of the *compression function* of these hash functions is at most that of a single block length hash function [11]. Next L. Knudsen, X. Lai, and B. Preneel showed that for all hash functions in this class, a (2nd) preimage attack requires at most 2^m operations, and a collision attack requires at most $2^{3m/4}$ operations (for most schemes this can be reduced to $2^{m/2}$) [17].

Several schemes of rate less than 1 have been proposed. From the few that have survived, the most important ones are MDC-2 and MDC-4 with hash rate 1/2 and 1/4 respectively. MDC-2 can be described as follows:

$$\begin{aligned} T_i^1 &= E_{H_{i-1}^1}(x_i) \oplus x_i = LT_i^1 \parallel RT_i^1 & \qquad H_i^1 &= LT_i^1 \parallel RT_i^2 \\ T_i^2 &= E_{H_{i-1}^2}(x_i) \oplus x_i = LT_i^2 \parallel RT_i^2 & \qquad H_i^2 &= LT_i^2 \parallel RT_i^1 \,. \end{aligned}$$

MDC-2 has been included in Part 2 of ISO/IEC 10118-2 [14]. The standard does not specify the block cipher; it also requires the specification of two mappings u, v from the ciphertext space to the key space such that $u(IV^1) \neq v(IV^2)$. For DES, these mappings from 64 to 56 bits drop the parity bits in every byte and fix the second and third key bits to 01 and 10 respectively (to preclude attacks based on (semi-)weak keys).

One iteration of MDC-4 consists of two MDC-2 steps, where the plaintexts in the second instance are equal to respectively H_{i-1}^2 and H_{i-1}^1; the keys are the same for both instances. The security level of the hash functions MDC-2 and MDC-4 (with fixed IV's) and of their compression functions is listed in Table 1. These attacks are described in [19, 20]. Note that the compression function is not very strong and that the protection of the hash function against collision attacks is not very high if DES is used.

Table 1. Security level for MDC-2 and MDC-4 based on a block cipher with equal block and key length and based on DES.

	hash function		compression function	
	collision	(2nd) preimage	collision	(2nd) preimage
MDC-2	2^m	$2^{3m/2}$	$2^{m/2}$	2^m
MDC-4	2^m	$2^{7m/4}$	$2^{3m/4}$	$2^{3m/2}$
MDC-2 (DES)	2^{55}	2^{83}	2^{28}	2^{54}
MDC-4 (DES)	2^{56}	2^{109}	2^{41}	2^{90}

4.5 Schemes with a 'security proof'

Two constructions have a 'proof of security' for the collision resistance of the compression function. They both start from the assumption that the Matyas-Meyer-Oseas single block length hash function (or one of its variants) is secure.

R. Merkle describes several constructions that achieve a security level of 2^m (2^{55} for DES) against collision attacks [23]. The fastest version has rate 0.27 for DES. The simplest scheme (with rate 1/18.3 for DES) can be described as follows:

$$H_i = \mathrm{chop}_{16}\left[E_{0\|H^1_{i-1}}(H^2_{i-1}\|x_i) \oplus (H^2_{i-1}\|x_i) \parallel E_{1\|H^1_{i-1}}(H^2_{i-1}\|x_i) \oplus (H^2_{i-1}\|x_i)\right] .$$

Here H_{i-1} is a string consisting of 112 bits, the leftmost 55 bits of which are denoted H^1_{i-1}, and the remaining 57 are denoted H^2_{i-1}; x_i consists of 7 bits only. This hash function is similar to MDC-2, but has a secure compression function at the cost of a low speed. Note that in fact additional measures have to be taken to preclude attacks based on weak keys.

The constructions proposed by L.R. Knudsen and B. Preneel [18, 19] offer better trade-offs between rate and security level; moreover the security level can be higher than 2^m. They have the disadvantage that they require a larger internal memory, and an output transformation to preclude partial collisions i.e., collisions where only part of the outputs collide. The security proof assumes that the best attack on a number of 'coupled' single block length hash functions (i.e., with dependent inputs) is a brute force attack. This coupling is achieved by encoding the input bits to the compression function with a code over $GF(2^2)$ (or, in general $GF(2^t)$, $t \geq 2$) to obtain the plaintext and key input of the individual single block length hash functions. For an $[n, k, d]$ code, the number of parallel instances of a single block length hash function equals n and a collision requires at least $2^{(d-1)/2}$ encryptions. Table 2 gives the properties of the constructions using codes over $GF(2^4)$.

Table 2. Parameters of constructions for hash functions based on codes over $GF(2^4)$.

code	rate	collision
$[6, 4, 3]$	1/4	$\geq 2^m$
$[8, 6, 3]$	1/2	$\geq 2^m$
$[12, 10, 3]$	2/3	$\geq 2^m$
$[9, 6, 4]$	1/3	$\geq 2^{3m/2}$
$[16, 13, 4]$	5/8	$\geq 2^{3m/2}$

4.6 Block ciphers with longer key lengths

R. Merkle observed already in [22] that if the key length of a block cipher is larger than the block size, it can be used as the compression function of a single block length hash function by just fixing the plaintext, and considering the mapping from key to ciphertext: $H_i = E_{H_{i-1} \,\|\, x_i}(C)$, with C a constant. X. Lai and J. Massey have extended these constructions in [20] to double block length hash functions. L. Knudsen and the author provide improved constructions with security proof for this case as well [19].

Appeared in *Cryptography and Coding, 6th IMA International Conference*, Lecture
Notes in Computer Science 1355, M. Darnell (ed.), Springer-Verlag, pp. 270–282.

5 MAC algorithms

As in the previous section, first generic attacks are discussed, and then an overview of constructions for MAC algorithms is given.

5.1 Generic attacks

The attacks discussed in this section depend only on the size of the parameters of the MAC, and not on its internal structure: brute force key search, guessing of the MAC, and a birthday forgery attack.

An **exhaustive key search** consists of running through the key space and checking whether a key corresponds to the known message-MAC pairs. About k/m values are sufficient to determine the key uniquely; for most applications this value lies between 1 and 4. The expected computational effort is 2^{k-1} MAC evaluations, where k is the number of (effective) key bits.

Brute force key search can be precluded by choosing a sufficiently large key. A value of 56 bits offers only marginal security, while 75 to 90 bits is sufficient for 15 years or more. Currently finding a 56-bit key in a period of 1 year requires an investment of 50 000$, and it can be done in a few months by using idle cycles on the Internet, as was demonstrated in the Spring of 1997; with a 1 million US$ investment, the search time can be reduced to a few hours [5, 35]. Moreover, one also has to take into account the empirical observation that the computing power for a given cost is multiplied by four every 3 years (Moore's 'Law').

It is important to recover a MAC key within its active lifetime, which can be very short in communications applications. After that period, the key is completely useless. However, one can mount the attack during the complete lifetime of the system; it is sufficient to feed to the search machine the current text/MAC pairs. In order to assess the feasibility of this attack, one has to compare the cost of an attack with the profit which can be made by recovering one or more keys during the lifetime of the system.

Another attack strategy is to pick an arbitrary input and **guess the MAC value** by choosing an m-bit string uniformly at random. For a good MAC algorithm, one expects that the success probability equals $1/2^m$ (here the probability is taken over all keys). A related strategy consists in guessing the value of the key, and computing the MAC value. Its success probability is $1/2^k$. The success probability of the combined attacks is equal to $1/2^{\min(k,m)}$. The value of k is typically larger than that of m to preclude a brute force key search. This forgery attack is clearly not verifiable. However, one should take it into account when selecting a MAC algorithm. The value of m depends on the expected profit of a successful attempt, as well as on the number of trials that are allowed by the system, i.e., the way the system reacts to wrong MAC values. For most applications $m = 32 \ldots 64$ is sufficient to render this attack uneconomical.

Recently a **birthday forgery attack** has been developed that applies to all iterated MAC algorithms (B. Preneel and P.C. van Oorschot, [29]). The basic idea behind the attack is the birthday paradox. Its feasibility depends on the bitsizes n of the chaining variable and m of the MAC result, and the nature

Appeared in *Cryptography and Coding, 6th IMA International Conference*, Lecture
Notes in Computer Science 1355, M. Darnell (ed.), Springer-Verlag, pp. 270–282.

of the output transformation g. If $n = m$, and g is a permutation, the attack requires one chosen text and about $2^{n/2}$ known message-MAC pairs. If $n > m$, an additional number of about 2^{n-m} chosen message-MAC pairs is required. The attack requires at least one chosen text, hence in applications where any access to the MAC algorithm with the correct key is precluded, it should not be considered a problem. Also, the forged message is of a special form: if the MAC is known for three messages of the form x, x', and $x\|y$, one can forge the MAC for $x'\|y$. Note that further optimisations of the attack are possible, that reduce the number of known texts. A more serious concern is that for certain MAC algorithms, the forgery attack can be extended to a key recovery attack.

This attack motivates the use of a MAC result that is smaller than the size of the internal memory n. It can be precluded by appending a sequence number at the *end* of every message. Such a sequence number if useful to prevent replay attacks as well [7]. An alternative is to randomise the output transformation.

5.2 Constructions for MACs

MAC algorithms based on block ciphers have been proposed already in the late seventies. Together with DES, **CBC-MAC** based on DES became very popular, especially in the banking world. It has been standardized in several documents comprising ANSI X9.9 [1], ANSI X9.19 [2], ISO 8731 [12], and ISO/IEC 9797 [13]. Only during the last years, a thorough security analysis has been performed. The compression function of CBC-MAC has the following form:

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

with $H_0 = 0$. An output transformation is needed to preclude the following simple forgery: given $h_K(x)$, $h_K(x\|y)$, and $h_K(x')$, one knows that $h_K(x'\|y') = h_K(x\|y)$ if $y' = y \oplus h_K(x) \oplus h_K(x')$.

One approach is for g to select the leftmost m bits. However, L.R. Knudsen has shown that the simple attack can be extended to this case [16]. It requires then approximately $2^{(n-m)/2}$ chosen texts and 2 known texts.

A stronger and widely used 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 ANSI X9.19:

$$g(H_t) = E_{K_1}(D_{K_2}(H_t))\,.$$

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.

All these variants are vulnerable to the birthday forgery attack, that requires a single chosen message and about 2^{32} known messages (if DES is used with $m = 64$). If $m = 32$, an additional 2^{32} chosen messages are required. Note that with a fast DES implementation on a PC, this number of texts can be collected in a single day. Bellare et al. provide a proof of security for CBC-MAC [3], i.e., they establish a lower bound to break the system under certain assumptions on the block cipher. It almost matches the upper bound provided by the birthday forgery attack.

For the ANSI retail MAC, 2^{32} known texts allow for a key recovery requiring $3 \cdot 2^k$ encryptions, compared to 2^{2k} encryptions for exhaustive search [30]. 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 $\min(m, 2k)$ to $\min(m, k)$. It seems possible to preclude these key recovery attacks by introducing also a triple-DES encryption in the first iteration, but further research is necessary to confirm this.

An alternative to CBC-MAC is **RIPE-MAC**, that adds a feedforward [33]:

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

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 [4]. 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 parallellisable and that it is incremental, i.e., 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 30% to 50% slower than CBC-MAC.

Note that cryptanalysis of the underlying block cipher can often be extended to an attack on the MAC, in spite of the fact that an attacker obtains less information than in conventional cryptanalysis on the ECB mode. Examples can be found in the literature for linear and differential cryptanalysis of CBC-MAC based on DES [26, 28].

6 Concluding remarks

The construction of a new cryptographic primitive (hash function, MAC algorithm) based on an existing primitive (a block cipher) is a common paradigm in cryptography. This requires a very good understanding of the security of both primitives, and of additional assumptions that have to be made. For MAC algorithm, significant progress has been made, both from the side of the attacks (upper bound on the security level) and from the side of the constructions (lower bound on the security level based on certain assumptions). For hash functions, the cryptanalytic side has achieved some successes, and has resulted in new constructions with some provable properties. However, important open problems remain such as the formalization of the security of single block length hash functions.

It is likely that DES will be replaced in a few years by the AES (with a 128-bit block size). It will then be possible to achieve a higher security level using the existing constructions. However, it is an interesting open problem to improve the security and performance of constructions that use the DES or other 64-bit block ciphers.

Appeared in *Cryptography and Coding, 6th IMA International Conference*, Lecture Notes in Computer Science 1355, M. Darnell (ed.), Springer-Verlag, pp. 270–282.

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. M. Bellare, J. Kilian, P. Rogaway, "The security of cipher block chaining," *Advances in Cryptology, Proc. Crypto'94, LNCS 839*, Y. Desmedt, Ed., Springer-Verlag, 1994, pp. 341–358.
4. M. Bellare, R. Guérin, P. Rogaway, "XOR MACs: new methods for message authentication using block ciphers," *Advances in Cryptology, Proc. Crypto'95, LNCS 963*, D. Coppersmith, Ed., Springer-Verlag, 1995, pp. 15–28.
5. 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.
6. I.B. Damgård, "A design principle for hash functions," *Advances in Cryptology, Proc. Crypto'89, LNCS 435*, G. Brassard, Ed., Springer-Verlag, 1990, pp. 416–427.
7. D. Davies, W. Price, *Security for Computer Networks*, 2nd ed., Wiley, 1989.
8. 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.
9. FIPS 46, *Data encryption standard,* NBS, U.S. Department of Commerce, Washington D.C., Jan. 1977.
10. FIPS 180-1, *Secure hash standard,* NIST, US Department of Commerce, Washington D.C., April 1995.
11. W. Hohl, X. Lai, T. Meier, C. Waldvogel, "Security of iterated hash functions based on block ciphers," *Advances in Cryptology, Proc. Crypto'93, LNCS 773*, D. Stinson, Ed., Springer-Verlag, 1994, pp. 379–390.
12. ISO 8731:1987, *Banking – approved algorithms for message authentication, Part 1, DEA, Part 2, Message Authentication Algorithm (MAA).*
13. ISO/IEC 9797:1993, *Information technology - Data cryptographic techniques - Data integrity mechanisms using a cryptographic check function employing a block cipher algorithm.*
14. ISO/IEC 10118:1994, *"Information technology – Security techniques – Hash-functions, Part 1: General and Part 2: Hash-functions using an n-bit block cipher algorithm,".*
15. L.R. Knudsen, "New potentially 'weak' keys for DES and LOKI," *Advances in Cryptology, Proc. Eurocrypt'94, LNCS 950*, A. De Santis, Ed., Springer-Verlag, 1995, pp. 419–424.
16. L.R. Knudsen, "Chosen-text attack on CBC-MAC," *Electronics Letters*, Vol. 33, No. 1, 1997, pp. 48–49.
17. L.R. Knudsen, X. Lai, B. Preneel, "Attacks on fast double block length hash functions," *Journal of Cryptology*, in print.
18. L.R. Knudsen, B. Preneel, "Hash functions based on block ciphers and quaternary codes," *Advances in Cryptology, Proc. Asiacrypt'96, LNCS 1163*, K. Kim and T. Matsumoto, Eds., Springer-Verlag, 1996, pp. 77–90.
19. L.R. Knudsen, B. Preneel, "Fast and secure hashing based on codes," *Advances in Cryptology, Proc. Crypto'97, LNCS 1294*, B. Kaliski, Ed., Springer-Verlag, 1997, pp. 485–498.

Appeared in *Cryptography and Coding, 6th IMA International Conference*, Lecture Notes in Computer Science 1355, M. Darnell (ed.), Springer-Verlag, pp. 270–282.

20. X. Lai, J.L. Massey, “Hash functions based on block ciphers,” *Advances in Cryptology, Proc. Eurocrypt’92, LNCS 658*, R.A. Rueppel, Ed., Springer-Verlag, 1993, pp. 55–70.
21. S.M. Matyas, C.H. Meyer, J. Oseas, “Generating strong one-way functions with cryptographic algorithm,” *IBM Techn. Disclosure Bull.*, Vol. 27, No. 10A, 1985, pp. 5658–5659.
22. R. Merkle, *“Secrecy, Authentication, and Public Key Systems,”* UMI Research Press, 1979.
23. R. Merkle, “One way hash functions and DES,” *Advances in Cryptology, Proc. Crypto’89, LNCS 435*, G. Brassard, Ed., Springer-Verlag, 1990, pp. 428–446.
24. S. Miyaguchi, M. Iwata, K. Ohta, “New 128-bit hash function,” *Proc. 4th International Joint Workshop on Computer Communications*, Tokyo, Japan, July 13–15, 1989, pp. 279–288.
25. J.H. Moore, G.J. Simmons, “Cycle structure of the DES for keys having palindromic (or antipalindromic) sequences of round keys,” *IEEE Trans. on Software Engineering*, Vol. SE–13, No. 2, 1987, pp. 262–273.
26. K. Ohta, M. Matsui, “Differential attack on message authentication codes,” *Advances in Cryptology, Proc. Crypto’93, LNCS 773*, D. Stinson, Ed., Springer-Verlag, 1994, pp. 200–211.
27. B. Preneel, R. Govaerts, J. Vandewalle, “Hash functions based on block ciphers: a synthetic approach,” *Advances in Cryptology, Proc. Crypto’93, LNCS 773*, D. Stinson, Ed., Springer-Verlag, 1994, pp. 369–379.
28. 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, Proc. Crypto’93, LNCS 773*, D. Stinson, Ed., Springer-Verlag, 1994, pp. 212–223.
29. B. Preneel, P.C. van Oorschot, “MDx-MAC and building fast MACs from hash functions,” *Advances in Cryptology, Proc. Crypto’95, LNCS 963*, D. Coppersmith, Ed., Springer-Verlag, 1995, pp. 1–14.
30. 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.
31. M.O. Rabin, “Digitalized signatures,” in *“Foundations of Secure Computation,”* R. Lipton, R. DeMillo, Eds., Academic Press, New York, 1978, pp. 155–166.
32. V. Rijmen, B. Preneel, “Improved characteristics for differential cryptanalysis of hash functions based on block ciphers,” *Fast Software Encryption, LNCS 1008*, B. Preneel, Ed., Springer-Verlag, 1995, pp. 242–248.
33. 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.
34. P.C. van Oorschot, M.J. Wiener, “Parallel collision search with application to hash functions and discrete logarithms,” *Proc. 2nd ACM Conference on Computer and Communications Security*, ACM, 1994, pp. 210–218. Final version to appear in Journal of Cryptology.
35. 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.
36. G. Yuval, “How to swindle Rabin,” *Cryptologia*, Vol. 3, No. 3, 1979, pp. 187–189.

Appeared in *Cryptography and Coding, 6th IMA International Conference*, Lecture Notes in Computer Science 1355, M. Darnell (ed.), Springer-Verlag, pp. 270–282.