On Efficient Message Authentication Via Block Cipher Design Techniques

G. Jakimoski and K. P. Subbalakshmi*

Department of Electrical and Computer Engineering
Stevens Institute of Technology, Hoboken, NJ 07030, USA
goce.jakimoski@stevens.edu, ksubbala@stevens.edu

Abstract. In an effort to design a MAC scheme that is built using block cipher components and runs faster than the modes of operation for message authentication, Daemen and Rijmen have proposed a generic MAC construction ALRED and a concrete ALRED instance Pelican. The Pelican MAC uses four rounds of AES as a building block to compute the authentication tag in a CBC-like manner. It is about 2.5 times faster than a CBC-MAC with AES, but it is not proven secure. Minematsu and Tsunoo observed that one can build almost universal (AU_2) hash functions using differentially uniform permutations (e.g., four AES rounds with independent keys), and hence, provably secure MAC schemes as well. They proposed two MAC schemes MT-MAC and PC-MAC. MT-MAC hashes the message using a Wegman-Carter binary tree. Its speedup for long messages approaches 2.5, but it is not very memory efficient. PC-MAC hashes the message in a CBC-like manner. It is more memory efficient. However, its speedup over the message authentication modes is about 1.4.

We notice that using a non-linear permutation as a building block, one can construct almost XOR universal ($\mathrm{A}XU_2$) hash functions whose security is close to the maximum differential probability of the underlying non-linear permutation. Hence, using four AES rounds as a building block will lead to efficient Wegman-Carter MAC schemes that offer much better security than the modes of operation for message authentication. If the target security is that of the message authentication modes with AES, then one can use non-linear permutations defined on 64-bit blocks and achieve greater speedup and better key agility. For instance, the ideally achievable speedup when using the 64-bit components we suggest is 3.3 to 5.0 as opposed to the 2.5 speedup when using four AES rounds.

Keywords: message authentication, Wegman-Carter construction, universal hash functions, block ciphers, maximum differential probability

1 Introduction

MESSAGE AUTHENTICATION. Message authentication is one of the basic information security goals, and it addresses the issues of source corroboration

* This work was funded in part by NSF CT grant number: 0627688 and US Army ARDEC/Picattiny Arsenal.

and improper or unauthorized modification of data. The message authentication model usually involves three participants: a sender, a receiver and an adversary. The sender and the receiver have agreed on a secret key. Prior to sending a message, the sender uses a signing algorithm that given the message and the secret key outputs an authentication tag (or MAC). The sender sends the tag along with the message to the receiver. On receipt, the receiver uses a verification algorithm that given the secret key, the message and the tag returns 1 if the MAC is valid, or returns 0 otherwise. The goal of the adversary is to trick the receiver into accepting a message that was not sent by the sender.

Message authentication has been heavily addressed in the literature. We briefly overview some of the results. There are three common approaches to message authentication. One approach involves using cryptographic hash functions. The first such schemes were proposed by Tsudik [49] and Kaliski and Robshaw [29], and later analyzed by Preneel and Van Oorschot [44, 45]. A popular hash function based MAC is the HMAC construction of Bellare, Canetti and Krawczyk [3, 5].

Another approach to message authentication involves secure block ciphers modeled as pseudorandom permutations. The CBC MAC [20, 25] is probably the most studied MAC construction based on block ciphers. Bellare, Kilian and Rogaway proved its security for fixed-length messages [2]. Petrank and Rackoff [43] (another proof was provided by Vaudenay [50]) showed that EMAC [6], a CBC MAC variant using additional encryption, is secure when the message length is a multiple of the block size. Black and Rogaway [13] proposed a solution for arbitrary message lengths that uses three keys and only one key scheduling of the underlying block cipher. Jaulmes, Joux and Valette proposed RMAC [27], which is an extension of EMAC using two keys and a randomness. Iwata and Kurosawa provided solutions that use only two [37] and one key [27]. There are also block cipher based MAC constructions that do not follow the CBC paradigm (e.g., the PMAC construction of Black and Rogaway [14]).

The third approach is the universal hash function approach. Wegman and Carter were the first to propose the notion of universal hash functions [16] and their use in message authentication [51]. The construction proposed by Wegman and Carter provides unconditional security. A computationally secure scheme can be obtained if the random keys are replaced by pseudorandom keys. This approach was first studied by Brassard [15]. The topics related to universal hash functions and unconditional message authentication have been studied a lot in the past years. Some of the results include the following. Unconditional message authentication was first considered by Gilbert, Williams and Sloane [22]. Simmons [47] developed the theory of unconditional authentication and derived some lower bounds on the deception probability. The use of universal hashing to construct unconditionally secure authentication codes has also been studied by Stinson [48] and by Bierbrauer et al.[9]. The notion of almost XOR universal hash functions is due to Krawczyk [29]. A bucket hashing technique for constructing an AXU_2 families of universal hash functions and their use to construct computationally secure MACs were proposed by Rogaway [46]. Afanassiev, Gehrmann and Smeets [1] proposed an efficient procedure for polynomial evaluation that can be used for fast message authentication. MMH, proposed by Halevi and Krawczyk [23], and SquareHash, proposed by Etzel, Patel and Ramzan [19], are examples of fast universal hash functions. An efficient universal hash function family NH and a message authentication code UMAC based on NH were also proposed by Black et al.[12]. Another fast message authentication scheme and stronger bounds for Wegman-Carter-Shoup authenticators were recently provided by Bernstein [8, 7].

DIFFERENTIAL PROBABILITY BOUNDS. Since the publication of the differential cryptanalysis attacks on DES (Biham and Shamir [10]), differential cryptanalysis has become one of the most studied general attacks on block ciphers, and the resistance to differential cryptanalysis has become one of the basic block cipher design criteria. The round keys used by block ciphers are derived from a single key using a key scheduling algorithm. However, in order to augment the belief that certain block cipher structures are secure against differential cryptanalysis, some researchers have provided security proofs assuming random and independent round keys. The provable security against differential cryptanalysis of some Feistel structures has been studied by Matsui [38]. Hong et al. [24] proved an upper bound on the maximum differential probability for 2 rounds of a substitution permutation network with highly diffusive linear transformation. Kang et al. [30] provided a bound for any value of the branch number of the linear transformation. Keliher, Meijer and Tavares [31, 32] proposed a new method for finding the upper bound on the maximum average linear hull probability for substitution permutation networks (SPN) and applied their method to AES. Park et al. proved that the maximum differential probability of four rounds of AES is upper bounded by 1.06×2^{-96} [41], and later proved a better bound 1.144×2^{-111} in [42]. A slightly better bound (2^{-113}) was provided by Keliher and Sui [33].

CLOSELY RELATED WORK. Daemen and Rijmen [17] have recently proposed a new heuristic MAC construction ALRED, and a concrete MAC scheme Pelican [18]. The Pelican MAC uses four rounds of AES as a building block to compute the authentication tag in a CBC-like manner, and it is about 2.5 times faster than a CBC-MAC with AES. However, it is not proven secure. Minematsu and Tsunoo [40] observe that one can obtain provably secure almost universal hash functions (AU_2) by using differentially uniform permutations such as four rounds of AES with independent keys in a Wegman-Carter binary tree. They also propose a message authentication scheme MT-MAC that makes use of the proposed AU_2 hash function. However, they note that such construction is not memory efficient, and suggest a CBC-like AU_2 hash PCH (Periodic CBC Hash) and a proven secure MAC scheme PC-MAC based on PCH. The speedup of PC-MAC over the modes with AES is 1.4.

OUR CONTRIBUTION. We propose a CBC-like $\mathrm{A}XU_2$ hash UHC (Universal Hash Chaining) and a variant of a Wegman-Carter binary tree $\mathrm{A}XU_2$ hash (the MACH hash). Both constructions use a non-linear invertible transformation as a building block. Their proven security is somewhat smaller than the maximum differential probability of the underlying non-linear permutation, and it does not change with the message length as in the polynomial constructions or PCH. Hence, if one uses four rounds of AES with independent keys as a building block one can obtain a message authentication scheme that is more time efficient and offers significantly greater security compared to the message authentication modes with AES. If the target security is that of the message authentication modes with AES, then one can use non-linear permutations defined on 64-bit strings (blocks). This allows for greater speedup and better key agility. For instance, the non-linear transformations that we suggest use 128- and 192-bit keys as opposed to the 512-bit key required by four rounds of AES. If these components are used in a Wegman-Carter single-binary-tree hash, then the achievable speedup for lengthy messages approaches 4.5 on 8-bit architectures, 3.3 on 32-bit architectures and 5 on 64-bit architectures with relatively large L1 cache as opposed to the 2.5 speedup achievable when the non-linear permutation is four rounds of AES. In order to improve the memory efficiency, MACH, the message authentication scheme we propose, uses the modified Wegman-Carter tree AXU_2 hash function (the MACH hash) instead of a single tree. The estimated speedup of the resulting scheme is somewhat smaller, but still significant (see Section 4.3 for more details).

2 Basic building blocks

In this section, we propose some basic AXU_2 and AU_2 hash functions. We use these functions as building blocks to construct efficient message authentication schemes.

2.1 AXU_2 hash functions based on block cipher design techniques

Given a (keyed) non-linear function F, one can construct an AXU_2 hash function as follows. To hash a message x, two keys K and K_r are chosen randomly. The hash of x is $F(K, x \oplus K_r)$. If F is not a keyed transformation, then the hash of x is $F(x \oplus K_r)$. The role of the key K_r is to randomize the input of F since the maximum differential probability is defined for a randomly selected input and a constant input difference. The AXU_2 definition on the other hand requires both the input and the input difference to be constant. A more formal analysis is given below.

Lemma 1. *Let $F : \{0,1\}^k \times \{0,1\}^m \to \{0,1\}^n$ be a mapping that maps a pair of a k-bit key and a message (block) of length m into an n-bit string. The family of hash functions $\mathcal{H} = \{h_{K,K_r} : \{0,1\}^m \to \{0,1\}^n | K \in \{0,1\}^k, K_r \in \{0,1\}^m, h_{K,K_r}(x) = F(K, x \oplus K_r)\}$ is ϵ-AXU_2, where ϵ is equal to the* maximum (expected) differential probability *of F*

$$\mathrm{DP}_F = \max_{\Delta x \neq 0, \Delta y} \frac{\#\{(K, x) \in \{0,1\}^k \times \{0,1\}^m | F(K, x \oplus \Delta x) \oplus F(K, x) = \Delta y\}}{2^{m+k}}.$$

The non-linear function defined by four rounds of AES is a good candidate for constructing AXU hash functions. To hash a 128-bit block x, one selects four uniformly random keys and "encrypts" x using the four keys as round keys. Here, we assume that the key addition is at the beginning of the rounds, not at the end of the rounds. We also assume that the fourth round is a final AES round. It was shown in [33] that the maximum differential probability of four rounds of AES is at most about 2^{-113} when the round keys are independent. Hence, the hash function family $\mathcal{H}_{AES}$ consisting of the transformations defined by four rounds of AES for all possible values of the round keys is ϵ-AXU$_2$, where $\epsilon \approx 2^{-113}$. We propose two additional constructions.

The first AXU_2 family of hash functions that we suggest is defined by the Feistel structure depicted in Fig. 1. The 64-bit input is transformed into a 64-bit hash using three Feistel rounds. Each round uses a new 64-bit key. The round function is depicted in Fig. 1(b). It is constructed using AES components. That is, the S-box and the mixing transformation used in the round function are same as those used in AES. Each key defines a hash function that maps a 64-bit string (message) into a 64-bit hash, and we denote by $\mathcal{H}_{FES}$ the family of hash functions defined by the 2^{192} possible keys.

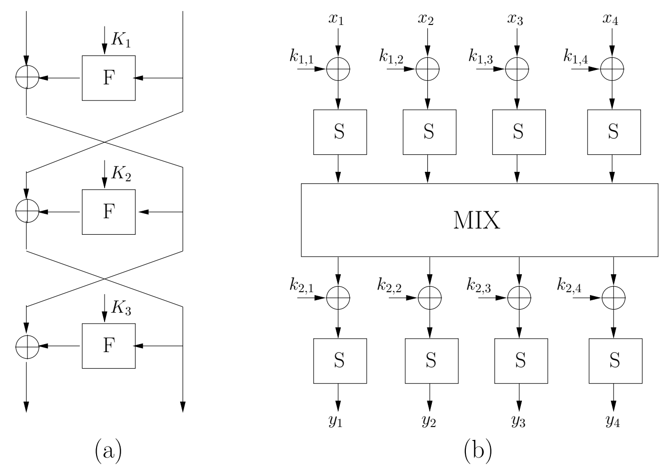
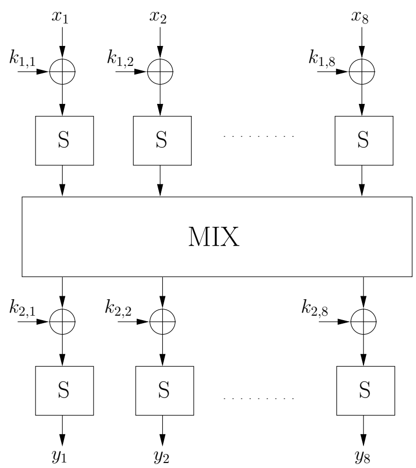


Fig. 1. A Feistel AXU construction: (a) the 64-bit message is hashed using three Feistel rounds with independent keys, (b) The round function is an SPN structure. The S-box and the mixing transformation are those used in AES.

The security of $\mathcal{H}_{FES}$ is provided by the following lemma.

Lemma 2. *The $\mathcal{H}_{FES}$ family of hash functions is ϵ-AXU_2, where $\epsilon = 1.52 \times 2^{-56}$.*

The second AXU_2 family of hash functions that we suggest is defined by the keyed nonlinear transformation shown in Fig. 2. It is a two-round SPN structure that transforms a 64-bit input into a 64-bit output. The S-box that is used in the construction is same as the one used in AES. The mixing transformation is given by the circulating-like MDS matrix proposed in [28] (p. 167). The multiplication and addition are over GF(256) modulo the irreducible polynomial $x^8+x^4+x^3+x^2+1$ over GF(2). The coefficients are given by the following polynomials over GF(2): $a = x+1$, $b = x^3+1$, $c = x^3+x^2$, $d = x$, $e = x^2$ and $f = x^4$. Each key defines a hash function that maps a 64-bit message into a 64-bit hash, and we denote by $\mathcal{H}_{F64}$ the family of 2^{128} hash functions whose members are determined by the possible key values.



$$M_8 = \begin{bmatrix} f & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & a & b & c & d & e & f \\ 1 & f & 1 & a & b & c & d & e \\ 1 & e & f & 1 & a & b & c & d \\ 1 & d & e & f & 1 & a & b & c \\ 1 & c & d & e & f & 1 & a & b \\ 1 & b & c & d & e & f & 1 & a \\ 1 & a & b & c & d & e & f & 1 \end{bmatrix}$$

Fig. 2. An SPN AXU construction: (a) The global structure, (b) The 8×8 matrix used in the linear mixing layer. The multiplication and addition are over GF(256) modulo $x^8+x^4+x^3+x^2+1$ over GF(2). The coefficients are $a = x+1$, $b = x^3+1$, $c = x^3+x^2$, $d = x$, $e = x^2$ and $f = x^4$.

The following lemma establishes the security of $\mathcal{H}_{F64}$.

Lemma 3. *The $\mathcal{H}_{F64}$ family of hash functions is ϵ-AXU_2, where $\epsilon = 1.25 \times 2^{-54}$.*

2.2 The AU_2 hash functions

Given a keyed non-linear function that can be represented as a composition of two non-linear transformations whose keys[1] are independent (see Fig. 3(a)), one can construct an AU hash function (see Fig. 3(b)) as follows.

[1] We consider a more general case. However, F_1 and F_2 does not have to be keyed transformations (i.e., the lengths of the keys K_1 and K_2 can be zero as well).

Lemma 4 (Twisting Lemma). *Let* $F(K,x)$ *be defined as* $F(K,x) = F_2(K_2, F_1(K_1,x) \oplus K_s)$*, where* $K = K_1|K_s|K_2$*,* $F_1 : \{0,1\}^{k_1} \times \{0,1\}^l \to \{0,1\}^n$*, and* $F_2 : \{\{0,1\}^{k_2} \times \{0,1\}^n \to \{0,1\}^n\}$ *is a bijection for any key value* K_2*. Then, the family of hash functions* $\mathcal{H} = \{h_{K_1,K_{r1},K_2,K_{r2}} : \{0,1\}^l \times \{0,1\}^n \to \{0,1\}^n | h_{K_1,K_{r1},K_2,K_{r2}}(x_1,x_2) = F_1(K_1, x_1 \oplus K_{r1}) \oplus F_2^{-1}(K_2, x_2 \oplus K_{r2})\}$ *is an* ϵ*-A*U_2*, where* $K_1 \in \{0,1\}^{k_1}; K_2 \in \{0,1\}^{k_2}; x_1, K_{r1} \in \{0,1\}^l; x_2, K_{r2} \in \{0,1\}^n$*, and* $\epsilon = \mathrm{DP}_F$*.*

The structure of the function F depicted in Fig. 3(a) can be found in almost any block cipher and allows for a variety of AU_2 hash function constructions by "twisting" block ciphers. One such example is the construction proposed in [40], which is depicted in Fig. 3(c). The function F in this case is a composition of an identity map and the inverse of a differentially uniform permutation. The twisting lemma is slightly abused since no key is added to the first block. Such key addition will be canceled when we consider differences and increases the time complexity since one has to generate a random key K_{r1}.

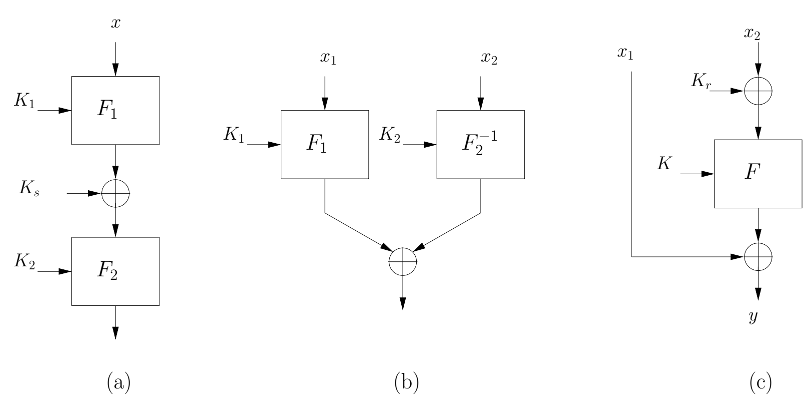


Fig. 3. AU$_2$ construction by "twisting" block ciphers: (a) the original non-linear transformation F, (b) the non-linear transformation F' obtained by "twisting" F, (c) AU$_2$ construction proposed in [40].

The general construction of Lemma 4 offers a somewhat greater level of parallelism than the one of Fig. 3(c) (one can evaluate F_1 and F_2 in parallel). However, the overall impact on the schemes proposed in this paper is not significant, and we use a variant of Fig. 3(c) which is derived by extending its domain to include messages of length 0 and 1 blocks:

$$g_F(x_1,x_2) = \begin{cases} \lambda & if \;\; x_1 = x_2 = \lambda \\ x_1 & if \;\; x_1 \neq \lambda, x_2 = \lambda \\ x_1 \oplus F(K, x_2 \oplus K_r) & if \;\; x_1 \neq \lambda, x_2 \neq \lambda \end{cases}$$

where λ is the empty string, $x_1, x_2 \in \{0,1\}^n \bigcup \{\lambda\}$, $K_r \in \{0,1\}^n$ and F is a (keyed) non-linear permutation on $\{0,1\}^n$.

Let $\mathcal{G}_F$ be the family of the hash functions defined as above. We have the following lemma.

Lemma 5. *The family of hash functions $\mathcal{G}_F$ is ϵ-AU_2, where $\epsilon = \mathrm{DP}_F$.*

The AU_2 families that we use are obtained when the AXU_2 hash function $F(K, K_r \oplus x)$ is realized using the transformations discussed in Section 2.1. We denote by $\mathcal{G}_{AES}$, $\mathcal{G}_{FES}$ and $\mathcal{G}_{F64}$ the families of hash functions when $F(K, K_r \oplus x)$ is realized using four AES rounds, the Feistel structure of Fig. 1 and the SPN structure of Fig. 2 respectively. According to the previous discussion, $\mathcal{G}_{AES}$, $\mathcal{G}_{FES}$ and $\mathcal{G}_{F64}$ are ϵ-AU_2 with ϵ being 2^{-113}, 1.52×2^{-56} and 1.25×2^{-54} correspondingly.

3 AXU_2 hash functions defined for arbitrary-length messages

The universal hash functions introduced in the previous section operate on message blocks. In this section, we consider some techniques for extending the domains to include arbitrary-length messages. The proposed constructions use a large number of keys. However, these keys are derived from a single 128-bit key in the message authentication scheme we propose in Section 4.

3.1 A CBC-like construction

CBC is a popular approach to MAC design. The Pelican MAC of [18] and the PCH (Periodic CBC Hash) of [40] resemble CBC as well. Here, we present another CBC-like family of hash functions $\mathcal{H}_{UHC}$ (Universal Hash Chaining). The advantage of UHC over the Pelican construction is that it is proven secure. Its advantage over PCH is that the security does not decrease with the message length. Assuming small differential probabilities, the provided upper bound on the collision probability of PCH is roughly $l^2/2^n$, where l is the message length and n is the block length. If the message length is about 2^{40}, this results in about 2^{-50} proven security when using four rounds of AES as a building block. The proven security of UHC in this case will be about 2^{-112}.

$\mathcal{H}_{UHC}$ is depicted in Fig. 4. We assume that F is a permutation on the set of n-bit strings for a given key. To hash a message consisting of l segments of m blocks, we select randomly $m-2$ randomization keys $K_3^r, \ldots, K_m^r$ and $m-1$ keys $K_2, K_3, \ldots, K_m$ for the non-linear map F. These keys are used for all segments of the message. In addition, two fresh randomization keys $K_{i,1}^r, K_{i,2}^r$ and a fresh key $K_{i,1}$ for the non-linear map are selected anew for each segment of the message. The message is "digested" in a CBC-like manner using these keys as depicted in Fig. 4. The resulting family of hash functions is AXU_2.

Lemma 6. *$\mathcal{H}_{UHC}$ is ϵ-AXU_2, where $\epsilon = 2\mathrm{DP}_F$.*

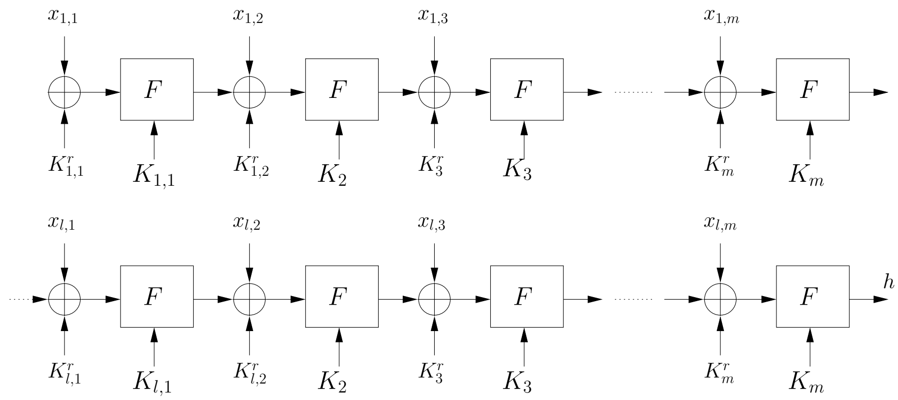


Fig. 4. A CBC-like AXU construction. Fresh keys are used only for the first two blocks of each segment.

3.2 A modified Wegman-Carter binary tree construction

MACH, the MAC scheme that we propose, uses the following variant of the Wegman-Carter binary tree hash.

To hash a message M consisting of l blocks[2], we first "append" λ-"blocks" [3] so that the number of blocks in the message is a multiple of 2^N. The resulting λ-padded message is partitioned into segments consisting of 2^N blocks. Each segment is hashed using the same secret member of $\mathcal{G}_F$ in a binary hash tree of height N. Recall that that the members of $\mathcal{G}_F$ were defined as $g_F(x_1, x_2) = x_1 \oplus F(K, K_r \oplus x_2)$ if x_2 is not λ, $g_F(x_1, \lambda) = x_1$, and $g_F(\lambda, \lambda) = \lambda$, where F is a (keyed) non-linear permutation on n-bit strings. The output of each binary tree is hashed using F as in Lemma 1, and the resulting n-bit blocks are xor-ed to give the final hash value. The keys used in the last step are generated independently for different segments of the message. We use $\mathcal{H}^F_{MACH}(N)$ to denote the family of hash functions described above. An example when $N = 2$ is given in Fig.5.

The time complexity of the MACH hash is determined by the time to generate the required keys, and the time to hash the message. Assuming that the keys are already generated, the time to hash the message is one F evaluation per n-bit block of the message, and it is same as that of UHC. The same levels of the binary trees in the MACH hash use the same key. So, one has to generate and memorize N keys that will be used by the binary trees. In addition, one has to generate one potentially large key per segment for the last step of the hashing procedure. Since the length of the segments is 2^N blocks, the MACH hash is advantageous over the UHC hash where one has to generate fresh keys every N blocks.

---

[2] We assume that the message length is a multiple of the block length.

[3] The sole purpose of the λ padding is to simplify our description and analysis. In practice, the λ padding will be omitted.

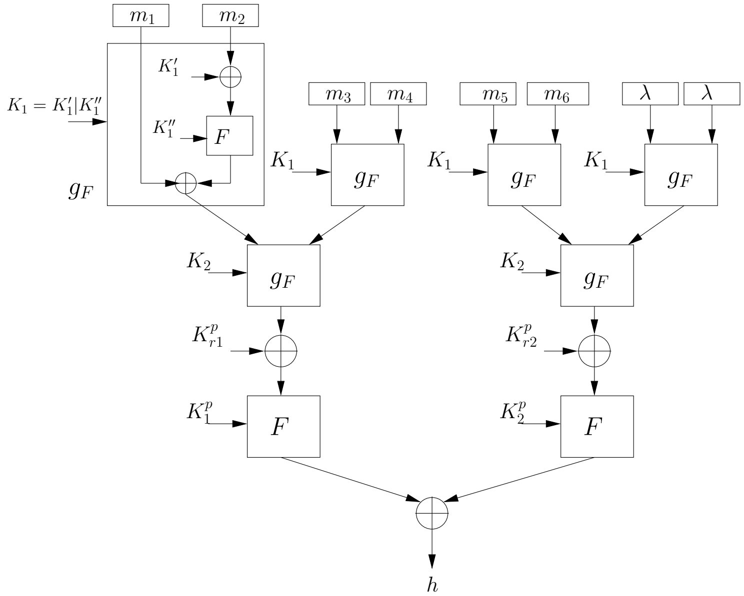


Fig. 5. An AXU_2 construction using a modification of the Wegman-Carter binary tree ($N = 2$).

Using a single binary tree will lead to greater speedup for long messages. However, one will have to memorize a large number of keys to allow hashing of lengthy messages. In the MACH hash, the fresh keys can be "thrown away" after their use. So, by carefully selecting the value of N, one can achieve close to a single-binary-tree speed while significantly improving the key agility and memory efficiency compared to the single-binary-tree construction.

$\mathcal{H}^F_{MACH}(N)$ is basically a composition of an AU_2 hash function (the binary trees in parallel) and an AXU_2 hash function (the xor of the AXU_2 hash functions). As it was case with the $\mathcal{H}_{UHC}$, the security of $\mathcal{H}^F_{MACH}(N)$ does not decrease with the message length.

Lemma 7. *$\mathcal{H}^F_{MACH}(N)$ is an ϵ-$\mathrm{A}XU_2$ family of hash functions, where $\epsilon = (N+1) \times \mathrm{DP}_F$.*

The message authentication schemes that we propose in this paper use the $\mathcal{H}^{AES}_{MACH}(5)$, $\mathcal{H}^{FES}_{MACH}(7)$ and $\mathcal{H}^{F64}_{MACH}(7)$ hash function families. Here, $\mathcal{H}^{AES}_{MACH}(5)$ is the MACH hash functions where the binary trees are of height 5, the AU hash function family used in the binary trees is $\mathcal{G}_{AES}$ of Section 2.2, and the AXU hash function family used in the last step is the $\mathcal{H}_{AES}$ hash function family described in Section 2.1. Similarly, $\mathcal{H}^{FES}_{MACH}(7)$ (resp., $\mathcal{H}^{F64}_{MACH}(7)$) is the MACH hash function family that uses binary trees of height 7, and whose non-linear function F is implemented using the Feistel (resp., SPN) structure of Fig. 1 (resp., Fig. 2).

4 MACH: An efficient Wegman-Carter MAC scheme based on block cipher design techniques

In this section, we present MACH. MACH, where H stands for the use of hash functions, is a Wegman-Carter MAC scheme that is obtained by applying the technique presented in [46] to $\mathcal{H}^F_{MACH}$.

4.1 The signing (tagging) and verifying algorithms of MACH

Signing A pseudo-code of the MACH signing algorithm is given in Algorithm 1. It takes as input a secret key K, a 64-bit counter value $Cntr < \mathsf{MAX\_CNTR}$ associated with that key and a message M of bit length $|M| < \mathsf{MAX\_LEN}$. The secret key K and the counter value $Cntr$ are used as an input to a pseudorandom generator that outputs two keys K_h and K_T. The key K_h specifies which member $\mathcal{H}^F_{MACH}$ will be used to hash the messages, and the key K_T is used to encrypt the hash of the message. Given the key K_h, a hash $h = h_{K_h}(M|10^i)$ of the 10^i padded message is computed using the $\mathcal{H}^F_{MACH}$ family of hash functions. The authentication tag τ is the pair consisting of the counter value $Cntr$ and $h_\tau = h \oplus K_T$.

Algorithm 1 MACH.Sign(K,$Cntr$,M)

Input: A (128-bit) secret key K, a 64-bit counter value $Cntr$ and a message M.
Output: An authentication tag τ.

$Cntr++$
$\mathsf{len} \leftarrow |M|$ // len is the bit length of the message M.
$K_h, K_T \leftarrow \mathsf{Gen}(Cntr, K)$

$i \leftarrow (n - ((\mathsf{len} + 1) \bmod n)) \bmod n$
$h \leftarrow h_{K_h}(M|10^i)$
$h_\tau \leftarrow h \oplus K_T$

return $\tau \equiv (Cntr, h_\tau)$

The keys K_h and K_T can be generated using a pseudorandom generator (i.e., a stream cipher). The key generation in this case will be faster than using a block cipher, and the resulting scheme will be more competitive for small message lengths. However, there are some practical advantages of generating the keys using a block cipher in a counter-like mode. So, we suggest the keys to be generated using AES as follows. The key K_T is computed as $K_T = \mathsf{trun}(AES_K(1|0^{63}|Cntr))$, where $Cntr$ is a 64-bit counter value, and $\mathsf{trun}(\cdot)$ selects the first $|h|$ bits of $AES_K(1|0^{63}|Cntr)$. The words of the key K_h are computed as $K_h[i] = \mathrm{AES}_K(0^{64}|\langle i \rangle)$, where $\langle i \rangle$ is a 64-bit representation of i. If the length of K_h is not a multiple of 128, then the last "word" $K_h[\mathsf{K\_BLCKS}]$ of K_h is derived by selecting the first $|K_h[\mathsf{K\_BLCKS}]|$ bits of $\mathrm{AES}_K(0^{64}|\langle \mathsf{K\_BLCKS} \rangle)$.

Here, K_BLCKS is the number of blocks in K_h, and it is determined by the length of the key material we need to hash a message of length $\mathsf{MAX\_LEN}$.
Remark. To simplify our description and security analysis, we have assumed that the key K_h is generated at the beginning, and that it is long enough to hash messages of maximum length. Clearly, such implementation is not practical at all. In practice, to avoid expensive key setup and increase the memory efficiency of the scheme, the keys will be generated on the fly, and only a small portion of the keys (e.g., the keys used by the binary trees) will be memorized when computing the hash of the message.

Verifying Given a message M, an authentication tag $\tau = (Cntr, h_\tau)$ associated with the message and the secret key K, the verifier computes the keys K_h, K_T, and recomputes the authentication tag using these keys. If the recomputed tag $(Cntr, K_T \oplus h_{K_h}(M|10^i))$ is equal to the one that was sent, then the verifier accepts the message M as authentic. Otherwise, the verifier rejects the message M.

4.2 Security of MACH

The security of MACH is established by the following theorem.

Theorem 1. *The advantage of any forger of MACH that runs in at most t time and makes at most q_v forgery attempts is upper bounded by*

$$\mathbf{Adv}^{\mathrm{wuf-cma}}_{MACH}(t, q_v) \leq \mathbf{Adv}^{\mathrm{prp}}_{\mathsf{AES}}(c_1 t + c_2, Q_e) + q_v(1 - \frac{Q_e - 1}{2^{128}})^{-Q_e/2}(N+1)\mathrm{DP}_F,$$

where c_1 and c_2 are small implementation dependent constant, $Q_e = \mathsf{K\_BLCKS} + \mathsf{MAX\_CNTR}$, N is the height of the binary trees used by the hash function, DP_F is the maximum differential probability of the nonlinear permutation F used by the hash function, and $\mathbf{Adv}^{\mathrm{prp}}_{\mathsf{AES}}(c_1 t + c_2, Q_e)$ is the advantage of distinguishing AES from a random permutation when running in at most $c_1 t + c_2$ time and querying an encryption oracle at Q_e distinct message blocks.

4.3 MACH variants, security and performance comparison

We suggest three MACH variants MACH-AES, MACH-FES and MACH-F64. As their names suggest, the proposed MACH variants are obtained when the messages are hashed using $\mathcal{H}^{AES}_{MACH}(5)$, $\mathcal{H}^{FES}_{MACH}(7)$ and $\mathcal{H}^{F64}_{MACH}(7)$ respectively (see Section 3.2 for a description of these hash functions). In the following, we briefly discuss the security and performance of these schemes.

Security A comparison of the proposed variants in terms of their security and the speedup over the modes for message authentication that use AES as a building block is given in Table 1. The security expressions are derived using Theorem 1. We assume that both $\mathsf{K\_BLCKS}$ and $\mathsf{MAX\_CNTR}$ are 2^{64}. The number of encryption queries in this case will be $Q_e = 2^{65}$, and δ is the advantage of distinguishing AES from a random permutation given Q_e pairs of plaintext/ciphertext blocks.

Table 1. Security and performance comparison of the MACH variants

Scheme	Speedup over message authentication modes						Security
	8-bit c.a.		32-bit c.a.		64-bit c.a.		
	1 KB	∞	1 KB	∞	1 KB	∞	
MACH-AES	1.25	2.10	1.19	1.90	1.19	1.90	$\delta + q_v \times 1.28 \times 2^{-110}$
MACH-FES	2.37	3.76	1.99	2.88	1.99	2.88	$\delta + q_v \times 1.30 \times 2^{-52}$
MACH-F64	1.85	2.10	1.51	1.86	2.10 – 2.94	2.81 – 4.64	$\delta + q_v \times 1.07 \times 2^{-50}$

The tag length and the security of MACH-FES and MACH-F64 are comparable to those of the modes of operation for message authentication using AES as a building block. Note that the security of MACH-FES and MACH-F64 is determined by the number of forgery attempts. If the application allows the verifier to limit the number of forgery attempts, then one can achieve good security for a large number of very long messages. For example, assume that the verifier keeps a track of the number of invalid message/tag pairs. If this number exceeds 2^{20}, then the verifier assumes it is under attack and rejects any subsequent message. Under these circumstances, we can use MACH-FES and MACH-F64 to authenticate 2^{64} messages of length $\gtrsim 2^{64}$ blocks with $\approx 2^{-30}$ forgery probability. However, using 2^{64} signing queries and a single forgery attempt, one can easily break most of the existing modes of operation for message authentication. If the maximum allowed message length is relatively large, then the security of MACH-FES and MACH-F64 is comparable or better than that of the polynomial-based constructions too. For example, if one allows messages of length $> 2^{52}$ blocks, then the proven security of Poly1305-AES becomes smaller than that of MACH-FES and MACH-F64.

Assuming that the advantage of distinguishing AES from a random permutation given 2^{65} plaintext/ciphertext pairs is small, MACH-AES provides significantly better security than MACH-FES and MACH-F64. The tag length (including the counter) of MACH-AES is 192 bits, and it is larger than that of the modes of operations for message authentication.

Performance Performance evaluation of a given message authentication scheme is not an easy task since it depends on the specific platform, the implementation of the algorithms and the message length distribution. The speedup estimates given in Table 1 are computed by making the following assumption: the algorithms are implemented using basic arithmetic and memory reference instructions available on RISC computer architectures. The speedup is computed by dividing the time needed to compute the tag using AES in a message authentication mode and the time needed to compute the tag using the proposed schemes.

The execution time on the other hand is estimated based on the number of arithmetic and memory reference instructions required to compute the tag.

We have considered two cases. In the first case, which is denoted 1 KB, the message length is 1024 bytes as in [7]. The speedup in this case approximates the speedup when the message length distribution follows the IP packet size distribution on the Internet. The time to compute the tag in this case includes the time needed to generate all the keys that are required to hash the message. In the second case, which is denoted ∞, we assume that the keys used by the binary trees of the hash function are already generated and memorized. The time to compute the tag includes the time needed to generate the fresh keys used for the different segments of the message, but it does not include the time to generate the keys used by the binary trees in the MACH hash. The speedup in the second case approximates the speedup when authenticating a single long message or authenticating a relatively long sequence of short messages (e.g., stream authentication, authenticating the packets exchanged between two routers in a VPN, authenticating the packets exchanged during a single communication session, etc.).

MACH-AES and MACH-FES use AES components as building blocks. Hence, the estimation of their speedups is easier. The key generation cost is one AES encryption per 128 bits of the key material. Given the keys, the cost of hashing per 128-bit block is about 4 AES rounds for MACH-AES and 3 AES rounds for MACH-FES on 32-bit and 64-bit architectures. The AES matrix multiplication is relatively costly on 8-bit architectures (about 40 arithmetic operations). The mixing transformation is omitted in the fourth round of the non-linear function used by MACH-AES. Thus, on 8-bit architectures, the cost of hashing is about 3.5 AES rounds per 128-bit block when using MACH-AES. For similar reasons, the cost of hashing is about 2.2 AES rounds per 128-bit block on 8-bit architectures when using MACH-FES.

MACH-F64 uses an 8×8 multiplication matrix which is not a component of AES. Hence, the computation of the speedup is more complicated. A detailed discussion on implementing this matrix multiplication on various platforms can be found in [28]. We will only note that the largest speedup values on 64-bit architectures are computed assuming that the non-linear transformation of MACH-F64 is implemented using 8 look-up tables each one containing 256 64-bit entries. The memory required to store these tables is 16 KB, which is a relatively small portion of the L1 cache of many processors. For example, AMD Athlon, UltraSparc III and Alpha 21264 have 64 KB L1 cache, PowerPC G4 and G5 have 32 KB L1 cache, etc.

Summary MACH-AES is less time and memory efficient than MACH-FES and MACH-F64. However, it provides much better security, and the achievable speedup over the message authentication modes is significant in some settings. MACH-F64 and MACH-FES provide security and tag lengths that are comparable to those of the message authentication modes. The target computer architecture of the MACH-F64 design was a 64-bit architecture with large L1 cache, and it is extremely efficient on these architectures. MACH-FES on the other hand is very efficient on 8-bit architectures, and achieves a significant speedup on 32- and 64-bit architectures as well. Both MACH-AES and MACH-FES are built using AES components. So, they have the advantage of reusing AES software and hardware.

References

1. V. Afanassiev, C. Gehrmann and B. Smeets, “Fast message authentication using efficient polynomial evaluation,” In Proceedings of FSE 1997, pp. 190–204.
2. M. Bellare, J. Kilian and P. Rogaway,“The security of the cipher block chaining message authentication code,” JCSS, vol. 61, no. 3, 2000. Earlier version in Advances in Cryptology - CRYPTO ’94, LNCS 839, pp. 341–358, Springer-Verlag, 1994.
3. M. Bellare, R. Canetti and H. Krawczyk,“Keying hash functions for message authentication,” In Advances in Cryptology - CRYPTO ’96, LNCS 1109, p. 1, Springer-Verlag, 1996.
4. M. Bellare and C. Namprempre,“Authenticated encryption: relations among notions and analysis of the generic composition paradigm,” In Advances in Cryptology - ASIACRYPT 2000, LNCS 1976, Springer-Verlag, 2000.
5. M. Bellare,“New Proofs for NMAC and HMAC: Security without Collision-Resistance,” In Advances in Cryptology - CRYPTO 2006, LNCS 4117, Springer-Verlag, 2006.
6. A. Berendschot, B. den Boer, J. P. Boly, A. Bosselaers, J. Brandt, D. Chaum, I. Damgard, M. Dichtl, W. Fumy, M. van der Ham, C. J. A. Jansen, P. Landrock, B. Preneel, G. Roelofsen, P. de Rooij and J. Vandewalle, “Final Report of RACE Integrity Primitives,” LNCS 1007, Springer-Verlag, 1995.
7. D. Bernstein, “Stronger security bounds for Wegman-Carter-Shoup authenticators,” In Advances in Cryptology - EUROCRYPT 2005, LNCS, pp. 164–180, Springer-Verlag, 2005.
8. D. Bernstein,“The Poly1305-AES message authentication code,” In the Proceedings of FSE 2005, LNCS 3557, pp. 32–49, Springer-Verlag, 2005.
9. J. Bierbrauer, T. Johansson, G. Kabatianskii and B. Smeets, “On families of hash functions via geometric codes and concatenation,” In the Proceedings of CRYPTO ’93, LNCS, pp. 331–342, Springer-Verlag, 1993.
10. E.Biham and A.Shamir, ”Differential cryptanalysis of DES-like cryptosystems,” Journal of Cryptology, 4(1):3–72, 1991.
11. E. Biham, O. Dunkelman and N. Keller,“Related-key impossible differential attacks on 8-round AES-192,” CT-RSA 2006, LNCS 3860, pp. 21–33, Springer-Verlag, 2006.
12. J. Black, S. Halevi, H. Krawczyk, T. Krovetz and P. Rogaway, “UMAC: Fast and secure message authentication,” In Advances in Cryptology - CRYPTO ’99, LNCS, pp. 216–233, Springer-Verlag, 1999.
13. J. Black and P. Rogaway,“CBC MACs for arbitrary-length messages: The three key constructions,” In Advances in Cryptology - CRYPTO 2000, LNCS 1880, pp. 197–215, Springer-Verlag, 2000.
14. J. Black and P. Rogaway,“A block-cipher mode of operation for parallelizable message authentication,” In Advances in Cryptology - EUROCRYPT 2002, LNCS 2332, pp. 384–397, Springer-Verlag, 2002.

15. G. Brassard,"On computationally secure authentication tags requiring short secret shared keys," In Advances in Cryptology - CRYPTO'82, LNCS, pp. 79–86, Springer-Verlag, 1982.
16. L. Carter and M. Wegman,"Universal classes of hash functions," JCSS, 22:265–279, 1981.
17. J. Daemen and V. Rijmen, "A New MAC Construction ALRED and a Specific Instance ALPHA-MAC," In the Proceedings of FSE 2005, LNCS 3557, pp. 1–17, 2005.
18. J. Daemen and V. Rijmen, "The Pelican MAC Function," IACR ePrint Archive, 2005/088.
19. M. Etzel, S. Patel and Z. Ramzan,"Square Hash: Fast message authentication via optimized universal hash functions," In Advances in Cryptology - CRYPTO '99, LNCS 1666, Springer Verlag, 1999.
20. FIPS 113. Computer data authentication. Federal Information Processing Standards Publication 113, U. S. Department of Commerce / National Bureau of Standards, National Technical Information Service, Springfield, Virginia, 1994.
21. Advanced Encryption Standard (AES), FIPS Publication 197, November 26, 2001, available at http://csrc.nist.gov/encryption/aes.
22. E. Gilbert, F.M. Williams and N. Sloane,"Codes which detect deception," Bell System Technical Journal, 53(3):405–424, 1974.
23. S. Halevi and H. Krawczyk,"MMH: Message authentication in software in the gbit/second rates," In the Proceedings of the 4th FSE Workshop, LNCS, Springer-Verlag, 1997.
24. S. Hong, S. Lee, J. Lim, J. Sung, D. Cheon and I. Cho, "Provable security against differential and linear cryptanalysis for the SPN structure," In Proceeding of the 7th FSE Workshop, LNCS 1978, pp. 273-283, Springer, 2000.
25. ISO/IEC 9797-1. Information technology - security techniques - data integrity mechanism using a cryptographic check function employing a block cipher algorithm. International Organization for Standards, Geneva, Switzerland, 1999. Second edition.
26. T. Iwata and K. Kurosawa,"OMAC: One-Key CBC MAC," In Proceedings of FSE 2003, LNCS 2887, pp. 129–153. Springer-Verlag, 2003.
27. É. Jaulmes, A. Joux and F. Valette,"On the security of randomized CBC-MAC beyond the birthday paradox limit: A new construction," In the Proceedings of FSE 2002, LNCS 2365, pp. 237–251, Springer-Verlag, 2002.
28. P. Junod, "Statistical Cryptanalysis of Block Ciphers," PhD Thesis, EPF, Switzerland.
29. B. Kaliski and M. Robshaw,"Message authentication with MD5," Technical newsletter of RSA Laboratories, 1995.
30. J.-S. Kang, S. Hong, S. Lee, O. Yi, C. Park and J. Lim,"Practical and provable security against differential and linear cryptanalysis for substitution-permutation networks," ETRI Journal, 23(4):158-167, 2001.
31. L. Keliher, H. Meijer and S.Tavares, "New method for upper bounding the maximum average linear hull probability for SPNs," In Advances in Cryptology - EUROCRYPT 2001, LNCS 2045, pp. 420-436, Springer-Verlag, Berlin, 2001.
32. L. Keliher, H. Meijer and S. Tavares, "Improving the upper bound on the maximum average linear hull probability for Rijndael," In the Proceedings of Selected Areas in Cryptography, 8th Annual International Workshop, LNCS 2259, pp. 112-128, Springer, 2001.

33. L. Keliher and J. Sui, Exact Maximum Expected Differential and Linear Probability for 2-Round Advanced Encryption Standard (AES), IACR ePrint Archive, 2005/321.
34. J. Kelsey, B. Schneier and D. Wagner,"Related-key cryptanalysis of 3-WAY, Biham-DES, CAST, DES-X, NewDES, RC2 and TEA," In the Proceedings of ICICS '97, LNCS 1334, pp. 233–246, Springer-Verlag, 1997.
35. H. Krawczyk,"LFSR-Based Hashing and Authentication," In Advances in Cryptology - CRYPTO '94, LNCS 839, pp. 129–139, Springer-Verlag, 1994.
36. T. Krovetz and P. Rogaway,"Fast universal hashing with small keys and no preprocessing: The PolyR construction," In the Proceedings of ICICS 2000, pp. 73–89, Springer-Verlag, 2000.
37. K. Kurosawa and T. Iwata,"TMAC: Two-Key CBC MAC," Cryptology ePrint Archive, Report 2002/092, http://eprint.iacr.org/.
38. M. Matsui, "New Structure of Block Ciphers with Provable Security against Differential and Linear Cryptanalysis," In the Proceedings of FSE 1996, LNCS 1039, Springer Verlag, 1996.
39. M. Matsui, "New block encryption algorithm MISTY," In the Proceedings of FSE 1997, LNCS 1267, pp. 53-67, Springer-Verlag, 1997.
40. K. Minematsu and Y. Tsunoo, "Provably Secure MACs from Differentially-Uniform Permutations and AES-Based Implementations," In the Proceedings of FSE 2006, LNCS 4047, pp. 226–241, 2006.
41. S. Park, S. H. Sung, S. Chee, E.-J. Yoon and J. Lim, "On the security of Rijndael-like structures against differential and linear cryptanalysis," In Advances in Cryptology - ASIACRYPT 2002, LNCS 2501, pp. 176-191, Springer, 2002.
42. S. Park, S.H. Sung, S. Lee and J. Lim, "Improving the upper bound on the maximum differential and the maximum linear hull probability for SPN structures and AES," In the Proceedings of FSE 2003, LNCS 2887, pp. 247–260, Springer, 2003.
43. E. Petrank and C. Rackoff,"CBC MAC for real-time data sources," J.Cryptology, vol. 13, no. 3, pp. 315–338, Springer-Verlag, 2000.
44. B. Preneel and P. van Oorschot,"MDx-MAC and building fast MACs from hash functions," In Advances in Cryptology - CRYPTO '95, LNCS 963, pp. 1–14, Springer-Verlag, 1995.
45. B. Preneel and P. van Oorschot,"On the security of two MAC algorithms," In Advances in Cryptology - EUROCRYPT '96, LNCS 1070, pp. 19–32, Springer-Verlag, 1996.
46. P. Rogaway,"Bucket hashing and its application to fast message authentication," In Advances in Cryptology - CRYPTO '95, LNCS 963, pp. 29–42, Springer-Verlag, 1995.
47. G.J. Simmons,"Authentication theory / Coding theory," In Advances of Cryptology - CRYPTO '84, LNCS, pp. 411–432, Springer-Verlag, 1984.
48. D. Stinson,"Universal hashing and authentication codes," Designs, Codes and Cryptography 1994; 4:369–380.
49. G. Tsudik,"Message authentication with one-way hash functions," In the Proceedings of Infocom '92, IEEE Press, 1992.
50. S. Vaudenay,"Decorrelation over infinite domains: The encrypted CBC-MAC case," Communications in Information and Systems (CIS), vol. 1, pp. 75–85.
51. M. Wegman and L. Carter, "New hash functions and their use in authentication and set equality," JCSS, 22:265–279, 1981.