

An Update on the Analysis and Design of NMAC and HMAC Functions

Praveen Gauravaram¹, Shoichi Hirose², and Suganya Annadurai³

(Corresponding author: Praveen Gauravaram)

Information Security Institute (ISI), Queensland University of Technology (QUT)¹

7/126 Margaret Street, Brisbane, QLD-4000, Australia (Email: p.gauravaram@isi.qut.edu.au)

The University of Fukui, 3-9-1, Bunkyo, Fukui-shi 910-8507, Japan²

Society for Electronic Transactions and Security (SETS)³

No. 21 (Old No. 11), Mangadu Swamy Street, Nungambakkam, Chennai, 600034, Tamil Nadu, India

(Received Jan. 8, 2006; revised and accepted Apr. 26 & May 7, 2006)

Abstract

In this paper, we investigate the issues in the analysis and design of provably secure message authentication codes (MACs) Nested MAC (NMAC) and Hash based MAC (HMAC) proposed by Bellare, Canetti and Krawczyk. First, we provide security analysis of NMAC using *weaker assumptions* than stated in its proof of security. This analysis shows that, theoretically, one cannot further weaken the assumptions in the proof of security of NMAC to obtain a secure MAC function NMAC and for a secure MAC function NMAC, both keys *must* be secret. This analysis also provides a solution to an open question in Preneel's thesis on the security of MAC functions when the attacker has knowledge of the key(s) in relation to NMAC and HMAC. Next, we propose a new variant to the NMAC function by altering the standard padding used for the hash function in NMAC. This variant is slightly more efficient than NMAC especially for short messages. The analysis and performance aspects of this variant are compared with other efficient MAC functions based on hash functions. Next, we provide another new variant to NMAC by altering the position of the trail key used in NMAC. This variant has some advantages over NMAC from the perspective of key-recovery attacks. Finally, we formally show how to convert NMAC and HMAC functions into pseudorandom functions.

Keywords: Message authentication codes, NMAC and HMAC, provable security

1 Introduction

One of the important applications of cryptographic hash functions is their use in the construction of efficient message authentication codes (MACs) [2, 25, 26, 27, 30]. Hash functions based on Merkle-Damgård construction [7, 22] such as SHA-1 are used with minor or no modifications

in constructing MAC schemes due to their efficiency and free availability.

The first formal security analysis for MACs based on hash functions was given by Bellare, Canetti and Krawczyk for the nested MAC (NMAC) and hash based MAC (HMAC) functions [2]. HMAC is a practical variant of the formally analyzed NMAC function. NMAC was proved to be secure if the compression function with fixed length input is a secure MAC and iterated hash function with variable length inputs is a weakly collision resistant hash function. Anyhow, not much analysis was provided for these functions based on weaker assumptions on the hash function and compression function than stated in the proof of security of NMAC [2]. In this paper, we address this issue and our result shows that by reducing the assumptions on the hash functions, NMAC becomes insecure against straight forward length extension attacks. Specifically, we show that in order to prevent extension attacks on NMAC, both *keys* must be secret.

While it is known that a MAC function must be both one-way and collision resistant when the attacker has no knowledge of the key(s), whether it has to be collision resistant or one-way when someone has knowledge of the key(s) depends on the application in which the function is used [26, p.19]. While the prime motivation behind the design of NMAC and HMAC functions is to authenticate information over an insecure medium, there are some applications that may use these proposals, especially HMAC (see [15]) requiring extra protection against the insider attacks from someone who has knowledge of the secret keys. The analysis of NMAC based on *weaker assumptions* on the hash functions explains the properties that one would naturally require from the NMAC function when the attacker has knowledge of the key(s).

Next, we revisit the proof of security of NMAC and observe that the security proof and the definitions used in the proof are independent of the padding of the hash

function and the compression function used in NMAC. We actually show why and how the proof of security of NMAC is independent of the padding of the hash function by proposing a variant to the NMAC function called NMAC-1. The padding for the inner and outer functions of NMAC-1 is dependent on the size of the message to be authenticated. NMAC-1 still observes the design principle of NMAC which is to call the compression function of the hash function as a black-box. A formal security analysis for NMAC-1 is provided.

The performance of MAC functions on short messages is important. For example, the MAC function used in IPSec operates on 43-1500 bytes [19], message authentication of signaling operate on messages that fit in one or two blocks [25] and the MAC function used in TLS operates on 0-17 kilobyte. There are also applications such as entity authentication in a mobile environment where saving of one block is important. We note that NMAC-1 is slightly more efficient than NMAC when it is used to authenticate short messages. In addition, we compare the analysis and performance aspects of the NMAC-1 function with other efficient MACs based on hash functions proposed in the literature.

The analysis of MAC schemes based on dedicated hash functions [2, 27, 28, 29, 30] shows that one has to pay attention in using the key and the hash function while designing a MAC based on the hash function. The applicability of forgery and key-recovery attacks on MACs based on hash functions depend on *how and where one uses the key and the hash function in the MAC scheme*. Following this observation, we propose a new variant to NMAC called modified NMAC (M-NMAC) by using the trail key in NMAC as a block instead of as an initial state for the outer compression function. The advantage of this variant over NMAC is that it has flexibility of using larger keys up to the block size of the compression function for the trial key which makes it much harder to perform the complete key-recovery attack on M-NMAC than on NMAC.

Finally, we note that applications such as IPSec's Key Exchange (IKE) protocol use HMAC as a pseudorandom function (PRF) to derive secret keys. Yet no explicit analysis of NMAC and HMAC functions as PRFs appeared in the literature though it appears to be that the proof techniques of [2] can be used to prove the pseudorandomness of these functions. In this work, we fill this gap by giving a formal analysis of NMAC as a pseudorandom function which applies to HMAC as well¹.

Several MAC functions based on dedicated hash functions [2, 27, 28, 29, 30] were analyzed. See Appendix C for a survey on the analysis of MACs based on hash functions. Hirose [13] has shown that weakly collision resistance of the iterated hash function in NMAC is not implied by the pseudorandomness of the compression function. He has also shown that weakly collision resistance of the iterated

hash function in NMAC implies collision resistance of its compression function if the compression function is pseudorandom. Patel [25] has proposed a variant to NMAC called Enhanced NMAC (ENMAC) by altering the standard padding scheme used in the underlying hash function to improve the efficiency of NMAC for short messages. The ISO/IEC 9797-2 [16] standard specifies a mechanism which is a variant of MDx-MAC [27] that offers high performance for applications that process short messages of upto 256 bits. Bellare *et.al* [3] have shown that the pseudorandomness of the compression function transfers to the pseudorandomness of the Merkle-Damgård iterated construction using the notion of prefix-free distinguishers.

In Section 2, we describe NMAC and HMAC functions. In Section 3, analysis of NMAC based on *weaker assumptions* than stated in its proof of security is provided. In Section 4, the proof of security of NMAC-1 is provided and in Section 5, a new variant of NMAC called M-NMAC is proposed. In Section 6, we show how to convert NMAC and HMAC to pseudorandom functions and conclude the paper in Section 7.

2 NMAC and HMAC Functions

The first formal security analysis for MACs based on hash functions was given by Bellare, Canetti and Krawczyk [2] in the form of NMAC and HMAC. The NMAC and HMAC functions are discussed below.

2.1 The NMAC Function

NMAC algorithm including its security proof was presented in [2]. An essential design goal of NMAC is to use the compression function of the hash function “as is” (as a black box).

The NMAC algorithm is defined as follows:

If k_1 and k_2 are two independent and random keys to the hash function F iterated over the compression function f , then the MAC function NMAC on an arbitrary size message x split into blocks x_1, x_2, \dots, x_n is given by

$$\text{NMAC}_k(x) = F_{k_1}(F_{k_2}(x)).$$

If the concrete realisation of NMAC uses the iterated hash function F for the inner and outer functions, then k_2 would be the IV for the inner keyed iterated hash function and k_1 would be the initial state (IV) for the outer keyed iterated hash function, which is expected to perform only one round of operation. The sizes of both keys is the same which is equal to the length of IV of the hash function F^2 . Since the two keys k_1 and k_2 act as IVs for the inner and outer functions respectively, it is clear that NMAC calls the compression function f of the hash function F as a black-box as shown in Figure 1.

¹Very recently, Bellare [1] has shown the pseudorandomness of NMAC and HMAC functions based on the pseudorandomness of the compression function.

²For example, if F is a SHA-1 hash function then $|k_1| = |k_2| = 160$

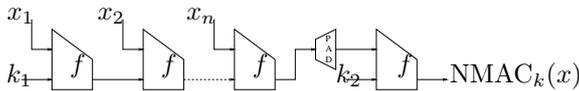


Figure 1: The NMAC construction

Since only one round of the outer function iteration is required, the compression function of the outer hash function is invoked only once. Therefore, the outer function can be renamed f and the above NMAC equation can be written as

$$\text{NMAC}_k(x) = f_{k_1}(F_{k_2}(x)),$$

where f_{k_1} is the keyed compression function. In the above NMAC equation, a standard padding technique [14] is defined for the hash function F such that the last block x_n contains the binary encoded representation of the length of the message. The output of the function $F_{k_2}(x)$ is also padded using the same standard padding operation as for the inner function, which is denoted by the PAD function in Figure 1. The PAD function is defined as $\text{PAD}(F_{k_2}(x)) = F_{k_2}(x), 1, 00 \dots 00, |F_{k_2}(x)|$ where comma indicates the concatenation operation.

HMAC is a “fixed IV” variant of NMAC and uses the hash function F as a black box. The HMAC function that works on an arbitrary length message x is defined as:

$$\text{HMAC}_k(x) = F_{IV}(\bar{k} \oplus \text{opad}, F_{IV}(\bar{k} \oplus \text{ipad}, x)).$$

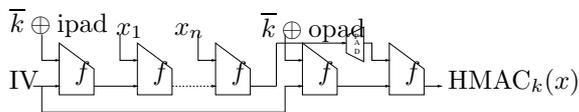


Figure 2: The HMAC construction

HMAC is a particular case of NMAC and both can be related as $\text{HMAC}_k(x) = F_{k_1}(F_{k_2}(x))$ where $k_1 = f_{IV}(\bar{k} \oplus \text{opad})$, $k_2 = f_{IV}(\bar{k} \oplus \text{ipad})$, f is the compression function of the hash function, opad and ipad are the repetitions of the bytes 0x36 and 0x5c as many times as needed to get a b -bit block, \bar{k} indicates the completion of the key k to a b -bit block by padding k with 0's and comma is the concatenation operation. Since the outer function in this expression processes only one message block, it can be written as $\text{HMAC}_k(x) = f_{k_1}(F_{k_2}(x)) = \text{NMAC}_{k_1, k_2}(x)$. The security analysis provided for NMAC applies to HMAC under the assumption that the compression function used to derive the keys k_1 and k_2 for HMAC works as a pseudorandom function [2]. The HMAC function is shown in Figure 2.

NMAC and HMAC algorithms were proved to be secure [2] given some reasonable assumptions on the underlying hash functions. The following definitions are considered in giving the security analysis of the NMAC function.

Definition 1. A MAC based on the keyed compression function f is an (ϵ_f, q, t, b) -secure MAC if any attacker,

without knowledge of the key k_1 requesting q chosen messages x_i (where $i = 1 \dots q$ and $\max(|x_i|) = b$) to the keyed compression function f , cannot break the scheme in a total time t except with probability less than ϵ_f . In other words, ϵ_f is the maximum probability of forging f_{k_1} .

Definition 2. A keyed iterated hash function F is an (ϵ_F, q, t, L) - weakly collision resistant hash function if any attacker, without knowledge of the key k_2 requesting q chosen messages x_i (where $i = 1 \dots q$ and $\max(|x_i|) = L$) to the keyed iterated hash function F , cannot find two messages x and x' in a total time t such that $F_{k_2}(x) = F_{k_2}(x')$ with probability better than ϵ_F . In other words, ϵ_F is the maximum probability of finding collisions for F_{k_2} .

3 Analysis of NMAC using Weaker Assumptions on the Hash Functions

An ideal MAC function must be both one-way and collision resistant for someone who does not know the secret key. Whether the MAC function must be one-way or collision resistant for someone who knows the secret key depends on the application [26, p.19], [21, p.327]. Though NMAC and HMAC functions were shown to be secure under the assumption that the adversary who tries to forge them has no knowledge of the key, there may be some applications where one expects these functions to satisfy some additional properties for the protection against insiders who know the secret key. An example application based on HMAC-SHA-1 requiring extra protection from the insider attacks is given in Appendix A.

In the following, we show security analysis of NMAC and HMAC using *weaker assumptions* on the hash functions. We also provide the properties that are essential for the protection of these MAC functions from the insider attacks. Our analysis shows that theoretically NMAC as a MAC function is not always secure if the assumptions on the hash functions are weaker than the assumptions in its original formal analysis [2].

3.1 Security Analysis

The Remark 4.9 of [2] has motivated us to split the analysis on NMAC into three cases. In particular, by splitting the analysis into three cases, we are able to show the main result of this analysis (case:2) that it is the *keyed* application of the external function in the MAC function NMAC which prevents extension attack but not just the plain application. Our analysis shows that, theoretically, one cannot further weaken the assumptions in the proof of security of NMAC to obtain a secure MAC function NMAC and to obtain a secure MAC function NMAC, both keys *must* be secret. A similar analysis can be applied to HMAC. This analysis also partially solves the open question in Preneel's thesis [26] on the properties required from the MAC function when the

adversary has knowledge of the key(s).

Case 1: Only k_1 is secret

In this case, we assume that the adversary knows the key k_2 of NMAC. Hence, she can find offline collisions on F_{k_2} which is easier to perform than finding collisions when the key k_2 is random and secret as the attacker needs to contact the legitimate owners of the key k_2 to get collisions for F_{k_2} .

Once the attacker finds x and x' such that $F_{k_2}(x) = F_{k_2}(x')$, this MAC function is attackable using the chosen message attack. The attacker sends x to the NMAC oracle and gets the MAC value as response and uses this MAC value as a forgeable value for the message x' . Once the attacker gets a collision on the internal function, the attacker can extend the collisions on the collided pair of messages x and x' to the format $F_{k_2}(x||s) = F_{k_2}(x'||s)$ where s is an arbitrary text appended to the collided messages. This attack was observed in [2]. Hence, to attain a secure MAC function NMAC, the internal function F_{k_2} must be collision resistant which implies the complexity of finding collisions on F_{k_2} must be at least $2^{|k_2|/2}$ according to birthday attack.

Case 2: Only k_2 is secret

This case assumes that the attacker knows the key k_1 used in NMAC and provides the security analysis of the scheme against straight forward extension attacks. The analysis also assumes that the attacker can only access the NMAC oracle as a whole though it has knowledge of the key k_1 . This analysis was not provided in [2].

Hence, to forge the MAC scheme presented in this case, it seems that the attacker either has to invert the function f_{k_1} for the known output of the NMAC function or has to find collisions for NMAC as the attacker cannot get the actual result of $F_{k_2}(x)$ but only its value after applying f_{k_1} . The value $F_{k_2}(x)$ can be viewed as a *message dependent secret* input to the function f_{k_1} . Hence the function f_{k_1} must be one-way as it should be hard for the attacker to find $F_{k_2}(x)$ using x and the NMAC output.

Hirose [13] has shown that the weakly collision resistance of F_{k_2} implies collision resistance of f used in F_{k_2} under the assumption that f is a PRF. Assume that the the outer compression function f is different from the compression function in the inner iterated hash function F . From this discussion, it seems that naturally the *function f_{k_1} must be both one-way and collision resistant for some one who knows the key k_1* . In the following analysis, we show that one cannot always attain a secure MAC function NMAC with a security requirement weaker than weakly collision resistance for the function F_{k_2} even if f_{k_1} is both one-way and collision resistant.

Theorem 1. *A one-way and collision resistant external function f_{k_1} (k_1 is known) and a weakly-collision resistant F_{k_2} inner function do not always imply a secure MAC function NMAC.*

Proof. Let $F_{k_2} : \{0, 1\}^* \rightarrow \{0, 1\}^l$ and $f_{k_1} : \{0, 1\}^l \rightarrow \{0, 1\}^n$.

Let $G_{k_2} : \{0, 1\}^* \rightarrow \{0, 1\}^{l'}$ where $l > l'$.

Let $g_{k_1} : \{0, 1\}^{l'} \rightarrow \{0, 1\}^n$ be one-way and collision resistant.

The function F_{k_2} is defined as $F_{k_2}(x) = G_{k_2}(x)||0^{l-l'}$.

The function f_{k_1} is defined as $f_{k_1}(z'||z'') = z' || g_{k_1}(z'')$ where $z' \in \{0, 1\}^{l'}$. Then f_{k_1} is also collision resistant and one-way.

On the other hand, $f_{k_1}(F_{k_2}(x)) = G_{k_2}(x)||g_{k_1}(0^{l-l'})$ and $G_{k_2}(x)$ is obtained from the NMAC oracle. Hence, even if the function f_{k_1} is one-way and collision resistant, it cannot always prevent the straight-forward extension attack as the output of f_{k_1} gives $G_{k_2}(x)$. \square

Remarks:

- 1) The above analysis shows that the function G_{k_2} should be a secure MAC to make NMAC a secure MAC function. The function F_{k_2} is also a secure MAC if G_{k_2} is a secure MAC. The function G_{k_2} will work as a secure MAC function *only* when the input messages are *prefix-free* as extension attacks do not work on the hash functions based on Merkle-Damgård construction for *prefix-free* input messages [6].
- 2) From the theoretical point of view, the above analysis shows that the outer function with no secrecy is not always good enough to prevent straight forward extension attacks. Nevertheless, one can use more “natural” f_{k_1} and F_{k_2} to attain a secure MAC function to protect against the insiders who know the key k_1 .
- 3) The above analysis also conveys that there may be still some NMAC functions that are secure even if the underlying functions have the above stated properties.

This MAC scheme is weaker than NMAC from the perspective of complete key-recovery as it uses just one key. Note that the NMAC function does not achieve security against the key-recovery over the combined lengths of the keys due to the divide and conquer key recovery attack and the complexity of this attack on w is about $2^{|k_1|} + 2^{|k_2|}$ [2]. Note that this attack is impractical for reasonable key sizes of k_1 and k_2 . Anyhow, once the attacker gets the key k_2 employing this attack, the security of the NMAC function against forgery reduces to the secret prefix scheme and there is no need for the attacker to find the key k_1 to perform a forgery.

Case 3: Both k_1 and k_2 are not secret

When the attacker knows both the keys k_1 and k_2 of NMAC, it is obvious that both functions F_{k_2} and f_{k_1} must be collision resistant for a protection against insider attacks. In such a case, this scheme will provide $n/2$ -bit security level against extension attacks where n is the

size of output in bits. This scheme is basically the double hashing scheme proposed in [10] to obtain a higher security level against extension attacks.

4 On the Proof of Security of NMAC

In this section, we show that the proof of security of NMAC does not depend on padding technique used for the hash function by specifying a new and simple padding technique for the inner and outer functions in NMAC. That is, by proving the security of the NMAC function with the new padding technique, we demonstrate that the security definitions and proof used in [2] for establishing the security of NMAC are independent of the specification of padding. The NMAC function with the new specification for padding shall be called NMAC-1, which shall mean NMAC variant 1. Since padding of the message is not part of the compression function, NMAC-1, like NMAC uses the compression function f of the iterated hash function F “as is”.

4.1 Specification of NMAC-1

An arbitrary finite length message is defined as x . The message is considered as short if it fits in one block or less than one block. The maximum size of the block is b which is equal to 512 bits for hash functions such as SHA-1, SHA-256 and equals 1024 bits for functions such as SHA-384 and SHA-512 [9]. In general, $|b| \geq 2n$ where n is the size of chaining variable and also the size of the MAC.

The NMAC-1 function on an arbitrary length message x is defined as follows:

$$\text{NMAC-1}(x) = f_{k_1}(F_{k_2}(x)),$$

where $F_{k_2}(x)$ is defined as follows based on the size $|x|$ in bits of the input message x . $F_{k_2}(x) =$

$$\begin{cases} f_{k_2}(x) & \text{if } |x| = b \\ f_{k_2}(x) \text{ with the input } x = x||10\dots 0 & \text{if } |x| < b \\ \text{iteration of } f_{k_2} \text{ with the input } x||10\dots 0 & \text{if } |x| > b. \end{cases}$$

For the case, $|x| \neq b$, the message x is padded with a bit 1 followed by 0's (possibly none) to make x a multiple of the block length b of the compression function. That is, when $|x| = b - 1$, only bit 1 is padded to x and when $|x| = b - 2$, x is padded with bits 1 and 0.

The padding for the outer function f_{k_1} is based on the size of the input message x . If $|x| = b$, then f_{k_1} is padded with 0's else it is padded with a bit 1 followed by 0's.

The improvement in the efficiency of NMAC-1 over NMAC is considerably high for short messages. Note that the total number of calls to the function f of NMAC-1 are two compared to three in NMAC when $|x| = b$. See Appendix B for the performance comparison.

4.2 Security Analysis of NMAC-1

The terminology used in the proof of security of NMAC-1 shall be the same as those used in [2] for the sake of clarity. The analytical results in this paper are given referring to chosen or adaptive chosen message attacks. The main analytical result of NMAC-1 uses the definitions of a secure MAC and weakly collision resistant hash function given in Section 2. The analysis also uses the following definition on weakly collision resistant compression function.

Definition 3. A keyed compression function f_{k_2} is an (ϵ'_f, q, t, b) - weakly collision resistant compression function if any attacker, without knowledge of the key k_2 , requesting q chosen messages x_i (where $i = 1 \dots q$ and $\max(|x_i|) = b$) to the the function f_{k_2} cannot find two messages x and x' in a total time t such that $f_{k_2}(x) = f_{k_2}(x')$ with a probability better than ϵ'_f . In other words, ϵ'_f is the maximum probability of finding collisions for f_{k_2} .

Theorem 2. The keyed compression function f is an (ϵ_f, q, t, b) -secure MAC implies that the NMAC-1 function is an $(\epsilon_f + \epsilon_F + \epsilon'_f, q, t, L)$ secure MAC under the assumption that the keyed iterated hash function F is an (ϵ_F, q, t, L) -weakly collision resistant hash function and the fixed input keyed compression function f is an (ϵ'_f, q, t, b) -weakly collision resistant compression function where $L \geq b$.

Proof. The parameters q, t and L for the number of queries to the NMAC-1 oracle, the total attack time t and the maximum length L of each finite length message x_i (where $i = 1, 2, \dots, q$) to be queried are fixed.

The attacker A_N that tries to break NMAC-1 sends each message x_i to the NMAC-1 oracle which gives response $\text{NMAC-1}_k(x_i)$ for every queried message x_i . Finally, the attacker A_N outputs the message x and its forged tag y . The forgery is successful if $x \neq x_i$ and $\text{NMAC-1}_k(x) = y$. Let ϵ_N be the maximum probability that A_N succeeds in forging NMAC-1.

Using A_N , we build an attacker A_f that aims to forge the MAC function f_{k_1} , on inputs of messages of length b by sending q queries to the oracle f_{k_1} in time t , with a maximum probability of ϵ_f . The proof model of NMAC applies to NMAC-1 because the adversary A_f in NMAC processes each message x_i block after block padding the last block and computing $F_{k_2}(x_i)$ for every unpadded input message x_i sent by A_N to the NMAC oracle. So, A_f can be simulated to perform $F_{k_2}(x_i)$ for every message x_i using the proposed padding given for the internal function in NMAC-1. Moreover, the proof of NMAC allows the adversary A_f to choose its own messages to query the MAC function f_{k_1} . Therefore, A_f can be simulated to query f_{k_1} using the proposed padding scheme given for the outer function in NMAC-1. Hence, the security of the NMAC function [2] under the given proof model is independent of the padding technique employed for the inner keyed iterated hash function and the outer keyed compression function.

The algorithm of A_f is given below:

Choose random k_2

For $i = 1, \dots, q$ perform the following steps:

- 1) $A_N \rightarrow x_i$
- 2) A_f computes $F_{k_2}(x_i)$ according to the specified padding technique
- 3) A_f queries f_{k_1} with $\overline{F_{k_2}(x_i)}$ and gets $f_{k_1}(\overline{F_{k_2}(x_i)})$ where $\overline{F_{k_2}(x_i)}$ represents the padding of $F_{k_2}(x_i)$.
 $\overline{F_{k_2}(x_i)} = F_{k_2}(x_i) || 00 \dots 0$ if $|x| = b$ and
 $\overline{F_{k_2}(x_i)} = F_{k_2}(x_i) || 10 \dots 0$ if $|x| \neq b$.
- 4) $A_N \leftarrow f_{k_1}(\overline{F_{k_2}(x_i)})$

A_N outputs (x, y) where $x \neq x_i$

A_f outputs $(F_{k_2}(x), y)$

Let ϵ_F and ϵ'_f be the maximum probabilities of any adversary for which $F_{k_2}(x) = F_{k_2}(x_i)$ and $f_{k_2}(x) = f_{k_2}(x_i)$ respectively. The probability with which the adversary A_f fails to forge f_{k_1} is given as:

$$\Pr[A_f \text{ fails}] \leq \Pr[A_N \text{ fails}] + \Pr[A_N \text{ succeeds} \wedge |x| \leq b \exists i (|x_i| \leq b \wedge f_{k_2}(x) = f_{k_2}(x_i))] + \Pr[A_N \text{ succeeds} \wedge |x| \geq b \exists i (|x_i| \geq b \wedge F_{k_2}(x) = F_{k_2}(x_i))].$$

That is, $1 - \epsilon_f \leq 1 - \epsilon_N + \epsilon'_f + \epsilon_F$

$$\Rightarrow \epsilon_f \geq \epsilon_N - \epsilon'_f - \epsilon_F.$$

$$\Rightarrow \epsilon_N \leq \epsilon_f + \epsilon'_f + \epsilon_F.$$

Hence, the probability of forging NMAC-1 is at most sum of the maximum probabilities of finding collisions for the inner keyed compression function and the keyed iterated hash function and forging the outer keyed compression function. \square

4.3 Comparison of NMAC-1 with other Efficient MACs Based on Hash Functions

Patel [25] has proposed Enhanced NMAC (ENMAC) to improve the efficiency of NMAC for short messages. A short message was defined as a message with size $|x|$ in bits less than or equal to $|b - 2|$. ENMAC function is defined as below:

$$\text{ENMAC}_k(x) = \begin{cases} f_{k_1}(x) \text{ with the input } x = x || 11 & \text{if } |x| = b - 2 \\ f_{k_1}(x) \text{ with the input } x = x || 10 \dots 01 & \text{if } |x| < b - 2 \\ f_{k_1}(x_{pref}, F_{k_2}(x_{suff}), 0) & \text{if } |x| > b - 2, \end{cases}$$

where x_{pref} contains the bits from 1 to $b - n - 1$ and x_{suff} contains bits from $b - n$ to $|x|$. They are represented as $x_{pref} = x^1, \dots, x^{b-n-1}$ and $x_{suff} = x^{b-n}, \dots, x^{|x|}$.

When $|x| \leq b - 2$, a unique padding technique is employed by appending x with a compulsory bit 1 followed by necessary 0 bits to make the size of x equal to the size of b . In this case, the last bit is always set to 1 and the concatenation of 0 bits depends on whether $|x| < b - 2$.

The last bit indicates whether ENMAC is used to process single message block. For example, to authenticate a short message say 500 bits, ENMAC based on SHA-1 requires just one call to the compression function f of SHA-1 whereas NMAC requires three calls and NMAC-1 requires two calls to the function f .

When $|x| > b - 2$, the message x is split into two parts, prefix (x_{pref}) and suffix (x_{suff}). ENMAC first processes the x_{suff} using the internal function F_{k_2} and then the prefix part x_{pref} using the output of $F_{k_2}(x_{suff})$. For example, see SHA-1-ENMAC discussed in [25]. In this case, if x_{suff} begins at a non-word border, all words in x_{suff} need to be re-aligned. To overcome such problems, a practical variant for ENMAC was proposed in [25] and is defined as follows:

$$\text{ENMAC}_k(x) = \begin{cases} f_{k_1}(x) \text{ with the input } x = x || 11 & \text{if } |x| = b - 2 \\ f_{k_1}(x) \text{ with the input } x = x || 10 \dots 01 & \text{if } |x| < b - 2 \\ f_{k_1}(F_{k_2}(x_{pref}), x_{suff}, 0) & \text{if } |x| > b - 2, \end{cases}$$

where $x_{pref} = x^1, \dots, x^{|x|-(b-n-1)}$ and $x_{suff} = x^{|x|-(b-n)}, \dots, x^{|x|}$.

Assume³ a standard padding technique employed for both x_{pref} and x_{suff} to obtain a secure MAC function ENMAC. Now this variant of ENMAC requires knowledge of length of x to calculate x_{pref} and x_{suff} as length of the message x determines the content in the last two blocks of x_{pref} where the last block of x_{pref} is the padded block. The knowledge of the length of the message x is required even if the message x is padded with a bit 1 followed by 0's as the specification for x_{pref} has $|x|$ as an argument. In general, the length of data to be hashed is known ahead of time in many applications and in rare situations it is not [8].

In general, any MAC function based on a hash function used to protect the authenticity and integrity of the communicated data does not know the length of the message in advance. However, a machine evaluating the ENMAC function to generate authentication codes for the communicated data, must know the length of x in advance to find the content of x_{pref} . The machine may also perform the ENMAC computation as follows: when it receives the authentication tag attached to the message (specifically, to the last block), it has to look at the intermediate MAC value obtained at the previous one or two blocks before the final block of x and has to re-compute x_{pref} accordingly taking into account the padding for x_{pref} . Then it has to evaluate the outer function of ENMAC. In this case, when ENMAC is used to process large data, there would be a slight performance inefficiency due to the above process. This problem can be solved using a special code [8] to tell the ENMAC routine that the total length of x_{pref} is not known when the processing of data is begun and that it will be input with the final chunk of data for x_{pref} and the x_{suff} follows x_{pref} . The special code used for

³The padding employed on x_{pref} and x_{suff} is not specified in the specification of ENMAC or its variant in [25]

this purpose needs to be clearly different from the valid total length code which appears in the last block so that correct processing of ENMAC function can be done.

We note that prior knowledge of the message or any special code is not essential to evaluate tags on messages of more than a block using NMAC, HMAC and NMAC-1 functions. They only require to know the length encoding of the message in the last block (if length encoding is used as part of the padding) or the authentication tag attached to the last block to indicate the end of data stream for that particular session in the communication channel.

The security treatment of NMAC and NMAC-1 take into account the maximum length L of the message to be processed as a security parameter. During the chosen message or adaptive chosen message attack on any of these functions, their respective oracles return response to a query of arbitrary length in one step. However, this is not realistic measure. Hence, it is reasonable to assume that the processing time of a MAC is proportional to the length of the message and the length of the message to be processed must be a security parameter. We observe that the analysis of ENMAC and its variants do not consider length of the message as a security parameter.

As observed in [2], one can convert NMAC into a hybrid MAC function using two different functions as long as the assumptions stated in the proof of security of NMAC hold. This holds for NMAC-1 too. For example, one can use SHA-256 for the internal function of NMAC-1 and a block cipher in the CBC mode for the external function of NMAC-1. In this sense, the result of NMAC-1 is more general than stated in the proof. In addition, one can use the wide-pipe hash [20] (e.g, SHA-512 and truncating half of the output bits) for the inner function and the compression function of SHA-256 for the external function in NMAC and NMAC-1. We note that design of such hybrid schemes using ENMAC poses penalty on the performance as the second function used in ENMAC not only uses the output of the first function but also part of the message to be authenticated.

Finally, the ISO/IEC 9797-2 [16] standard specifies a mechanism which is a variant of MDx-MAC [27] that offers high performance for applications that process short messages of upto 256 bits. MDx-MAC unlike NMAC and its variants, calls complete hash function once but it makes a small modification to the compression function by adding a key to the additive constants in the compression function. In addition, MDx-MAC and its variant were not formally analysed as the analysis of NMAC and its variants.

5 A New Variant of NMAC

As pointed out in [2], keying the IVs of the hash functions as was done in NMAC and HMAC functions allows for a better modeling of the keyed hash functions with some significant analytical advantages. However, the question is how it differs from using the keys as data blocks which is

the other common way of designing keyed hash functions. In this section, we shall explore this concept further and see what advantages we can get by doing so.

Instead of using the second key k_1 as an IV to the external function f in NMAC, we use it as a block and use the output of F_{k_2} as the IV to the external function f . We call this variant as M-NMAC which shall mean modified-NMAC as shown in Figure 3 and is defined as below:

$$\text{M-NMAC}_k(x) = f_{F_{k_2}(x)}(\overline{k_1}).$$

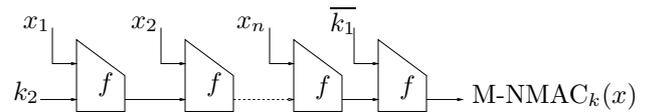


Figure 3: The M-NMAC construction

M-NMAC can also be seen as a kind of envelope MAC scheme [27, 28, 29] except that it uses the key k_2 as an IV instead of as a block. $\overline{k_1}$ denotes the key k_1 made to a block size b of the compression function f . That is, if the function f is the compression function of SHA-1 then the length $|k_1|$ of the key k_1 is at most 512 bits. It is recommended that $|k_1| \geq |k_2|$ and k_1 is completed to size b by appending 0's if $|k_1| < b$. The length of message x after padding must be a multiple of block length b of the compression function f as in NMAC. The proof of security of M-NMAC follows from NMAC under the assumption that the keyed function F_{k_2} is a weakly collision resistant hash function and the external function $f_y(\overline{k_1})$ is a secure MAC where $y = F_{k_2}(x)$.

We note that birthday attack is the best known forgery attack on M-NMAC. The advantage of M-NMAC over NMAC is the flexibility in using variable key lengths as large as size b of the block for the trail key k_1 . Note that HMAC has also the provision of using larger keys upto size of the block b . While the maximum lengths of both the keys k_1 and k_2 in NMAC depends on the sizes of the initial states of the functions F and f used in NMAC, *only* size of the key k_2 is dependent on the initial state of the function F . The total complexity of divide and conquer complete key recovery attack [27, 28, 29] on M-NMAC (which applies to NMAC [2]) is about $2^{|k_1|} + 2^{|k_2|}$. If $|k_1| > |k_2|$, then this total cost approximates to $2^{|k_1|}$. Since the key k_1 is used in a separate block independent of the message, the slice by slice trail key recovery attack [28, 29] does not work against M-NMAC as this attack requires the trail key to be split across the blocks. Anyhow, due to the divide and conquer key recovery attack on M-NMAC, once the attacker finds the key k_2 , the security of M-NMAC against forgery, like NMAC, reduces to the secret prefix MAC scheme [27].

6 On the Pseudorandomness of NMAC and HMAC

It is well known that any pseudorandom function would work as a MAC and the security reduction is standard [4, 5, 11, 12]. However, it is not the other way round. A MAC function may not work as a PRF. Nevertheless, HMAC with SHA-1 is used as pseudorandom function to derive keys in applications such as PKCS #5 [18] and IPsec's Key Exchange (IKE) protocol. Though it has been pointed out that [18] security analysis given for HMAC as a MAC function [2] can be modified to accommodate the requirements of a PRF using strong security assumptions, no explicit security analysis of NMAC and HMAC as PRFs has appeared in the literature. In this section, we provide the security analysis of NMAC as a PRF, and it applies to HMAC as well.

6.1 Security Analysis

The terminology used in this section shall be the same as those used in [2, 3] for the sake of clarity. The analytical result of NMAC as a PRF is given referring to chosen or adaptive chosen message attacks. The result uses the definition of weakly collision resistant hash function given in Section 2. In addition, the analysis uses the definition on fixed-length input-pseudorandom function (FI-PRF) families following [3].

Informally, a pseudorandom function is a family of functions with the property that the input-output behavior of a random member of the family is computationally indistinguishable from that of a random function. Let $F' : \{0, 1\}^b \times \{0, 1\}^l \rightarrow \{0, 1\}^l$ be a family of keyed compression functions or FI-PRF family where l is the length of the key k . That is, there is a set of keys and each key k of length l names a function from the family F' . This is denoted by F'_k or f . Let $R : \{0, 1\}^b \rightarrow \{0, 1\}^l$ be a family of all functions with the distribution being uniform; that is picking a function at random from this family just means drawing a random function of $\{0, 1\}^b$ to $\{0, 1\}^l$. If S is a probability space then picking a string x from S is denoted by $x \stackrel{\$}{\leftarrow} S$.

Now F' is said to be a pseudorandom function if the input-output behavior of a random member of this family is computationally indistinguishable from the behavior of a function picked at random from the family R . This is formalized using the notion of distinguishers [3]. A distinguisher is given an oracle for the function f chosen at random from one of the two families and is allowed to decide the family from which f is chosen. Formally, to any such distinguisher a number between 0 and 1 called *prf-advantage* is associated and is defined as,

$$\text{Adv}_{F'}^{\text{prf}}(D) = \Pr_{f \stackrel{\$}{\leftarrow} F'}[D^f = 1] - \Pr_{f \stackrel{\$}{\leftarrow} R}[D^f = 1]$$

with the probabilities taken over the choices of f and the coin tosses of D .

The security of the family F' as a pseudorandom function depends on the resources that D uses that include running-time and number and length of oracle queries. The running-time t includes the time taken to execute $f \stackrel{\$}{\leftarrow} F'$, time taken to compute responses to oracle queries made by D and the memory which includes the size of the description of D . The distinguisher $D(t, q, b, \epsilon^*)$ distinguishes F' from R if it runs for time t , makes q oracle queries each of block of length b bits and $\text{Adv}_{F'}^{\text{prf}}(D) \leq \epsilon^*$ where ϵ^* is called the distinguishing probability. This is defined below:

Definition 4. A family of fixed-length input keyed compression functions $\{F'_t\}$ is (ϵ^*, q, t, b) -FI-PRFs if any distinguisher that is not given the key k , is limited to spend total time t and sees the outputs of the given function f computed on q distinct inputs each of size b bits, cannot distinguish the function f from a random function of the family R except with a probability less than ϵ^* .

Now we state the main analytical result on NMAC as a PRF. The analysis uses Definitions 2 and 4.

Theorem 3. Suppose $F' : \{0, 1\}^b \times \{0, 1\}^l \rightarrow \{0, 1\}^l$ be a fixed-length input function family where l is the length of the key. Suppose F' is an (ϵ^*, q, t, b) -pseudorandom on inputs of length b bits. Let $\{F_k\}$ be a family of (ϵ_F, q, t, L) -weakly collision resistant keyed hash functions where $L \geq b$. Let $NMAC : \{0, 1\}^L \times \{0, 1\}^l \rightarrow \{0, 1\}^l$ be a function family where l is the length of each key k_1 and k_2 and $L \geq b$. Then the NMAC function is $(\epsilon^* + \epsilon_F, q, t, L)$ -pseudorandom.

Proof. The parameters q, t and L for the number of queries to the NMAC oracle, the total attack time t and the maximum length L of each finite length message x_i (where $i = 1, 2, \dots, q$) to be queried are fixed.

Let $R : \{0, 1\}^b \times \{0, 1\}^l \rightarrow \{0, 1\}^l$ be a family of all functions with a uniform distribution. Let $G : \{0, 1\}^L \rightarrow \{0, 1\}^l$ be a family of all functions with a uniform distribution.

Let A_N be the distinguisher that tries to break NMAC as a PRF. Namely, A_N is given an oracle of the function g chosen at random from one of the two families; NMAC or G . A_N 's task is to distinguish the function g from a random function. A_N queries the oracle g with x_i and gets the response $g(x_i)$ for every queried message x_i where i ranges from 1 to q . Finally, A_N succeeds if it distinguishes NMAC from G after looking at q input-output examples of g in time t . Let ϵ_N be the probability of success of A_N . After querying the oracle of the function g with q queries, A_N outputs a bit 0 or 1. The output is 1 if A_N succeeds in correctly distinguishing NMAC from G and the output is 0 if A_N fails in distinguishing NMAC from G .

Using A_N , we build an attacker A_f that aims to tell whether the function f belongs to F' or R on inputs of messages of length b by sending q queries to the oracle of

the function f in time t with a maximum probability of ϵ^* . That is, the goal of A_f is to distinguish the family F' from the random function family R .

The algorithm of A_f is given below:

Choose random k_2

For $i = 1, \dots, q$ perform the following steps:

- 1) $A_N \rightarrow x_i$
- 2) A_f computes $F_{k_2}(x_i)$
- 3) A_f queries f with $\overline{F_{k_2}(x_i)}$ and gets $f(\overline{F_{k_2}(x_i)})$ where $\overline{F_{k_2}(x_i)}$ denotes the padding of $F_{k_2}(x_i)$.
- 4) $A_N \leftarrow f(\overline{F_{k_2}(x_i)})$

A_N outputs its decision, a bit 0 or 1.

A_f outputs its decision, a bit 0 or 1.

Analysis of success probabilities

Now we analyze the success probability ϵ^* of the distinguisher A_f in distinguishing f_{k_1} from a truly random function. Let ϵ_F be the maximum probability that there exists at least one collision in F_{k_2} when A_N queries the NMAC function. The distinguisher A_f fails:

- 1) whenever A_N fails to distinguish NMAC from G and outputs a bit 0.
- 2) whenever A_N outputs a bit 1 and the inner function F_{k_2} is not weakly collision resistant i.e $F_{k_2}(x_p) = F_{k_2}(x_q)$ for distinct p and q . In this case, the answer of A_N is not useful to tell whether the outer function f_{k_1} belongs to F' or R because $g(x_p) = g(x_q)$ for both f_{k_1} and a truly random function.

If F_{k_2} is weakly collision resistant then the probability that there exists p and q such that $F_{k_2}(x_p) = F_{k_2}(x_q)$ is negligible. Then, the answer of A_N is useful to tell whether the outer function is f_{k_1} or a truly random function because the inputs to the outer function are almost always distinct.

Hence, $\Pr[A_f \text{ fails}] \leq \Pr[A_N \text{ fails}] + \Pr[A_N \text{ succeeds} \wedge \exists p, q (p \neq q \wedge F_{k_2}(x_p) = F_{k_2}(x_q))]$

$$\Rightarrow 1 - \epsilon^* \leq 1 - \epsilon_N + \epsilon_F$$

$$\Rightarrow \epsilon_N \leq \epsilon^* + \epsilon_F$$

Hence, the probability of breaking NMAC as a PRF is at most sum of the maximum probabilities of finding collisions for the inner keyed compression function and distinguishing the outer function from that of a random function. \square

Remarks:

- 1) Let $\epsilon_{max} = \max(\epsilon^*, \epsilon_F)$

$$\Rightarrow \epsilon_N \leq \epsilon_{max} + \epsilon_{max}$$

$$\Rightarrow \epsilon_N \leq 2 \cdot \epsilon_{max}$$

$$\Rightarrow \epsilon_{max} \geq (1/2) \cdot \epsilon_N.$$

That is, given an adversary that distinguishes the NMAC function from a random function, one can explicitly show an algorithm that using the same resources breaks the underlying hash function with at least half of that probability.

7 Conclusion

In this paper, we have considered some issues in the design and analysis of NMAC and HMAC functions that were not covered in [2]. The first result of this paper is analysis of these MAC functions using *weaker assumptions* than stated in their proofs of security. Next, we have proposed an efficient variant to NMAC called NMAC-1 which has some advantages over other efficient MACs based on hash functions. We have also proposed a variant to the NMAC function which has some additional advantages over NMAC from the perspective of complete key-recovery attack. Finally, we have formally analysed the pseudorandomness of NMAC and HMAC functions.

Acknowledgments

The authors thank Kapali Viswanathan (Technology Development Department, ABB Corporate Research Centre, Bangalore, India) for all his encouragement and useful discussions on the subject in the paper. Many thanks to William Millan and Juanma González Nieto of ISI, QUT for their comments on the earlier drafts of the paper. We also thank anonymous reviewers for useful comments that improved the quality of the paper.

References

- [1] M. Bellare, "New proofs for NMAC and HMAC: Security without collision-resistance," in *Advances in Cryptology (Crypto'06)*, colored pp. ???, 2006.
- [2] M. Bellare, R. Canetti, and H. Krawczyk, "Keying hash functions for message authentication," in *Advances in Cryptology (Crypto'96)*, LNCS 1109, pp. 1-15, 1996.
- [3] M. Bellare, R. Canetti, and H. Krawczyk, "Pseudorandom functions revisited: The cascade construction and its concrete security," in *Proceedings of the 37th Annual IEEE Symposium on Foundations of Computer Science (FOCS'96)*, pp. 514–523, 1996.
- [4] M. Bellare, J. Kilian, and P. Rogaway, "The security of cipher block chaining," in *Advances in Cryptology (Crypto'94)*, LNCS 839, pp. 341-358, Springer Verlag, 1994.
- [5] M. Bellare, J. Kilian, and P. Rogaway, "The security of the cipher block chaining message authentication code," *Journal of Computer and System Sciences*, vol. 61. no.3, pp. 362-399, 2000.

- [6] J. Coron, Y. Dodis, C. Malinaud, and P. Puniya, “Merkle-damgard revisited: How to construct a hash function,” in *Advances in Cryptology (Crypto’05)*, LNCS 3621, pp. 430-448, 2005.
- [7] I. Damgard, “A design principle for hash functions,” in *Advances in Cryptology (Crypto’89)*, LNCS 435, pp. 416-427, 1989.
- [8] D. Johnson, *Improving Hash Function Padding*, Technical report, National Institute of Standards and Technology NIST, Oct. 2005. (<http://www.csrc.nist.gov/pki/HashWorkshop/2005/program.htm>).
- [9] “Federal Information Processing Standard (FIPS),” *Secure Hash Standard*, National Institute for Standards and Technology, Aug. 2002. (<http://csrc.nist.gov/publications/fips/fips180-2/fips180-2.pdf>)
- [10] N. Ferguson and B. Schneier, *Practical Cryptography, Chapter Hash Functions*, pp. 83-96, John Wiley & Sons, 2003.
- [11] O. Goldreich, S. Goldwasser, and S. Micali, “On the cryptographic applications of random functions (extended abstract),” in *Advances in Cryptology (Crypto’84)*, LNCS 196, pp. 276-288, Springer-Verlag, 1984.
- [12] O. Goldreich, S. Goldwasser, and S. Micali, “How to construct random functions,” *Journal of the ACM*, vol. 33, no. 4, pp. 792–807, Oct. 1986.
- [13] S. Hirose, “A note on the strength of weak collision resistance,” *IEICE Transactions on Fundamentals*, vol. E87-A, no. 5, pp. 1092-1097, 2004.
- [14] ISO/IEC FDIS 10118-3, *Information Technology - Security Techniques - Hash Functions - Part 3: Dedicated hash functions*, 2003.
- [15] ISO 15764, “Road Vehicles Extended Data Link Security,” International Organization for Standardization, 2004. Published Standard.
- [16] ISO/IEC 9797-2, “Information Technology - Security Techniques - Message Authentication Codes (MACs) - Part 2: Mechanisms Using A Dedicated Hash-Function,” International Organization for Standardization, Aug. 2002.
- [17] B. Kaliski and M. Robshaw, “Message authentication with MD5,” *CryptoBytes*, vol. 1, no. 1, pp. 5-8, Spring 1995.
- [18] B. Kaliski, PKCS #5: Password-based cryptography specification version 2.0. Internet Informational RFC 2898, Sept. 2000. (<http://www.ietf.org/rfc/rfc2898.txt>)
- [19] T. Krovetz, *Software-Optimized Universal Hashing and Message Authentication*, PhD thesis, University of California, Davis, Sep. 2000.
- [20] S. Lucks, “A failure-friendly design principle for hash functions,” in *Advances in Cryptology (Asiacrypt’05)*, LNCS 3788, pp. 474-494, Springer-Verlag, 2005.
- [21] A. Menezes, P. Oorschot, and S. Vanstone, *Handbook of Applied Cryptography, Chapter Hash Functions and Data Integrity*, The CRC Press series on discrete mathematics and its applications, pp. 321-383, CRC Press, 1997.
- [22] R. Merkle, “One way hash functions and DES,” in *Advances in Cryptology (Crypto’89)*, LNCS 435, pp. 428-446, Springer-Verlag, 1989.
- [23] P. Metzger and W. Simpson, *RFC 1828: IP Authentication Using Keyed MD5*, Aug. 1995. Status: PROPOSED STANDARD.
- [24] C. Mitchell, Personal Communication, Aug. 2005.
- [25] S. Patel, “An efficient MAC for short messages,” in *Annual International Workshop on Selected Areas in Cryptography*, LNCS 2595, pp. 353-368, Springer-Verlag, 2002.
- [26] B. Preneel, *Analysis and design of Cryptographic Hash Functions*. PhD thesis, Katholieke Universiteit Leuven, 1993.
- [27] B. Preneel and P. Oorschot, “MDx-MAC and building fast MACs from hash functions,” in *Advances in Cryptology (Crypto’95)*, LNCS 963, pp. 1-14, Springer-Verlag, 1995.
- [28] B. Preneel and P. Oorschot, “On the security of two MAC algorithms,” in *Advances in Cryptology (EuroCrypt’96)*, LNCS 1070, pp. 19-32, Springer-Verlag, 1996.
- [29] B. Preneel and P. Oorschot, “On the security of iterated message authentication codes,” *IEEE Transactions on Information Theory*, vol. 45, no. 1, pp. 188-199, 1999.
- [30] G. Tsudik, “Message authentication with one-way hash functions,” in *IEEE Infocom 1992*, pp. 2055-2059, 1992.
- [31] X. Wang, Y. Yin, and H. Yu, “Finding collisions in the full SHA-1,” in *Advances in Cryptology (Crypto’05)*, LNCS 3621, pp. 17-36, Springer-Verlag, 2005.
- [32] X. Wang and A. Yao and F. Yao, *Cryptanalysis of SHA-1 Hash Function*, Technical report, National Institute of Standards and Technology NIST, Oct. 2005. (<http://www.csrc.nist.gov/pki/HashWorkshop/program.htm>)

Appendix A: Application based on HMAC-SHA-1

Consider the vehicle to remote database application of ISO 15764 (Road Vehicles: Extended Data Link Security) [15]. This application uses HMAC-SHA-1 for entity authentication and data integrity security services to prevent replay attacks. The link between the vehicle and the external test equipment is local to the terminal. It is therefore under the control of the user who will be identified to the server before secure information is processed. The communication chain requires the following security services: entity authentication and data integrity to prevent replay attack, confidentiality to prevent eavesdropping and non-repudiation to prevent the user later denying establishing the link, thereby linking the user to the

audit trail. If the HMAC-SHA-1 for authentication and integrity services is not one-way and collision resistant for the user knowing the key, then signing the final message with the RSA private key will not provide non-repudiation for the entire message stream [24]. Considering the recent attacks on SHA-1 [31, 32], one would require the collision resistance property of the underlying hash function in the MAC schemes like HMAC and NMAC to provide additional protection to the application against insiders. Note that this is not the motivation behind the proposals NMAC and HMAC. Nevertheless, as pointed out by Preneel [26, p.20], if some additional protection against insider attacks is obtained from the protection of the MAC, the MAC function must be naturally collision resistant.

Appendix B: Performance aspects

In this section, we give performance comparison of NMAC-1 against NMAC in terms of number of calls to the compression function for various message sizes when hash functions from the MDx family are chosen as the underlying algorithms for NMAC and NMAC-1. A similar performance comparison between NMAC and ENMAC is given in [25].

The efficiency improvement of NMAC-1 over NMAC is calculated using the formula.

$$\% \text{Efficiency improvement} = [(\#(f) \text{ in NMAC} - \#(f) \text{ in NMAC-1}) / \#(f) \text{ in NMAC}] 100$$

From Table 1, we can clearly see that for messages of exactly one block length ($b = 512$ bits) and those that fall in the closed set $[n * 447, n * b]$ and with $n = 1$ ($n \in \{1, 2, \dots\}$), the efficiency of NMAC-1 is 33% over NMAC. The efficiency improvement decreases for the messages falling in the set $[n * 447, n * b]$ and with $n \geq 2$.

Appendix C: Survey of attacks on MACs based on hash functions

The following are some known generic attacks on MAC schemes [27, 21]. Here, the secret key k is used in computing an n -bit MAC value h for a message x . It is assumed that the adversary that attacks the MAC function does not possess the secret key k .

- **MAC forgery:** An adversary generates a new message-MAC pair (x, h) such that $\text{MAC}_k(x) = h$. If the message x is an arbitrary message then this is an *existential forgery*. If the message x is a particularly chosen message then this is a *selective forgery*. For an ideal MAC, the complexity of these attacks is $O(2^{\min(|k|, n)})$. Either guessing h for a given x or guessing x for a given h , has a success probability of 2^{-n} . It should be noted that the adversary may somehow, possibly by interacting with the sender or the receiver, determines the validity of the forged

(x, h) pairs. Of course, the adversary cannot verify the forged pairs even with known text-MAC pairs without interacting with the sender or receiver. A formal definition on MAC security is given in section 4.

- **Key recovery:** Using a single known text-MAC pair, an attacker finds the correct key k used in computing the MAC value. For an ideal MAC, the complexity of the key recovery attack should be same as the exhaustive key search attack over the entire key space which is $O(2^{|k|})$. It requires $|k|/n$ text-MAC pairs to verify this attack. Key recovery attack allows selective forgery of the MAC function.

Attacks on different MAC schemes based on cryptographic hash functions are presented below. It should be noted that the hash function F used in constructing MACs follows the Merkle-Damgård construction [7, 22]. The function F processes a message of arbitrary finite length in successive blocks of fixed equal length using the compression function f . It is assumed that the length of the chaining value, hash value and the MAC value (authentication tag) is n bits. For the working procedure on Merkle-Damgård hash functions see [2, 21].

- **Attack against the secret prefix method:** In the secret prefix method, the secret key k is prepended to the message for which MAC has to be computed [27, 30]. MAC computed on a message x using this method is given as $\text{MAC}_k(x) = F(k||x)$. This scheme is weak against extension attacks as one can use this MAC value to compute the MAC of a new message $x||x'$ by appending x' to x . The iterative structure of F allows extension attacks to happen. Moreover, any type of padding scheme employed for x initially do not prevent extension attacks as an attacker can cleverly choose x' related to the length of x and its padding.
- **Attack against the secret suffix method:** $\text{MAC}_k(x) = F(x||k)$ is the secret suffix method [30, 27] where key k is appended to the message x . An off-line collision attack against the function F can result in two messages x and x' such that $F(x) = F(x')$. For an ideal hash function F with an n -bit hash value, this requires at least $2^{n/2}$ off-line operations according to the Birthday attack. Once this is found, the attacker can append text y to x and get the chosen text $x||y$ and request the MAC_k function to get $\text{MAC}_k(x||y||k)$. Using the MAC value $\text{MAC}_k(x||y||k)$, the attacker can perform selective forgery (forgery for the message of its choice) on the message $x'||y||k$ to get $\text{MAC}_k(x'||y||k)$. Again the attacker does not have to know the secret key k to perform this attack as it is appending of the key k to the message that results in the attack.
- **Attacks against the Envelope method:** Envelope method is a combination of prefix and suffix

Table 1: Efficiency improvement of NMAC-1 with respect to NMAC

x in 32 byte increments	$\#(f)$ in NMAC	$\#(f)$ in NMAC-1	%Efficiency improvement
32	2	2	0
64	3	2	33
96	3	3	0
128	4	3	25
160	4	4	0
192	5	4	20
224	5	5	0
256	6	5	17

methods. In this method, one key k_1 is prepended to the message x and the other key k_2 is appended to the message x as given by $MAC_k(x) = F(k_1 || x || k_2)$.

Preneel and van Oorschot [2, 27, 28, 29] observed *divide and conquer exhaustive search key recovery attack* on such a scheme where both keys k_1 and k_2 can be recovered in a time around $(2^{|k_1|} + 2^{|k_2|})$ (where $|k_1| = |k_2| = n$) by first recovering key k_1 and then k_2 . This attack needs about $2^{(n+1)/2}$ known message-MAC pairs of equal length to find an internal collision (collision before last block gets hashed) on the chaining variables. The attacker then performs an exhaustive key search for k_1 with the effort about $2^{|k_1|}$ which results in a small set of possible keys for k_1 and determines the correct key k_1 with a few chosen messages thus reducing the security of the envelope method to the secret suffix method. The attacker then finds key k_2 with the effort $2^{|k_2|}$.

Preneel and van Oorschot [28] described a new divide and conquer key recovery attack to recover the trailing key k_2 in the envelope method. This is also called as *slice by slice key recovery of trail key in envelope method* [29]. This attack includes the case $k_1 = k_2$ of the envelope method which was proposed in RFC 1828 [23] and in [17]. This attack exploits the padding procedure of the hash functions such as MD5 used in the envelope scheme. This attack relies on the trial key k_2 being split across the blocks. For example, to find 64 bits of a 128-bit key k_2 in 4-bit slices (2^4 steps), the attack requires $2^{64.5}$ known texts at each step to get an internal collision (in total $2^{68.5}$ known texts) and 2^6 chosen texts at each step to identify the correct key bits (in total 2^{10} chosen texts). The attack works when the last before block contains 1 to 64 bits of the key k_2 and known messages have the same number of blocks. Exhaustive key search is then performed to recover the remaining 64 key bits of k_2 . Hence the overall time complexity of the attack to recover 128-bit key k_2 in 4-bit slices is on the order of $2^{68.5}$ known text-MAC pairs instead of 2^{128} . Finding key k_1 then takes $2^{|k_1|}$ effort. But if one knows k_2 , the security of the envelope scheme reduces to secret prefix method.

Praveen Gauravaram holds a Bachelor of Technology (Electrical and Electronics Engineering) degree from Sri Venkateswara University College of Engineering (SVUCE), Tirupati, India, and a Master of Information Technology (MIT) degree from Queensland University of Technology (QUT), Australia. He is currently working as a research associate at Information Security Institute (ISI), QUT. He has recently completed his doctoral dissertation (PhD) in the area of cryptanalysis, design and applications of cryptographic hash functions. His current research interests include analysis and design of cryptographic primitives, applications of cryptography and information security. His home page is available at <http://www.isi.qut.edu.au/people/subramap>.

Shoichi Hirose received the B.E., M.E. and D.E. degrees in information science from Kyoto University, Kyoto, Japan, in 1988, 1990 and 1995, respectively. From 1990 to 1998, he was a research associate at Faculty of Engineering, Kyoto University. From 1998 to 2005, he was a lecturer at Graduate School of Informatics, Kyoto University. From 2005, he is an associate professor at Faculty of Engineering, The University of Fukui. His current interests include cryptography, information security and computational complexity.

Suganya Annadurai holds Masters degree in VLSI Design from SASTRA University and Bachelors in Computer Science and Engineering degree from Annamalai University, Tamil Nadu, India. She has joined Society for Electronic Transactions and Security (SETS) in June 2004, where she is currently working as Research Associate. Her research interests include analysis and design aspects of cryptographic hash functions and efficient hardware realisation of secure communication systems.