On the Security of Iterated Message Authentication Codes*

Bart Preneel[†], *Member, IEEE,* and Paul C. van Oorschot[‡]

May 29, 1998

Abstract

The security of iterated message authentication code (MAC) algorithms is considered, and in particular those constructed from unkeyed hash functions. A new MAC forgery attack applicable to all deterministic iterated MAC algorithms is presented, which requires on the order of $2^{n/2}$ known text-MAC pairs for algorithms with n bits of internal memory, as compared to the best previous general attack which required exhaustive key search. A related key recovery attack is also given which applies to a large class of MAC algorithms including a strengthened version of CBC-MAC found in ANSI X9.19 and ISO/IEC 9797, and envelope MAC techniques such as "keyed MD5". The security of several related existing MACs based directly on unkeyed hash functions, including the secret prefix and secret suffix methods, is also examined.

Index terms—Message Authentication Codes, data authentication, cryptanalysis, hash functions, collisions

*The material in this paper was presented in part at Crypto'95, Santa Barbara, CA, USA, August 27–31, 1995 and Eurocrypt'96, Zaragoza, Spain, May 12–16, 1996. Bart Preneel is an F.W.O. postdoctoral researcher, supported by the Fund for Scientific Research — Flanders (Belgium).

[†]Bart Preneel is with the Department of Electrical Engineering-ESAT, Katholieke Universiteit Leuven, Kardinaal Mercierlaan 94, B–3001 Heverlee, Belgium. Email: `bart.preneel@esat.kuleuven.ac.be`.

[‡]Paul C. van Oorschot is with Entrust Technologies, 750 Heron Road, Suite E08, Ottawa, Canada, K1V 1A7.

Contents

I. Introduction

Message authentication code (MAC) algorithms have received widespread use for data integrity and data origin authentication, e.g. in banking applications [15, 31]. They are closely related to hash functions, which play a fundamental role in many areas of modern cryptography, including a primary role in conjunction with digital signature algorithms. When combined with a secret key, hash functions may provide conventional techniques for message authentication; in this case it is preferable that the secret key be a distinct secondary input.

Relative to the extensive work on the design and analysis of hash functions [40], little attention has been given to the design of efficient MACs until recently [6, 7, 8, 41, 48]. This evidently resulted from the adoption of several early MAC proposals as standards which proved adequate in practice, including constructions based on the Cipher Block Chaining (CBC) and Cipher FeedBack (CFB) modes of a block cipher [1, 2, 28, 29]. By far the most common is the CBC mode (CBC-MAC), theoretical support for which was recently given [5]. Another early proposal dating back to 1983, the Message Authenticator Algorithm (MAA) [13, 14, 28], is a current ISO standard for which weaknesses of concern have only recently been raised [41, 42, 44]. MAA is relatively fast in software (about 40 per cent slower than MD5); its primary disadvantage historically was that the 32-bit result is considered unacceptably short for many applications. Recent research on authentication codes has resulted in very fast, scalable, and so-called unconditionally secure constructions [30, 34, 51], which require relatively short keys, and also [48] (see [4] for a summary of these an other schemes); their disadvantage is that a different key must be used for every message. If this is not acceptable, one can generate the key using a cryptographically strong pseudo-random string generator, but the resulting scheme is then (at most) computationally secure.

In the early 1990's, Rivest proposed two very fast hash functions: MD4 [46] and MD5 [47]. Other hash functions based on these were subsequently introduced, including RIPEMD [45], SHA-1 [24], and RIPEMD-160 [20]. In software, these hash functions have throughput as much as one order of magnitude higher than DES [22]. Several factors motivated their adoption as the basis for MAC algorithms: the additional implementation and deployment effort required to adopt these as MACs is minimal (code for the underlying unkeyed hash function can be called directly); MACs based on these outperform most other available options; and such MACs, avoiding the use of encryption algorithms, may have preferential export status. Consequently, MAC constructions based on these hash functions were adopted in Kerberos [35], SNMP [26], and SSL [25], and gained favor in the IPsec working group of the IETF [32, 38].

The current paper presents a new cryptanalytic method applicable to all deterministic iterated MACs, including MAA and CBC-MAC. The technique involves finding collisions on known text-MAC pairs after which a few additional chosen text-MAC pairs allow MAC forgery. The attack requires $\Theta(2^{n/2})$ known text-MAC pairs (here n is the bitlength of the MAC's internal memory, i.e. chaining variable), whereas the best previous general attack on MAC algorithms was an exhaustive key search. An extension of the new technique, also described, provides a more powerful attack in special cases. Three existing proposals for MAC algorithms based on hash functions are then analyzed: the secret prefix method, the secret suffix method, and the envelope method combining these. For the secret prefix and suffix methods, a systematic analysis is given which generalizes the known attacks. For the envelope method, application of the new general attack illustrates that earlier arguments [49] regarding the security level of this method, which claimed that security was exponential in the sum of the lengths of the secret keys used, are incorrect. Finally, variants of these methods are shown to be susceptible to more serious attacks which actually allow key recovery. These variants include keyed MD5 as specified per Internet RFC 1828 [38] and the strengthened CBC-MAC (hereafter CBC-MAC-Y) included in ANSI X9.19 [2] and ISO/IEC 9797 [29].

The conclusion is that many approaches which construct MACs directly from hash functions (particularly those which use the hash function as a "black box" without modifying its external interface, and involving only a single call thereof), achieve a security level significantly less than that suggested by the size of their parameters. In addition, the new attack calls into question the strength of MAA and CBC-MAC, including CBC-MAC-Y.

The sequel is organized as follows. Section II provides background definitions and reviews several existing MAC proposals. Section III presents a new general forgery attack on MACs, and an extension thereto. Section IV analyzes the security of various MAC proposals, taking into account this forgery attack. Section V presents three key recovery attacks, the first specifically on CBC-MAC-Y, and the latter two applicable to many envelope-type MAC constructions; all three are based in part on the central idea of the new forgery attack. Section VI provides a partial summary, and concluding remarks are given in Section VII.

II. Definitions and Background

A hash function h maps bitstrings of arbitrary finite length into strings of fixed length. Given h and an input x, computing $h(x)$ must be easy.[1]

First one may define hash functions not involving secret parameters, as follows. A *one-way hash function* must provide two properties: *preimage resistance* (it must computationally infeasible to find any input which hashes to any pre-specified output), and *second preimage resistance* (it must be computationally infeasible to find any second input which has the same output as any specified input). For an *ideal* one-way hash function with m-bit result, finding a preimage or a second preimage requires expected $\Theta(2^m)$ operations. A *collision resistant hash function* is a one-way hash function that provides the additional property of *collision resistance* (it must be computationally infeasible to find a collision, i.e. two distinct inputs that hash to the same result). For an *ideal* collision resistant hash function with m-bit result, no attack finding a collision requires less work than a birthday or square root attack of $\Theta(2^{m/2})$ operations.

A MAC is a hash function with a secondary input, the secret key K. Given h, an input x, and the secret key K, computing $h(x)$ must be easy; note K here is assumed to be an implicit parameter of $h(x)$. The strongest condition one may impose on a MAC is as follows: without knowledge of the secret key, it must be computationally infeasible to

[1]The intention of this section is to provide background to allow an understanding of subsequent results. Consequently, formality is sacrificed where informality aids understandability.

perform an *existential forgery*, i.e. to find any random message and its corresponding MAC. In contrast, for a *selective forgery*, an opponent is required to determine the MAC for a message of their own choosing. For a practical attack, one often requires that the forgery is *verifiable*, i.e. that the MAC is known to be correct (or correct with probability very close to 1). Here it is assumed that the opponent is capable of performing a *chosen text attack*, i.e. may obtain MACs corresponding to a number of messages of their choice. A stronger notion is that of *adaptive* chosen text attack, in which an opponent's requests may depend on the outcome of previous requests. To be meaningful, a forgery must be for a message different than any for which a MAC was previously obtained.

For an *ideal* MAC with m-bit result, any method to find the k-bit key is as expensive as an exhaustive search of expected $\Theta(2^k)$ operations. For a MAC which behaves as a random mapping (which is optimal), the expected number of text-MAC pairs required for verification of such an attack is approximately $\lceil k/m \rceil$. An opponent who has identified the correct key can compute the MAC for any message; that is, key recovery allows selective forgery. If the opponent knows no text-MAC pairs, or if $m < k$, his best strategy may be to simply guess the MAC corresponding to a chosen message; the probability of success is $1/2^m$. The disadvantage of a guessing attack is that it is not verifiable. A further desirable property of an ideal MAC is that finding a second preimage should require expected $\Theta(2^m)$ known text-MAC pairs. In some settings (e.g. multi-destination electronic mail [39]) it may be desirable that this requires expected $\Theta(2^m)$ off-line MAC computations even for someone with knowledge of the key.

Most hash functions h, and most MACs, are iterative processes which hash inputs of arbitrary length by processing successive fixed-size b-bit blocks of the input x, divided into t blocks of b bits each, x_1 through x_t. If the bitlength of x is not a multiple of b, x is padded using an unambiguous padding rule. h can then be described as:

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

Here f is the *compression function* of h, and H_i is the n-bit *chaining variable*, $n \geq m$, resulting after stage i. IV is short for *initial value*, which is a fixed constant. In the case of a MAC, one often applies an *output transformation* g to H_t, yielding the m-bit MAC result $h(x) = g(H_t)$. In the simplest case, g is the identity mapping. The secret key may be introduced in the IV, in the compression function f, and in the output transformation g.

CBC-MAC, standardized in ANSI X9.9 [1], ANSI X9.19 [2], ISO 8731-1 [28], and ISO/IEC 9797 [29], is based on a block cipher; it is defined as follows:

$$f(H_{i-1}, x_i) = E_K(H_{i-1} \oplus x_i)\,,$$

where $E_K(x)$ denotes the encryption of x with key K and $H_0 = 0$. Let $\|$ denote concatenation. This MAC requires an output transformation g to preclude the following (existential forgery) attack: given MAC(x), MAC($x\|y$), and MAC(x'), one knows that MAC($x'\|y'$) = MAC($x\|y$) if $y' = y \oplus \text{MAC}(x) \oplus \text{MAC}(x')$. One commonly used approach is for g to select the leftmost m bits; Knudsen showed recently that this is less secure than anticipated [33]. Alternatives are to replace processing of the last block by a two-key triple encryption (CBC-MAC-Y); or to define g as the encryption of H_t with a derived key (e.g. complement every other half-byte of K) [28, 29, 45].

Several MACs are similar to CBC-MAC. CFB-MAC [1] uses $f(H_{i-1}, x_i) = E_K(H_{i-1}) \oplus x_i$. RIPEMAC [45] uses $f(H_{i-1}, x_i) = E_K(H_{i-1} \oplus x_i) \oplus x_i$. For reasons as above, both also require an output transformation.

Many MAC algorithms derived from efficient hash functions have been proposed, including the secret prefix method, the secret suffix method, and several variants of the secret envelope method. The *secret prefix* method consists of prepending a secret key K_1 to the message x before the hashing operation: MAC(x) = $h(K_1\|x)$ for h an unkeyed hash function. If the key (possibly padded) consists of a complete block, this corresponds to a hash function with a secret IV. This method was suggested for MD4 independently by Tsudik [49] and by the Internet Security and Privacy Working Group for use in the Simple Network Management Protocol (SNMP) [26]. In the 1980s this was already proposed for at least two other hash functions (for example [11]). The *secret suffix method* specifies that a secret key K_2 be appended to the message: MAC(x) = $h(x\|K_2)$. This construction was proposed for SNMP (see Galvin et al. [26]).

The *envelope method* [49] combines the prefix and suffix methods. One prepends a secret key K_1 and appends a secret key K_2 to the message input: $\mathrm{MAC}(x) = h(K_1\|x\|K_2)$. Arising from the IP Security (IPsec) working group of the IETF, RFC 1828 [38], a proposed standard for authentication of IP (Internet Protocol) datagrams, specifies a variant of this method using MD5 and a single key K: $\mathrm{MAC}(x) = h(K\|p\|x\|K)$ (see also [32]). Here p denotes some padding bits chosen such that $K\|p$ fills the first 512-bit block (here $b = 512$). RFC 1828 allows a variable length key, but mandates support for bitlengths up to 128 bits. RFC 1828 was used in version 2.0 of the Secure Sockets Layer standard (SSL 2.0). Although it may be supported by a security proof under assumptions regarding the pseudo-randomness of MD5 [7], Section V demonstrates that this scheme is vulnerable to a key recovery attack if the unkeyed hash function h has a padding mechanism such as that of MD5.

The approach of the envelope method was taken one step further in the construction called *MDx*-MAC [41]. Important design elements are that three subkeys are derived from the input key, one of which is involved in every iteration of the compression function. This offers better protection against possible undiscovered weaknesses of the underlying hash function, and imposes less demanding requirements on the compression function (cf. Section V.D), without affecting the throughput. It is further recommended in *MDx*-MAC to keep only half of the output bits (a 64-bit result in case of MD5); the forgery attack of Section III then requires a large number of chosen texts (aside from known texts). Finally, to preclude a key recovery attack (Section V.B), the trailing subkey in *MDx*-MAC is positioned in a separate block.

An alternate approach to the construction of MACs is given by Bellare et al. [6]: here a provably secure construction is presented based on a finite pseudo-random function, which can be instantiated with a block cipher or a hash function. The scheme has the additional advantage that it is parallelizable and incremental. More recently, Bellare et al. [8] have rigorously examined the security of the construction named HMAC, defined as $\mathrm{MAC}(x) = h((K \oplus p_2)\|h((K \oplus p_1)\|x))$ and proven it is secure provided that $h(.)$ is collision resistant for random and secret IV, and that the complete output of the compression function f is hard to predict when its first input is random and secret. Before applying the strings p_1 and p_2 (with $p_1 \neq p_2$), one pads K out with zeroes to a full b-bit block. Despite two calls to h, the second is on a short (e.g. two-block) input and thus overall the construction remains quite efficient. This scheme has been included in the informational Internet RFC 2104 [9]. An earlier version of this scheme (without padding) was proposed in the note of Kaliski and Robshaw [32], with the option to choose $K_2 = K_1$. Another variant (proposed in a draft of [8] and included in SSL 3.0) used the strings p_1 and p_2 to pad out K to a b-bit block.

III. General Forgery Attack on MAC Algorithms

A new attack applicable to all (deterministic) iterated MACs is described here. The parameters (running time and text requirements) depend only on the bitsize n of the chaining variables and on the bitsize m of the hash result. The attacks are probabilistic, but the attackers can verify whether or not it will succeed; moreover, the success probability grows quadratically with the number of texts, which implies that it is very easy to make it arbitrarily close to 1. Initially (Lemma 1 and Propositions 1 and 2), no assumptions are made about the texts being hashed. Subsequently, an optimization is given for the case that texts have a common sequence of s trailing blocks (Proposition 3). Then the implications on CBC-MAC and MAA are discussed.

A. Basic Results

Propositions 1 and 2 below are facilitated by the following definitions and Lemma 1. Let g be the output transformation as defined above. Let (x, x') be a pair of message inputs with $h(x) = g(H_t)$ and $h(x') = g(H'_t)$.

Definition 1 *A* chaining variable collision *is said to occur when for some $i \leq t$, $H_i = H'_i$, i.e. the intermediate chaining values coincide. If the messages x and x' differ in their respective remaining portions subsequent to a chaining variable collision, then in general H_t and H'_t will differ. An* internal collision *is said to occur when a chaining variable collision* *results in the situation where $H_t = H'_t$; this may happen for example when the remaining message portions following a chaining variable collision are identical. In the following it will be assumed that g is deterministic (i.e. involves no randomization, but rather is a deterministic function of its inputs); an internal collision then yields a MAC collision. If $H_t \neq H'_t$ but $g(H_t) = g(H'_t)$, then an* external collision *is said to have occurred.*

As indicated above, the initial value, compression function, and output transformation can depend on the secret key. If key bits are included in the first message blocks, this corresponds to keying the initial value; while it is not exactly the same, for the purpose of our analysis this will be considered to be equivalent [7]. If the message formatting or padding results in the inclusion of key bits in the final message input blocks to be processed, this opens the question where the output transformation g starts. For subsequent analysis (and reasons to become apparent in the sequel), the output transformation g is defined to begin with the first block i which contains (any partial) keying material in the message input x_i.

Lemma 1 *An internal collision for an iterated MAC algorithm can be used to obtain a verifiable MAC forgery with a chosen text attack requiring only one requested MAC.*

Proof: For an internal collision (x, x'), note that

$$h(x \,\|\, y) = h(x' \,\|\, y) \tag{1}$$

for any single block y. Thus requesting a MAC for the single chosen text $x \,\|\, y$, permits forgery – the MAC for $x' \,\|\, y$ is the same (since the MAC algorithm is deterministic). This assumes that both x and x' fill entire b-bit blocks; otherwise the padding has to be taken into account. ■

The observation of Lemma 1, made in a conference paper [41] submitted in February 1995, has also appeared in a Spring 1995 note [32].

It follows that a security requirement for MACs is that it should be infeasible for an adversary to find internal collisions. This is somewhat analogous to collision resistance for hash functions. The attack of Lemma 1 can be precluded by making the output transformation g different for each MAC calculation, e.g., by including a sequence number or a sufficiently large random number in the computation of g. For example, the MD5 padding method could be augmented by including immediately prior to the length field in the final-block padding an appropriately-sized random bit field. Also, appending a length field within the output transformation would impose the restriction that the messages x and x' used in the attack are of the same length.

In the remainder of this paper, it will be assumed that the output transformation $g(.)$ and the compression function f (i.e. both $f(., x_i)$ for fixed x_i and $f(H_{i-1}, .)$ for fixed H_{i-1}) are either random permutations or random functions, but in both cases, deterministic once they have been chosen. In the first case, this means that they are chosen with uniform probability among the set of all permutations on the domain D. In the latter case, this means that they are chosen with uniform probability among the set of all functions mapping their domain D to their range R; the main property required of f is that two different inputs will have colliding outputs with probability close to $1/|R|$ (here $|R|$ denotes the size of R). If this probability is significantly larger for some elements in the domain, this can usually be exploited by the cryptanalyst to improve the attacks described here; in one case this is a disadvantage to him (this exception is clarified below).

Proposition 1 *Let h 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.*

Proof: Since g is a permutation (e.g. the identity mapping), there are no external collisions. For $u = \sqrt{2} \cdot 2^{n/2}$, a single internal collision is expected by the birthday paradox since $\binom{u}{2}/2^n \approx 1$. (See Note 1 for clarification of this statement.) Therefore the result follows by Lemma 1. More precisely, the number c of internal collisions is a Poisson distributed random variable with parameter $\lambda = u^2/2^{n+1} = 1$ [21, p. 59],[27], or

$$\Pr(c = a) = e^{-\lambda} \cdot \frac{\lambda^a}{a!}, \qquad a \geq 0.$$

If the number of internal collisions is at least 1, the attack (of Lemma 1) succeeds. It fails only if there is no internal collision, which happens with probability $\Pr(c = 0) = \exp(-\lambda)$ $= 1/e$. Such a failure can be easily detected, because all u MAC pairs are distinct in this case. Moreover, the failure probability of the attack decreases exponentially when u increases: if $u = \gamma \cdot \sqrt{2} \cdot 2^{n/2}$, the failure probability becomes $\exp(-\gamma^2)$. For example, $\gamma = 2$ yields a failure probability of $1/e^4$, and $\gamma = 4$ results in a failure probability of $1/e^{16} \approx 1.1 \cdot 10^{-7}$. ∎

Note 1 (see proof of Proposition 1): If for fixed H_{i-1}, $f(H_{i-1}, .)$ is a random function, the standard assumption for the birthday paradox is satisfied, and the statement follows. The situation is more complex if for fixed H_{i-1}, $f(H_{i-1}, .)$ is a permutation (this implies $b = n$). For simplicity assume the number of input blocks is $t = 2$; the argument can be extended to the case $t > 2$. In this case no collisions can be obtained immediately after the first iteration. However, it is clear that the mapping $f(f(IV, x_1^i), x_2^i)$ is a random function for fixed IV provided that the message blocks (x_1^i, x_2^i) have the properties that the x_1^i are all distinct and the x_2^i are all distinct. An internal collision can then occur anywhere after the second block. As a consequence, the restriction in Proposition 1 (and Proposition 2 below) that $t \geq 2$ can be removed if for fixed H_{i-1}, $f(H_{i-1}, .)$ is a random function.

Proposition 2 *Let h 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*

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

Proof: The distribution of internal collisions is identical to that for the proof of Proposition 1, which implies that again a single internal collision is expected. As g is a random function from n-bit to m-bit strings, the number of external collisions expected is

$t_1 = \binom{u}{2}/2^m \approx u^2/2^{m+1} = 2^{n-m}$. (If g results in more (less) collisions than a random mapping, a larger (smaller) number of external collisions will occur; the attack remains the same in principle, but the number of chosen texts required varies as will become clear.) Additional work is now required – for a verifiable forgery – to distinguish the internal collision from the external collisions (since Lemma 1 requires internal collisions). This may be done by appending a string y to both elements of each collision pair and checking whether the corresponding MACs are equal, requiring $2(t_1 + 1)$ chosen text-MAC requests. For an internal collision both results will always be equal; for an external collision this will be so with probability $1/2^m$ in the case that f is a permutation for fixed x_i. On the other hand, if f is a random mapping for fixed x_i, the probability that both MACs are equal is

$$1 - \left(1 - \frac{1}{2^n}\right)^q \cdot \left(1 - \frac{1}{2^m}\right) \tag{3}$$

where $q \geq 1$ is the number of blocks in y. Assume $q = 1$, in which case $2/2^m = 2^{-m+1}$ is an upper bound on the probability (3); this upper bound will be used for the remainder of the proof. The attacker now discards collision pairs corresponding to unequal MACs. The expectation is that after this stage, at most $t_2 = 2^{n-2m+1}$ external collision pairs plus the one internal collision pair remain. These however cannot yet be distinguished if the (total) number of remaining collision pairs is 2 or more. More generally, if $n - 2m + 1 > 0$, further external collisions must be filtered by appending a different y, and continuing with further filtering stages until only a single collision remains ($t_i \approx 0$), which with high probability is an internal collision. The expected number of filtering stages required is defined by the smallest α such that $(2^{1-m})^\alpha \cdot t_1 < 1$. Thus

$$\alpha = \left\lfloor \frac{n-m}{m-1} \right\rfloor + 1 = \left\lfloor \frac{n-1}{m-1} \right\rfloor . \tag{4}$$

If t_i is the total number of external collision pairs remaining at stage i, then the total number v of chosen texts required is $\sum_{i=0}^{\alpha-1} 2(t_i + 1) = 2\alpha + 2 \cdot \sum_{i=0}^{\alpha-1} t_i$, which is

$$2\alpha + 2 \cdot 2^{n-m} + 2 \cdot 2^{n-m} \cdot \sum_{i=1}^{\alpha-1} (2/2^m)^i \;\approx\; 2\alpha + 2 \cdot 2^{n-m} \tag{5}$$

for $m \gg 1$. The proof is completed by noting that instead of working stage by stage, one can eliminate the collision pairs one by one; if a pair survives after α stages, it is declared to be an internal collision. As a consequence, not all of the 2^{n-m} external collisions need be processed: if there are j internal collisions (an event with probability $\Pr(c = j) = e^{-1}/j!$), about 1 in $j + 1$ external collisions must be processed before one expects to find a first internal collision. The expected number which must be eliminated through processing can then be approximated by

$$2^{n-m} \cdot \sum_{j=0}^{\infty} \frac{e^{-1}}{j!} \cdot \frac{1}{j+1} = 2^{n-m} \left(1 - \frac{1}{e}\right) .$$

Note that if f is a permutation for fixed x_i, one obtains the same expression (5) but with $\alpha = \left\lfloor \frac{n}{m} \right\rfloor$, which gives the same results when n is a multiple of m (a common case in practice). ■

Note 2: To confirm that the calculations in the proof of Proposition 2 were good approximations, exact calculations were made for $m = n$ and $m = 2n$ (the most important cases in practice). For $m = n$, one can verify that the stated value of v is obtained when one takes into account that the number of external collisions is Poisson distributed with mean value 1 as well. The probability of a detected failure (no internal collisions) is $\exp(-1 - 2/2^m)$; it can be reduced by increasing u as noted in the proof of Proposition 1. An undetected failure occurs when an external collision is declared to be an internal collision; in this case the failure will only be detected after trying to apply the attack of Lemma 1 and noticing that the MAC verification fails. The probability of such a failure is upper bounded by $(1 - 1/e)2^{-m+1}$. Moreover, if $\alpha' > \alpha$ is used, the probability of an undetected failure decreases exponentially in $\alpha' - \alpha$. For $n = 2m$, (2) predicts $v = 2^{m+1}(1 - 1/e) + 4$, while the exact expression is $v = 2^{m+1}(1 - 1/e) + 6$. It can be shown that the relative error is upper bounded by $3/2^m$ for practical values of m.

Corollary 1 *Creating t MAC forgeries by the method of Lemma 1 requires one internal collision and t chosen-text MAC requests. The cost of the internal collision is given by Propositions 1 and 2.*

B. Extended Results with Common Trailing Blocks

The attack outlined in the proofs of Propositions 1 and 2 yields an internal collision (x, x'). If x and x' have a common sequence of s trailing blocks and if the compression function f is a permutation (for fixed x_i), the collision must occur at H_{t-s}, i.e. just before the common blocks. After deleting the s common blocks in x and x', one still has an internal collision. In this case the attack can be enhanced since this provides additional freedom in the choice of the text forged by the method of Lemma 1. In particular, if x and x' have the same length one can obtain a forgery on a text of that length. As a significant consequence, *in this case the attack cannot be precluded by prepending the length of the input before the MAC calculation or by fixing the length of the input.*[2]

If all the texts in the known text-MAC pairs of Propositions 1 and 2 have a common sequence of s trailing blocks, and if the compression function is a random mapping for fixed x_i, fewer known and chosen texts are required as per Proposition 3, although the total number of operations of the compression function increases. The proof of this requires a generalization of the birthday attack as given in Lemma 2.

Note 3: Lemma 2 is of independent interest for parallelizing a collision search when the constraint is the number of hash function evaluations rather than the number of evaluations of the compression function.

Lemma 2 *Let h be an iterated MAC with n-bit chaining variable, a compression function f that is a random function (for fixed x_i), and an output transformation g that is a permutation. Consider a set of $r \geq 2$ distinct messages which have the last s blocks in common, with $r \ll 2^n$. The probability that the set contains an internal collision for h is approximately*

$$1 - \exp\left(-\frac{r(r-1)(s+1)}{2^{n+1}}\right) . \tag{6}$$

[2]This is emphasized because elsewhere wide-ranging claims have appeared regarding the security provided by these measures, as a result of their success in precluding other attacks on various MAC algorithms.

Proof: For $s = 0$ the probability p that there is no internal collision is given by

$$p = \prod_{i=1}^{r-1} \left(1 - \frac{i}{2^n}\right) \qquad \text{or} \qquad \ln p = \sum_{i=1}^{r-1} \ln \left(1 - \frac{i}{2^n}\right) .$$

If $r \ll 2^n$, one can replace the ln by the first order term of its series expansion, yielding

$$\ln p \approx \sum_{i=1}^{r-1} \frac{i}{2^n} = -\frac{r(r-1)}{2^{n+1}} \qquad \text{or} \qquad p = \exp\left(-\frac{r(r-1)}{2^{n+1}}\right) .$$

Consider now $s > 0$: if a chaining variable collision occurs just before the s constant blocks, or after one of the s constant blocks, this will be an internal collision. If f is a random function, then the events that no chaining variable collision occurs in the different iterations are independent; consequently, the probability that no internal collision occurs is equal to p^{s+1}, and the probability for at least one collision can be approximated by (6). ■

From the expressions for $s = 0$ and $s > 0$, note that for $r \gg 1$, the effect of the common blocks corresponds to multiplying r by the factor $\sqrt{s+1}$.

Lemma 2 yields an optimization of Propositions 1 and 2 as follows.

Proposition 3 *Let h be an iterated MAC with n-bit chaining variable, m-bit result, a compression function f which is a random function (for fixed x_i), and output transformation g. An internal collision for h 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 as follows: $u = \sqrt{2/(s+1)} \cdot 2^{n/2}$; $v = 0$ if g is a permutation or $s+1 \geq 2^{n-m+6}$ (the expected number of external collisions is sufficiently small); if g is a random function, v is approximately*

$$2\left(\frac{2^{n-m}}{s+1} \cdot \left(1 - \frac{1}{e}\right) + \left\lfloor \frac{n-1-\log_2(s+1)}{m-1} \right\rfloor\right) . \tag{7}$$

Proof: It follows from Lemma 2 that for u as given above, the probability of one or more internal collisions is $1-1/e$. More precisely, the number of internal collisions will be Poisson distributed with parameter $\lambda = u^2(s+1)/2^{n+1}$ [21, p. 59], hence one expects approximately a single collision. (Note that if there are collisions, the different events are not independent, but this will be a good approximation since $u^2 s = \Theta(2^n)$.) If g is a permutation, then the proposition follows by Lemma 1. If g is a random function, then the number of external collisions is Poisson distributed with parameter $\lambda' = u^2/2^{m+1} = 2^{n-m}/(s+1)$. If $s+1 \geq 2^{n-m+6}$, $\lambda' \leq 1/64$. The attack will fail only if there is exactly one external collision and no internal collision, an event with probability $\lambda' e^{-\lambda'} e^{-1} \leq 0.006$. Note that this failure cannot be detected (as per Note 2), whereas the simple absence of internal collisions can be detected and resolved by increasing u. If the total number of collisions (internal and external) exceeds 1, additional tests are carried out to eliminate the external collisions. This gives only a negligible contribution to the expected number of chosen texts, since the probability of this event is $1 - e^{-\lambda'} - \lambda' e^{-\lambda'} e^{-1} \leq \lambda' \left(1 - \frac{1}{e} + \frac{\lambda'}{e}\right) < 0.01$. If $s+1 < 2^{n-m+6}$, the expected number of external collisions is larger and the same procedure as in the proof of Proposition 2 is used to eliminate external collisions. Since f is a random function for fixed x_i, the probability that an appended y (consisting of 1 block) survives a test is at most $2/2^m$. The expected number α of steps is found from

$$\left(\frac{2}{2^m}\right)^{\alpha} \cdot \left(\frac{2^{n-m}}{s+1}\right) < 1$$

or

$$\alpha = \left\lfloor \frac{n - 1 - \log_2(s+1)}{m-1} \right\rfloor .$$

One then obtains (7) by noting that the expected number of chosen texts can be computed as in the proof of Proposition 2, and is 2α plus $2(1-1/e)$ multiplied by the expected number of external collisions. ■

Given an internal collision with $s \geq 1$ common trailing blocks, the probability that it occurs before the last w blocks is $1 - w/(s+1)$. This event can be checked with a small number of additional chosen texts. Again the attack still works if one appends an arbitrary block y after the internal collision rather than at the end. This means that an attacker can replace or delete $w \leq s$ trailing blocks, and that *the attack is applicable even if the input is of fixed length or if the length is prepended to the input* (cf. [5]).

Corollary 2 *It follows from the proof of Proposition 3 that a non-verifiable version of the MAC forgery of Lemma 1 can be achieved using* $\sqrt{2/(s+1)} \cdot 2^{n/2}$ *known texts and only a single chosen text, with success probability approximately* $1\,/(1 + 2^{n-m}/(s+1))$.

C. Implications of Results on CBC-MAC and MAA

For CBC-MAC with $m = n = 64$, Proposition 1 requires $2^{32.5}$ known text-MAC pairs; the forgery attack requires a single chosen text. For $m = 32$, Proposition 2 indicates that about $2^{32.3}$ additional chosen texts are required. The attack of Proposition 3 fails for CBC-MAC, RIPEMAC, and CFB-MAC with maximal feedback, since for these the compression function is bijective on the chaining variable for fixed x_i (e.g. $H_i = f(H_{i-1}, x_i) = E_K(H_{i-1} \oplus x_i)$). However, it does apply to CFB-MAC with feedback shorter than a full block. Other specific schemes to which the attacks apply are discussed in Section IV. Proposition 3 answers in part an open question arising in the discussion of CBC-MAC [5] on whether a bijective compression function (for fixed input x_i) allows stronger security claims.

Proposition 2 applied to MAA (where $n = 64$ and $m = 32$) requires $2^{32.5}$ known text-MAC pairs and about $2^{32.3}$ chosen text-MAC pairs. Because of the internal properties of MAA, the known texts must have the same number of blocks $\bmod 32$. The function f of MAA behaves approximately as a random function for fixed x_i; the optimized version of Proposition 3 with $s = 2^{16} - 1$ (corresponding to a fixed but arbitrary 256 Kbyte trailing block) requires $2^{24.5}$ known texts and about 83 000 chosen texts. By exploiting the internal structure of MAA, these results can be further improved: about 2^{17} chosen texts of 256 Kbyte allow a MAC forgery [42, 44]; no known texts are required. The forgery attack can also be extended to a key recovery attack, and it leads to the definition of weak keys for MAA [42, 44]. Note that the designer of MAA realized that its compression function not being a bijection might lead to weaknesses, motivating a special mode in ISO 8731-2 [28] for messages longer than 1024 bytes. However, it turns out that the above attack is applicable to this mode as well. This is apparently the first attack on MAA which is more efficient than an exhaustive key search or guessing the MAC.

IV. Application of General MAC Forgery Attacks

This section discusses the security of several proposed MAC algorithms constructed from unkeyed hash functions. First the secret prefix and secret suffix methods are considered.

Forgery on the envelope method is then examined. Recall that these methods were reviewed in Section II.

A. Secret Prefix Method

It is well-known that the secret prefix method is insecure: a single text-MAC pair contains information essentially equivalent to the secret key, independent of the key size. An attacker may append any blocks to the message and update the MAC accordingly, using the old MAC as the initial chaining variable (taking into account the padding if necessary). The messages for which an attacker can compute the MAC are restricted to those having known texts as prefix, but this is a weak restriction. The appending attack may be precluded if only a subset of the hash output bits are used as the MAC (e.g. $m = n/2$ as for MD2.5 below), or by prepending the length of the message before hashing [49]. However, relying on a prepended length for security appears to make additional demands on the properties of the hash function. Moreover, because the compression function in MD4-based hash functions is of the form $H_i = E_{x_i}(H_{i-1}) + H_{i-1}$ (addition here is modulo 2^{32}) which behaves as a random function (for fixed x_i), the attack noted following Proposition 3 still applies for $s \geq 1$.

A variation of the prefix method with MD5 is used in Kerberos V5, under the name MD2.5 [35]. The 128-bit key K_1 is derived from a 56-bit DES key K by using DES as a keystream generator in Output Feedback Mode (OFB) with $IV = 0$. The MAC consists of the leftmost 64 bits of the 128-bit hash result. While the expansion does not preclude an exhaustive search for the DES key, it appears to provide some benefit. In addition, revealing only 64 bits of the hash result prevents simply appending one or more blocks to allow update of the MAC. However, the technique of Proposition 3 still applies if $s \geq 1$ and there is an internal collision before the last block ($w \geq 1$). Also, it remains conceivable that one could append carefully chosen blocks at the end in such a way that the new MAC depends only on the 64 known bits, implying that choosing $m < n$ imposes additional conditions on the hash function beyond those for which it was designed or has yet been analyzed. An unfortunate additional drawback is that, while one advantage of an MD5-based MAC over a DES-CBC MAC is avoidance of block ciphers and associated export issues, DES is nonetheless required for key expansion in MD2.5. Finally, the threat of exhaustive key search [52] may be of concern due to the (relatively short) 56-bit key.

B. Secret Suffix Method

A concern with the secret suffix method is that an off-line collision attack on the hash function may be used to obtain an internal collision. Therefore by a birthday attack, finding a pair (x, x') such that $h(x) = h(x')$ is possible in expected $\Theta(2^{n/2})$ off-line operations; Lemma 1 may then be applied. The candidate inputs in a collision search can also be chosen from an adversary-controlled set. Furthermore, this method is weak if an (off-line) second preimage attack on the underlying hash function is feasible – given one known text-MAC pair, a second hash function preimage (for that text) allows an existential MAC forgery. If t text-MAC pairs are known, finding a MAC second preimage requires $2^n/t$ rather than 2^n off-line trials; here, if the length of the message is not appended, t is the total number of blocks rather than the number of messages. The concern of off-line collision search also applies to DES MAC MD5 of [35], which consists of applying CBC-MAC to the image under MD5 of the data input.

C. Envelope Method

The envelope method used with MD4-based hash functions (see Section IV.A) is subject to the forgery of Lemma 1 regardless of the bitlength of the lead and trail keys. More specifically, Proposition 3 applies with $m = n = 128$ (assume the last block consists of K_2 only). For $s = 2^{16}$, one chosen text and $2^{56.5}$ known text-MAC pairs are required. Consequently, the key size $k_1 + k_2$ gives a misleading impression of the security for this scheme.

If K_2 is simply appended to the message, the last two blocks processed by the hash function will consist of some message bits followed by K_2, some padding bits and the length of the message (for more details, see Section V.B). In this case, the output transformation begins with the block containing some bits of K_2. The forgery attack works in the restricted case that all known messages are of the same length, and the message bits involved in the output transformation are constant.

V. Key Recovery Attacks

Three partitionable MAC key recovery attacks are presented below; one can consider these to be divide-and-conquer attacks as they allow to first recover the first half of the key (independent of the second half), and then to recover the second half by exhaustive search. The first two apply to variants of the envelope method. The last is a key recovery attack on CBC-MAC-Y, a strengthened variation of CBC-MAC [2, 29]. The section concludes with a note on how a statistical cryptanalytic technique which fails against an underlying unkeyed hash function might nonetheless succeed against a MAC construction (e.g. the envelope method) based thereon.

A. Divide and Conquer Exhaustive-Search Key Recovery on Envelope Method

Consider the envelope method (see Sections II and IV.C) with distinct keys K_1, K_2, where $k_i = |K_i|$ and $k_1 = n$ is the bitlength of the chaining variable. It has been claimed [49] (along with a sketch of proof) that a divide and conquer attack (or partitionable attack) against K_1 and K_2 is not possible, and that breaking this method requires exhaustive search for a key of $k_1 + k_2$ bits. This statement is now shown to be false.

Proposition 4 *For the envelope method with distinct keys K_1, K_2, $k_i = |K_i|$, with a chaining variable of bitlength $n = k_1$, a key recovery attack is known and uses $2^{(n+1)/2}$ known texts of at most t blocks each ($t \geq 2$) and exhaustive search involving at most $2(t-1) \cdot 2^{k_1} + 2^{k_2}$ operations.*

Proof: By Proposition 3 (assume $s = 0$ for simplicity) an internal collision for the chaining variables may be found using $2^{(n+1)/2}$ known texts (e.g. $2^{64.5}$ for MD5). With

s common trailing blocks, this can be further reduced (e.g. to $2^{56.5}$ text-MAC pairs for $s = 2^{16}$). An attacker can then perform an exhaustive search for K_1 in $2(t-1) \cdot 2^{k_1}$ off-line operations, eliminating all trial key values not yielding a collision before the last block (i.e. an internal collision).[3] Slightly over k_1/n internal collisions are required to determine K_1 uniquely; two suffice for $k_1 = n$. Once K_1 is known, the envelope method is effectively reduced to the (secret suffix) method wherein a secret key is only appended and which is vulnerable to an off-line exhaustive search for K_2. ∎

Proposition 4 indicates that using $K_1 \neq K_2$ offers substantially less additional security than one might suppose, relative to the k_1 bits of security resulting when $K_1 = K_2$. The attack does however require an unreasonably large number of known text-MAC pairs. Moreover, these texts must have equal length if the length is included in the padding bits.

B. Slice-by-Slice Key Recovery of Trail Key in Envelope Method

This section presents a key recovery attack against the trailing key K_2 in the envelope method. This includes the case where $K_1 = K_2$ of the method proposed in Internet RFC 1828 [38] (and [32])) described in Section II. The attack exploits the padding procedure of MD5, which was not designed to conceal secret keys. It also applies to any hash function with a similar trailing padding technique. The attack again requires a very large number of known text-MAC pairs (variable depending on choices made, but on the order of 2^{64} assuming 128-bit chaining variables); the work complexity for key recovery is on this same order, albeit dramatically less than exhaustive search.

Recall the padding procedure for MD5 for a message input y of bitlength ℓ, where $\ell = |y|$. A single '1' bit is appended to y, followed by z '0' bits ($0 \leq z \leq 511$), where z is chosen to make the sum of ℓ and the bitlength of the padding equal $448 \bmod 512$. The 64-bit integer representation of ℓ is then appended to complete the last block. For the special case of the IPsec envelope method with a 128-bit key, the data, after padding, processed by the compression function of MD5 has the form: $K \| p \| x \| K \| 1000\ldots000 \| \ell$. Here x is the

[3] More precisely, the output transformation $g(H_j)$ may involve 1 or 2 blocks (see Figure 1).

message on which a MAC is desired, $y = K\|p\|x\|K$, and $\ell = 512 + 128 + |x|$. Defining $r = |x| \bmod 512$,

$$z = \begin{cases} 319 - r & \text{if} \quad 0 \leq r \leq 319 \\ 831 - r & \text{if} \quad 320 \leq r \leq 511 \end{cases}$$

If $z \in [0, 319]$, the key K will lie completely in the last block, and the number of message bits in the last block is r. For $z \in [320, 446]$, $z - 319$ bits of the key K will be in the second last block, with the remaining key bits in the last block. For $z \in [447, 511]$, K falls completely in the second last block. In the latter two cases, there will be r message bits in the second last block (see Figure 1).

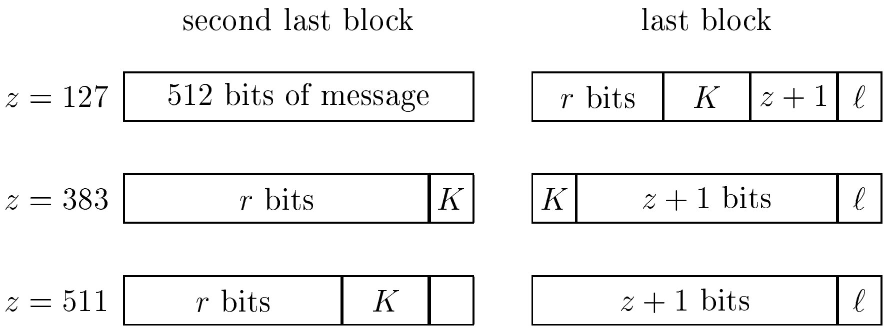

Figure 1: Message, key, padding and length fields in final blocks of envelope method.

Define an internal collision as a pair of inputs (x, x') which produce the same MAC output, and for which the internal chaining variables collide just before the block containing the key (or any partial key). Since such a collision is detectable only through a collision for the MAC, all blocks following the internal collision must be identical in the two members of the colliding input pair. Therefore the attack of Proposition 3 requires the lengths of all the messages to be equal, and the last r message bits (which are either in the last or in the second last block) to be the same. If $r = 0$ (i.e. $|x| = 0 \bmod 512$), there is no condition on the last message bits.

Consider the case $r = 511$ (i.e. $z = 320$). There is a single key bit in the second last block. Therefore 511 message bits in the second last block must be identical to allow for identification of an internal collision. However, if that key bit is simply guessed, the unknown key is restricted to the last block, and collisions after the second last block are again internal collisions (or *almost internal* collisions). A first observation is that this reduces the constraint on the message. A more significant consequence is that by using the attack of Lemma 1, one can actually verify the guess for that key bit. This leads to a powerful divide and conquer attack (or partitionable attack) against the key which may be illustrated as follows.

Let x be a 480-bit message. Then r=480, z=351, and the first block contains the padded key K. The second block contains 480 message bits and 32 key bits. The last block contains the 96 remaining key bits, a '1' bit followed by 351 '0' bits, and the 64-bit length field ℓ=1120 (i.e. $512+128+480$). If the MACs for about $2^{64.5}$ such messages x are known, one may expect (by Proposition 2 with $n = m = 128$) about two collisions: one after the second block (an almost internal collision), and one after the last block (an external collision). Denote the almost internal colliding pair (x, x'). Construct 2^{32} message pairs of the form $(x\|k_i\|y,\, x'\|k_i\|y)$, where k_i is 32 bits and ranges over all 2^{32} possible values, and y is now an arbitrary block. Request the 2^{33} corresponding MACs. When k_i takes on the value of the correct partial key, the two MACs agree; moreover, with probability $\approx 1-1/2^{96}$, no other pairs of MACs will be equal. This reveals 32 key bits. For the external collision, with overwhelming probability none of the pairs gives the same MAC.

It is easy to extend the attack to find further key bits. One possibility is to repeat the above procedure using messages of length 448 bits, yielding the next 32 key bits. The remaining 64 key bits are then most efficiently found (off-line) exhaustively. Alternatively, one could begin with messages of this length, which would require 2^{66} chosen texts, but reveal 64 bits of the key immediately. This reasoning allows the following result, which is stated for clarity specifically with respect to Internet RFC 1828, but is easily seen to be a more general result:

Proposition 5 *There exists a key recovery attack on the (RFC 1828) envelope method which uses $q = \lceil 64/d \rceil$ steps ($1 \leq d \leq 64$) to find 64 bits of the key. Step i ($1 \leq i \leq q$) requires $\sqrt{2} \cdot 2^{64}$ known texts of bitlength $c_i \cdot 512 - d \cdot i$ for some fixed $c_i > 1$, and 2^{d+2} chosen* *texts.*

Table 1 summarizes the complexity to find 64 key bits in d-bit slices, for different values of d. If a 128-bit key is used with the remaining bits found by exhaustive search, the overall time complexity is on the order of the number of known texts. The attack is easily modified for keys exceeding 128 bits; e.g. recovering a 256-bit key in three 64-bit slices requires about 2^{66} known text-MAC pairs and the same order of chosen texts. Thus relative to this attack, this MAC design makes very poor use of key bits beyond 128. For context, recall that linear cryptanalysis of DES [37], viewed as a tremendous breakthrough, requires 2^{43} known texts against a 56-bit key, while differential cryptanalysis requires 2^{47} chosen texts [10]. The new key recovery attack, relative to a larger 128-bit key, requires substantially fewer known texts (and time); this indicates that the general construction fails to make good use of key bits.

Table 1: Complexity of key recovery attack on envelope method (128-bit key)

d	# known texts	# chosen texts
2	$2^{69.5}$	2^{9}
4	$2^{68.5}$	2^{10}
8	$2^{67.5}$	2^{13}
16	$2^{66.5}$	2^{20}
32	$2^{65.5}$	2^{35}

The above attack requires that $|x| \bmod 512 \in [448, 511]$, because the number of bits of K in the penultimate block must be between 1 and 64; and that the known texts have the same number of blocks, because the value of ℓ must be the same for the two messages forming the internal collision. However, if a set of about 2^{73} "short" (say ten or fewer blocks) known messages is available, one expects to find among those a sufficient number of messages suitable for the attack (without fixing d in advance); the attack will still require a much smaller number (less than 2^{20}) of chosen texts to identify the key bits.

The attack relies on the key being split across blocks. While it is not practical, vulnerability to it represents a certificational weakness, and indicates an architectural flaw. It is certainly one of the reasons for the fact that SSL 3.0 [25] has replaced the MAC algorithm of RFC 1828 (that was used in Secure Sockets Layer SSL 2.0) by HMAC [8]. One concludes it is more secure to isolate the entire trailing key in a separate block (together with the message length and possibly a pseudo-random string). However, this requires changing the padding procedure for MD5, contravening an original motivating factor – being able to call the underlying hash function directly; an alternative is to use two nested calls as in the HMAC construction [8]. Nonetheless, customized MACs (as suggested in [32, 41]) appear to offer a more secure alternative to constructions relying directly on unkeyed hash functions.

This attack does not contradict the security proof for this scheme given by Bellare et al. [7], because the required number of known or chosen texts is larger than their security bound.

C. Key Recovery on CBC-MAC-Y (ANSI X9.19–ISO/IEC 9797)

CBC-MAC-Y is a modification of CBC-MAC (see Section II) intended to increase the security. It replaces the processing of the last block from $E_{K_1}(x_t \oplus H_{t-1})$ to a two-key triple encryption:

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

Aside from precluding the existential forgery attack noted in Section II, this is intended to prevent an exhaustive key search attack on K_1. This is a particular concern when DES is used as the block cipher E, because its key is only 56 bits [52]. However, a new divide and conquer (or partitionable) key recovery attack is possible provided (e.g. in the case $m = n$) $2^{(n+1)/2}$ known text-MAC pairs are available. This attack follows similarly from Proposition 1; for more details, see [43].

Proposition 6 *For the strengthened version CBC-MAC-Y [2, 29] of CBC-MAC, a key recovery attack yielding both keys K_1 and K_2 is known which uses $2^{(n+1)/2}$ known texts of at most t blocks each ($t \geq 2$) and exhaustive search involving at most $(2t-1) \cdot 2^k$ encryptions,* *where* $k = |K_1| = |K_2|$*,* $k \leq n$*, and* $m = n$*.*

Note 4: If triple encryption (e.g. triple-DES CBC-MAC) is used at each stage in the MAC operation, Proposition 6 does not apply. However, an output transformation is required to prevent the existential forgery attack on CBC-MAC described in Section II. *Moreover, the basic forgery attack of Proposition 1 does apply and the complexity (about* 2^{32} *known texts and 1 chosen text for* $n = 64$*) is independent of the key size.*

D. Statistical Cryptanalysis of Envelope MACs

Even if the above attacks are precluded in some way, one should keep in mind that the attacks are independent of possible weaknesses of the hash function. More sophisticated attacks might be found which exploit such weaknesses, even if they do not influence the one-wayness or collision resistance of the hash function (see for example [3]). As an example, consider the envelope method with a 128-bit key K; the remaining message input is under control of an attacker. Assume that there exists a number of probabilistic relations between the key bits in the data input and the bits of the MAC. Linear cryptanalysis, as proposed by M. Matsui [36, 37] could then be used to recover part of the key K using a number of known text-MAC pairs (for example, 2^{40} known text-MAC pairs might allow one to recover 60 bits of K). The remaining 68 bits could then be found by exhaustive search. Note that this type of attack is completely different from a (second) preimage attack, where an opponent only knows the hash result and possibly a single preimage. Here, he has at his disposal 2^{40} hash results, for which he knows the complete input except for the 128 input bits of K, which are the same in all these cases. This scenario illustrates another important point: the assumption that the complete output of the compression function is unpredictable when *part* of it is keyed (on which the security proof of [7] is based), is quite a different property than collision resistance or (2nd) preimage resistance. It should be noted that the open literature contains no analysis of existing hash functions with respect to the former property.

VI. Discussion of Results

Table 2 gives the number of text-MAC pairs required by the best known attacks on MAA and on CBC-MAC with m-bit result (using a block cipher such as DES with $n = 64$).

Table 2: Security of MAA and CBC-MAC

	no. of known texts	no. of chosen texts
MAA (general)	$2^{32.5}$	$2^{32.3}$
MAA (long message)	—	2^{17}
CBC-MAC ($m = 32$)	$2^{32.5}$	$2^{32.3}$
CBC-MAC ($m = 64$)	$2^{32.5}$	1

The weaknesses of the proposals based on hash functions of Section IV are summarized in Table 3. Storage requirements (e.g. for known text-MAC pairs) have been omitted, as well as the potential improvements due to common trailing blocks as discussed in Section III. The tabulated values, corresponding to the best known attacks, give *upper* bounds on the security of these constructions. Depending on the parameters, finding a second preimage may be easier by first obtaining the key with an exhaustive search; this type of attack is not noted in the table.

If the underlying hash function is collision resistant (implying n is sufficiently large), the figures in Table 3 (aside from the secret prefix method without additional precautions) indicate that the corresponding attacks are only *certificational* – breaking these schemes is easier than breaking an ideal MAC with the same parameters, although the attacks are clearly infeasible in practice. In particular, the number of known or chosen texts required is much smaller than would be ideal, and known texts can be replaced by off-line computations. It is however clear from Table 3 that if the hash function is only a one-way hash function (with n typically between 64 and 80 bits), then both the suffix and envelope methods are vulnerable as well. Also, it follows that in case of the envelope method k_1 must not be too small.

Table 3: Security of 3 proposals to build n-bit MACs ($n = m$) from hash functions. "#MAC" is the number of known text-MAC pairs; "C" the number of chosen texts; "#opn" the number of off-line compression function operations required for best known attacks; t is the number of messages (or blocks) available to an attacker; k, k_1, k_2 are key bitlengths.

	ideal MAC (k)		secret prefix (k_1)		secret suffix (k_2)		envelope ($k_1 + k_2$)	
	#MAC	#opn	#MAC	#opn	#MAC	#opn	#MAC	#opn
key recovery	$\lceil \frac{k}{n} \rceil$	2^k	1	0‡	$\lceil \frac{k_2}{n} \rceil$	2^{k_2}	$\lceil \frac{k_1+k_2}{n} \rceil$ $2^{n/2}$	$2^{k_1+k_2}$ $2^{k_1+1} + 2^{k_2}$
MAC forgery	$\lceil \frac{k}{n} \rceil$	2^k	1	1	1C	$2^{n/2}$	5C+$2^{n/2}$ $2^{n/2}$	0 2^{k_1}†
2nd preim.	2^n	0	t	$2^n/t$	t	$2^n/t$	2^n	0

†This attack reduces the envelope method to the secret suffix method only.

‡Information essentially equivalent to the secret key is known.

Even if keys are chosen sufficiently large that these attacks are computationally infeasible, one should keep in mind the attacks are independent of possible weaknesses of the hash function. More sophisticated attacks might be found which exploit such weaknesses (cf. Section V.D). This is a particular concern in light of recent techniques which allowed collisions to be found for MD4, and for two rounds of RIPEMD [18, 19].

VII. Concluding Remarks

The new forgery attack on iterated MACs requires expected $\Theta(2^{n/2})$ known text-MAC pairs and expected $\Theta(2^{n-m})$ chosen texts, where m is the bitlength of the hash result and n is that of the chaining variable. Thus a square-root attack applies to MACs as in many other cryptographic problems (albeit the square root is in the number of texts required). A naive non-verifiable attack always succeeds with probability 2^{-k} by guessing the k-bit key and computing the MAC, or 2^{-m} by guessing the MAC. These attack scenarios differ, but nonetheless suggest using $n = 2m$ and $k \geq m$. An important conclusion is that the attack can be avoided by varying the output transformation using a sequence number or a random number, although this adds the inconvenience that this latter value must be separately available for use in verification of the MAC.

The new attack may pose a relatively serious threat to certain applications of CBC-MAC (e.g. when $n = m = 64$), and illustrates that CBC-MAC-Y, the strengthened version of CBC-MAC in ANSI X9.19 and ISO/IEC 9797, offers increased strength against exhaustive key search only if less than expected $\Theta(2^{n/2})$ known text-MAC pairs are available. Its implications for the security of MAA are also serious. The analysis of existing proposals indicates that one must exercise care in designing MACs based on hash functions.

Acknowledgements

The authors would like to thank the anonymous referee(s), for comments resulting in a more readable paper, and in particular for encouraging clarification of the proofs of Propositions 1 and 2.

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. Anderson, "The classification of hash functions," in *Codes and Cyphers: Cryptography and Coding IV*, P. G. Farrell, Ed. Institute of Mathematics & Its Applications (IMA), 1995, pp. 83–93.

[4] M. Atici and D. Stinson, "Universal hashing and multiple authentication," in *Advances in Cryptology, Proc. Crypto'96, Lecture Notes in Computer Science, vol. 1109*, N. Koblitz, Ed. New York: Springer-Verlag, 1996, pp. 16–30.

[5] M. Bellare, J. Kilian, and P. R. Rogaway, "The security of cipher block chaining," in *Advances in Cryptology, Proc. Crypto'94, Lecture Notes in Computer Science, vol. 839*, Y. Desmedt, Ed. New York: Springer-Verlag, 1994, pp. 341–358.

[6] M. Bellare, R. Guérin, and P. R. Rogaway, "XOR MACs: new methods for message authentication using block ciphers," in *Advances in Cryptology, Proc. Crypto'95, Lecture Notes in Computer Science, vol. 963*, D. Coppersmith, Ed. New York: Springer-Verlag, 1995, pp. 15–28.

[7] M. Bellare, R. Canetti, and H. Krawczyk, "Pseudorandom functions revisited: The cascade construction and its concrete security," in *Proc. 37th Annual Symposium on the Foundations of Computer Science*, Los Alamitos: IEEE Computer Society Press, 1996, pp. 514–523. Full version: http://www-cse.ucsd.edu/users/mihir.

[8] M. Bellare, R. Canetti, and H. Krawczyk, "Keying hash functions for message authentication," in *Advances in Cryptology, Proc. Crypto'96, Lecture Notes in Computer Science, vol. 1109*, N. Koblitz, Ed. New York: Springer-Verlag, 1996, pp. 1–15. Full version: http://www.research.ibm.com/security/.

[9] M. Bellare, R. Canetti, and H. Krawczyk, "HMAC: Keyed-hashing for message authentication," *Request for Comments (RFC) 2104*, Internet Activities Board, Internet Privacy Task Force, Feb. 1997.

[10] E. Biham and A. Shamir, *Differential Cryptanalysis of the Data Encryption Standard.* New York: Springer-Verlag, 1993.

[11] F. Cohen, "A cryptographic checksum for integrity protection," *Computers & Security*, vol. 6, no. 5, pp. 505–510, 1987.

[12] I. B. Damgård, "A design principle for hash functions," in *Advances in Cryptology, Proc. Crypto'89, Lecture Notes in Computer Science, vol. 435*, G. Brassard, Ed. New York: Springer-Verlag, 1990, pp. 416–427.

[13] D. Davies, "A message authenticator algorithm suitable for a mainframe computer," in *Advances in Cryptology, Proc. Crypto'84, Lecture Notes in Computer Science, vol. 196*, G. R. Blakley and D. Chaum, Eds. New York: Springer-Verlag, 1985, pp. 393–400.

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

[15] D. Davies and W. Price, *Security for Computer Networks,* 2nd ed. New York: Wiley, 1989.

[16] B. den Boer and A. Bosselaers, "An attack on the last two rounds of MD4," in *Advances in Cryptology, Proc. Crypto'91, Lecture Notes in Computer Science, vol. 576*, J. Feigenbaum, Ed. New York: Springer-Verlag, 1992, pp. 194–203.

[17] B. den Boer and A. Bosselaers, "Collisions for the compression function of MD5," in *Advances in Cryptology, Proc. Eurocrypt'93, Lecture Notes in Computer Science, vol. 765*, T. Helleseth, Ed. New York: Springer-Verlag, 1994, pp. 293–304.

[18] H. Dobbertin, "Cryptanalysis of MD4," in *Fast Software Encryption, Lecture Notes in Computer Science, vol. 1039*, D. Gollmann, Ed. New York: Springer-Verlag, 1996, pp. 53–69.

[19] H. Dobbertin, "RIPEMD with two-round compress function is not collision-free," *J. Cryptology*, vol. 10, no. 1, pp. 51–69, Winter 1997.

[20] H. Dobbertin, A. Bosselaers, and B. Preneel, "RIPEMD-160: a strengthened version of RIPEMD," in *Fast Software Encryption, Lecture Notes in Computer Science, vol. 1039*, D. Gollmann, Ed. New York: Springer-Verlag, 1996, pp. 71–82.

[21] W. Feller, *An Introduction to Probability Theory and Its Applications,* vol. 1. New York: Wiley, 1968.

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

[23] FIPS 81, "DES modes of operation," Washington D.C.: NBS, US Department of Commerce, Dec. 1980.

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

[25] A. O. Freier, P. Karlton, and P. C. Kocher, "The SSL protocol version 3.0," *Internet Draft (work in progress)*, Internet Activities Board, Internet Privacy Task Force, March 1996.

[26] J. M. Galvin, K. McCloghrie, and J. R. Davin, "Secure management of SNMP networks," in *Integrated Network Management, II*, I. Krishnan and W. Zimmer, Eds. Amsterdam, The Netherlands: North-Holland, 1991, pp. 703–714.

[27] M. Girault, R. Cohen, and M. Campana, "A generalized birthday attack," in *Advances in Cryptology, Proc. Eurocrypt'88, Lecture Notes in Computer Science, vol. 330*, C. G. Günther, Ed. New York: Springer-Verlag, 1988, pp. 129–156.

[28] ISO 8731:1987, "Banking – Approved algorithms for message authentication, Part 1, DEA, Part 2, Message authentication algorithm (MAA)."

[29] ISO/IEC 9797:1993, "Information technology - Data cryptographic techniques - Data integrity mechanisms using a cryptographic check function employing a block cipher algorithm."

[30] T. Johansson, G. Kabatianskii, and B. Smeets, "On the relation between A-codes and codes correcting independent errors," in *Advances in Cryptology, Proc. Eurocrypt'93, Lecture Notes in Computer Science, vol. 765*, T. Helleseth, Ed. New York: Springer-Verlag, 1994, pp. 1–11.

[31] R. R. Jueneman, S. M. Matyas, and C. H. Meyer, "Message authentication with manipulation detection codes," in *Proc. 1983 IEEE Symposium on Security and Privacy*, Los Alamitos: IEEE Computer Society Press, 1983, pp. 33–54.

[32] B. Kaliski and M. Robshaw, "Message authentication with MD5," *CryptoBytes (RSA Laboratories Technical Newsletter)*, vol. 1, no. 1, pp. 5–8, Spring 1995.

[33] L. R. Knudsen, "A chosen text attack on CBC-MAC," *Electron. Lett.*, vol. 33, no. 1, pp. 48–49, 1997.

[34] H. Krawczyk, "LFSR-based hashing and authentication," in *Advances in Cryptology, Proc. Crypto'94, Lecture Notes in Computer Science, vol. 839*, Y. Desmedt, Ed. New York: Springer-Verlag, 1994, pp. 129–139.

[35] J. Linn, "The Kerberos Version 5 GSS-API Mechanism," *Request for Comments (RFC) 1964*, Internet Activities Board, Internet Privacy Task Force, June 1996.

[36] M. Matsui, "A new method for known plaintext attack of FEAL cipher," in *Advances in Cryptology, Proc. Eurocrypt'92, Lecture Notes in Computer Science, vol. 658*, R. A. Rueppel, Ed. New York: Springer-Verlag, 1993, pp. 81–91.

[37] M. Matsui, "The first experimental cryptanalysis of the Data Encryption Standard," in *Advances in Cryptology, Proc. Crypto'94, Lecture Notes in Computer Science, vol. 839*, Y. Desmedt, Ed. New York: Springer-Verlag, 1994, pp. 1–11.

[38] P. Metzger and W. Simpson, "IP Authentication using Keyed MD5," *Request for Comments (RFC) 1828*, Internet Activities Board, Internet Privacy Task Force, Aug. 1995.

[39] C. Mitchell and M. Walker, "Solutions to the multidestination secure electronic mail problem," *Computers & Security*, vol. 7, no. 5, pp. 483–488, 1988.

[40] B. Preneel, *Analysis and Design of Cryptographic Hash Functions.* Doctoral Dissertation, Katholieke Universiteit Leuven, Belgium, January 1993 (updated version to appear as *Cryptographic Hash Functions.* Boston, MA: Kluwer Academic Publishers).

[41] B. Preneel and P. C. van Oorschot, "MDx-MAC and building fast MACs from hash functions," in *Advances in Cryptology, Proc. Crypto'95, Lecture Notes in Computer Science, vol. 963*, D. Coppersmith, Ed. New York: Springer-Verlag, 1995, pp. 1–14.

[42] B. Preneel and P. C. van Oorschot, "On the security of two MAC algorithms," in *Advances in Cryptology, Proc. Eurocrypt'96, Lecture Notes in Computer Science, vol. 1070*, U. Maurer, Ed. New York: Springer-Verlag, 1996, pp. 19–32.

[43] B. Preneel and P. C. van Oorschot, "A key recovery attack on the ANSI X9.19 retail MAC," *Electron. Lett.*, vol. 32, no. 17, pp. 1568–1569, 1996.

[44] B. Preneel, V. Rijmen, and P. C. van Oorschot, "A security analysis of the Message Authenticator Algorithm (MAA)," *European Trans. Telecommunications*, vol. 8, no. 5, pp. 455–470, 1997.

[45] RIPE, *Integrity Primitives for Secure Information Systems. Final Report of RACE Integrity Primitives Evaluation (RIPE-RACE 1040), Lecture Notes in Computer Science, vol. 1007.* A. Bosselaers and B. Preneel, Eds. New York: Springer-Verlag, 1995.

[46] R. L. Rivest, "The MD4 message digest algorithm," in *Advances in Cryptology, Proc. Crypto'90, Lecture Notes in Computer Science, vol. 537*, S. Vanstone, Ed. New York: Springer-Verlag, 1991, pp. 303–311.

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

[48] P. R. Rogaway, "Bucket hashing and its application to fast message authentication," in *Advances in Cryptology, Proc. Crypto'95, Lecture Notes in Computer Science, vol. 963*, D. Coppersmith, Ed. New York: Springer-Verlag, 1995, pp. 29–42.

[49] G. Tsudik, "Message authentication with one-way hash functions," *ACM Computer Communications Review*, vol. 22, no. 5, pp. 29–38, 1992.

[50] S. Vaudenay, "On the need for multipermutations: cryptanalysis of MD4 and SAFER," in *Fast Software Encryption, Lecture Notes in Computer Science, vol. 1008*, B. Preneel, Ed. New York: Springer-Verlag, 1995, pp. 286–297.

[51] M. N. Wegman and J. L. Carter, "New hash functions and their use in authentication and set equality," *J. Computer Sys. Sciences*, vol. 22, no. 3, pp. 265–279, 1981.

[52] M. J. Wiener, "Efficient DES key search," presented at rump session of Crypto'93, Santa Barbara, CA. Reprinted in *Practical Cryptography for Data Internetworks*, W. Stallings, Ed. Los Alamitos: IEEE Computer Society Press, 1996, pp. 31–79.

figure 1

	second last block			last block			
$z = 127$	512 bits of message			r bits	K	$z + 1$	ℓ
$z = 383$	r bits	K		K	$z + 1$ bits	ℓ	
$z = 511$	r bits	K		$z + 1$ bits	ℓ		

figure caption

Fig. 1. Message, key, padding and length fields in final blocks of envelope method

table 1

d	# known texts	# chosen texts
2	$2^{69.5}$	2^{9}
4	$2^{68.5}$	2^{10}
8	$2^{67.5}$	2^{13}
16	$2^{66.5}$	2^{20}
32	$2^{65.5}$	2^{35}

table 2

	no. of known texts	no. of chosen texts
MAA (general)	$2^{32.5}$	$2^{32.3}$
MAA (long message)	—	2^{17}
CBC-MAC ($m = 32$)	$2^{32.5}$	$2^{32.3}$
CBC-MAC ($m = 64$)	$2^{32.5}$	1

table 3

	ideal MAC (k)		secret prefix (k_1)		secret suffix (k_2)		envelope ($k_1 + k_2$)	
	#MAC	#opn	#MAC	#opn	#MAC	#opn	#MAC	#opn
key recovery	$\lceil \frac{k}{n} \rceil$	2^k	1	0‡	$\lceil \frac{k_2}{n} \rceil$	2^{k_2}	$\lceil \frac{k_1+k_2}{n} \rceil$ $2^{n/2}$	$2^{k_1+k_2}$ $2^{k_1+1} + 2^{k_2}$
MAC forgery	$\lceil \frac{k}{n} \rceil$	2^k	1	1	1C	$2^{n/2}$	5C+$2^{n/2}$ $2^{n/2}$	0 2^{k_1}†
2nd preim.	2^n	0	t	$2^n/t$	t	$2^n/t$	2^n	0

†This attack reduces the envelope method to the secret suffix method only.

‡Information essentially equivalent to the secret key is known.

table captions

Table 1. Complexity of key recovery attack on envelope method (128-bit key)

Table 2. Security of MAA and CBC-MAC

Table 3. Security of 3 proposals to build n-bit MACs ($n = m$) from hash functions. "#MAC" is the number of known text-MAC pairs; "C" the number of chosen texts; "#opn" the number of off-line compression function operations required for best known attacks; t is the number of messages (or blocks) available to an attacker; k, k_1, k_2 are key bitlengths.

footnotes

1. The intention of this section is to provide background to allow an understanding of subsequent results. Consequently, formality is sacrificed where informality aids understandability.

2. This is emphasized because elsewhere wide-ranging claims have appeared regarding the security provided by these measures, as a result of their success in precluding other attacks on various MAC algorithms.

3. More precisely, the output transformation $g(H_j)$ may involve 1 or 2 blocks (see Figure 1).

biographies

Bart Preneel was born in Leuven, Belgium, on October 15, 1963. He received the Electrical Engineering degree and the Doctorate in Applied Sciences in 1987 and 1993 respectively, both from the Katholieke Universiteit Leuven in 1987. He is currently a postdoctoral researcher, sponsored by the F.W.O. (Fund for Scientific Research Flanders, Belgium). He is also a part-time associate professor at the K.U.Leuven and a visiting professor at the University of Bergen in Norway. During the academic year 1996-1997, he was a visiting professor at the Universiteit Gent, and during the academic year 1993-1994, he was a research fellow of the EECS Department of the University of California at Berkeley. His main research interests are cryptography and information security. He has authored more than 40 scientific publications and has participated in several research projects sponsored by the European Commission. As a consultant, he has been involved in a large number of security studies on banking and telecommunications systems. He is a member of board of directors of the International Association of Cryptologic Research (IACR). He has served on various program committees (Asiacrypt, Crypto, and Eurocrypt), has been program chair of the 1994 workshop on Fast Software Encryption and is program chair of Eurocrypt 2000. Since 1989, he is a Belgian expert in working group ISO/IEC JTC1/SC27/WG2 (Security Techniques and Mechanisms), where he is editor of several standards.

Paul Van Oorschot was born in Ontario, Canada on June 29, 1962. He completed B.Math, M.Math, and Ph.D. (Computer Science) degrees in 1984, 1986, and 1988, all from the U. of Waterloo (Canada). From 1988 through 1996, his work at Nortel and its R&D subsidiary, Bell-Northern Research, involved cryptographic research and consulting, product R&D, and participation in national and international security standards. He is currently Chief Scientist with Entrust Technologies (`http://www.entrust.com`), a company spun out from Nortel in January 1997. He has authored over 30 scientific publications, and has been an Adjunct Research Professor in Computer Science at Carleton University (Ottawa) since 1991. He is a member of the Board of Directors of the International

Association for Cryptologic Research (IACR), has served as General Chair of Crypto'93, and has been on various program committees including Crypto, Eurocrypt, and Asiacrypt. His research interests include all practical aspects of information security and electronic commerce, especially authentication, key management, certificate management, and Public Key Infrastructures (PKIs). He is the co-author of *An Introduction to Error Correcting Codes with Applications* (Kluwer Academic Publishers, 1989) and the *Handbook of Applied Cryptography* (CRC Press, 1997).