FRMAC, a Fast Randomized Message Authentication Code

Éliane Jaulmes[1] and Reynald Lercier[2]

[1] DCSSI Crypto Lab, 51 Bd de Latour Maubourg, F-75700 Paris 07 SP, France
eliane.jaulmes@wanadoo.fr,
[2] CELAR, Route de Laillé, F-35570 Bruz, France
reynald.lercier@m4x.org

Abstract. We revisit the randomized approach followed in the design of the RMAC message authentication code in order to construct a MAC with similar properties, but based on Wegman-Carter's ε-universal hash families instead of a classical CBC chain. This yields a new message authentication code called FRMAC whose security bounds are, as in RMAC, beyond the birthday paradox limit. With efficient hash functions in software, the performance of FRMAC for large messages is similar to those of the fastest previously known schemes. FRMAC can also be more efficient for small messages. Furthermore, due to relaxed requirements about the nonces in the security proof, the implementation of FRMAC in real applications tends to be easier.

Keywords: Block Ciphers, Message Authentication Codes.

1 Introduction

When shared symmetric keys are available, MACs (message authentication codes) are methods of choice for protecting the integrity of information. In this field of research, Wegman-Carter's approach yields efficient provably secure constructions [7, 8]. It consists in using, first, a keyed efficient hash function h_k, member of a so-called ε-almost delta universal family, in order to reduce the message $\mathcal{M}$ to a fixed size, and then, in adding modulo two the hash result and the image of a nonce R by a keyed pseudo-random function f_K. That is, the resulting MAC can be expressed as $f_K(R) \oplus h_k(\mathcal{M})$.
The most efficient of these MACs currently known on software targets for large messages is the UMAC-2000 scheme due to Black, Halevi, Krawczyk, Krovetz, and Rogaway (see [21] for a performance comparison of older constructions in software or [12] for a recent proposal). This MAC is an evolution of an older design based on CBC-MAC or HMAC [6] and was submitted by its authors to the NESSIE european project [23]. The UMAC-2000 hash families, UHash 16 or UHash 32, are especially well conceived. They are incredibly efficient on modern computers, easily parallelizable and their collision probabilities can be easily tuned to the user needs.
A nice feature of Wegman-Carter's MACs is that their security proofs are easily understandable. They mainly state that the forgery probability of an attacker can not exceed the sum of two terms. The first one is the advantage to distinguish a family F of functions with n-bit outputs from perfect pseudo-random functions. The second one is the number of forgery attempts, denoted q, times the collision probability ε. This yields a bound of the form $\mathbf{Adv}_F^{\mathbf{prf}}(\mathcal{A}) + q(\varepsilon + 1/2^n)$ [6]. The first term depends on F. For a secure cryptographic family of permutations of $\{0,1\}^n$, as for instance the AES block cipher [22], one can assume that $\mathbf{Adv}_F^{\mathbf{prf}}(\mathcal{A}) \simeq$

$q^2/2^{n+1}$ [2]. The second term is nearly the same as the advantage of an attacker whose strategy consists in checking whether a randomly generated tag is the correct one for a message $\mathcal{M}$. Despite the undeniable simplicity of the underlying theory, a few difficulties may arise when we try to implement such a MAC in practical applications. We list two of them below.

- It is crucial in the security proof that all nonces be distinct. If, for instance, a nonce R is used for the tag $T_{\mathcal{M}}$ of a message $\mathcal{M}$ and for the tag $T_{\mathcal{N}}$ of another message $\mathcal{N}$, then any other tag $T'_{\mathcal{M}}$ for $\mathcal{M}$ with nonce $R' \neq R$ can be forged into a tag for $\mathcal{N}$ with nonce R', that is $T'_{\mathcal{N}} = T'_{\mathcal{M}} \oplus T_{\mathcal{M}} \oplus T_{\mathcal{N}}$. In an application where several users share the same key, it can be a real difficulty to ensure the uniqueness of the nonces. Each user may for instance concatenate its identity and a counter but, first, an attacker can easily ascertain the number of messages which have been sent during a session, and second, the application has to manage very carefully a database of identities. It would be much more satisfactory, since in this case there is no need of a database, to use (pseudo-)random nonces. However, in order to avoid any repetition, the number of messages can not exceed $\sqrt{2^n}$ when R is a n-bit nonce.
- A second complication is that the families of functions that we have got in practice are most of the time n-bit permutations. We would expect that tags of n-bits yield a forgery probability equal to $q/2^n$ after q attempts. Unfortunately, due to the $\mathbf{Adv}_F^{\mathbf{prf}}(\mathcal{A})$ term, q can not exceed $\sqrt{2^n}$. In other words, with a $1/2^{128}$-almost delta universal family of hash functions with 128-bit outputs, we have to use the 256-bit block version of Rijndael and random nonces of size 256 bits in order to get a security bound which corresponds to the 128-bit size of the tags. It would have been more efficient to use the 128-bit block cipher AES for this level of security. Let us note that there exist solutions in the literature to build families of pseudo-random-function upon families of permutations [4, 1], but it is not clear to the authors how efficient it can be in practice.

We observe that the randomized approach followed in the RMAC construction [11, 10] seems to overcome these two difficulties if we assume an additional, but classical, security requirement on the block cipher (resistance to $\texttt{XOR}_n$ related key attacks). Especially, it yields slightly better security bounds, since they are beyond the birthday paradox limit, even with random nonces. Unfortunately, RMAC is not an efficient software MAC. So, FRMAC, the MAC that we study in this paper, combines the best part of both approaches. It consists, as in the Wegman-Carter approach, in using at first a hash function in order to reduce the message to a fixed size and then, as in RMAC, in applying a randomized permutation to the hash result. Although, for large messages, performance results may be the same as those of UMAC-2000, for small messages they can be slightly better, since a block cipher with a smaller block size can be used. Thus, for a given security bound, we are able to relax the requirements on nonces without any loss of performance.

The more closely related approach that the authors are aware of in this field is the construction $f_K(h_k(\mathcal{M}), R)$. When all nonces R are supposed to be distinct, a security proof can be, for instance, found in [6], where h_k is a hash function with output length proportional to the size of $\mathcal{M}$ and f_K is a CBC-MAC or a HMAC function. On the other hand, nothing seemed to be really known before this paper when collisions among the nonces are authorized.

The paper is composed of three parts. At first, we describe the mode of operations FRMAC (Sect. 2), then we analyze our construction in a quite general setting (Sect. 3) and finally, we take advantage of the generality of the security proof to propose extensions (Sect. 4). Some of them may be interesting for hardware targets.

Notations:

- $\mathcal{M}$ represents a message, that is a bit string. Its size is denoted $|\mathcal{M}|$ or m. The concatenation of two messages $\mathcal{M}$ and $\mathcal{N}$ is denoted $\mathcal{M}||\mathcal{N}$.
- H represents a family of 2^d hash functions. A function in the family is denoted h. The output of the hash functions has size n bits.
- F represents a family of functions. Elements of the family are denoted f. Likewise, Π represents a family of permutations and a permutation in the family is denoted π.
- Let (n, ν) be two integers. The set $\mathrm{Rand}(n, \nu)$ represents the set of all functions from $\{0,1\}^n$ to $\{0,1\}^\nu$.
- Let n be an integer. The set $\mathrm{Perm}(n)$ represents the set of all permutations of $\{0,1\}^n$.
- Let (k, n) be two integers. The set $\mathrm{FPerm}(k, n)$ represents the set of all families of 2^k permutations of $\{0,1\}^n$.

2 Fast Randomized MACs

2.1 Almost-Universal Families of Hash Functions

In the definition below, ε_m is a monotonic increasing function from $\mathbb{N}$ to $[1/2^n, \infty[$.

Definition 1. *A family H of 2^d hash functions is an ε_m-almost universal family (or ε_m-AU) of hash functions if, for all $\mathcal{M} \neq \mathcal{N}$ in $\{0,1\}^m$, $|\{h \in H \mid h(\mathcal{M}) = h(\mathcal{N})\}| \leqslant \varepsilon_m 2^d$.*

When $\varepsilon_m = \varepsilon$ is a constant, H is a classical ε-AU family of hash functions. In the cryptographic context, many families of ε-AU hash functions with fixed output length are ε_m-almost universal. Often, $\varepsilon_m = (cm + d)^e/2^n$ with c, d, e suitable positive constants.
Let now Π be a family of permutations of $\mathrm{Perm}(n)$, indexed by an r-bit block R. An element of this family is denoted π_R. The MAC that we study here is given by

$$\mathrm{FRMAC}_\Pi(\mathcal{M}) = (\pi_R(h(\mathcal{M})), R) \quad , \tag{1}$$

where R is an r-bit block randomly chosen for the MAC generation. The family Π and the function h are secret. In order to verify a MAC $\mathcal{C} = (T, R)$ corresponding to a message $\mathcal{M}$, we check whether $T = \pi_R(h(\mathcal{M}))$.

2.2 Fast Randomized MACs in Software

We focus in the paper on efficient MACs in software. Of course, the time needed for MACs generations or verifications depends on the size of the messages.

- For large messages, the costly part is the computation of h. The fastest such functions known for pentium microprocessors are the $\mathrm{UHash}\,16^e$ family of hash functions used in UMAC-2000 [15, 23]. Krovetz proved that these functions are ε^e-almost universal for $\varepsilon = 3/2^{16} + 1/2^{18} + 1/2^{28}$ and messages of size smaller than 2^{64} bits. For the purpose of completeness, we consider in the following a slight modification of these functions, the $\mathrm{UH}\,16^e$ functions, whose performance is similar to that of UMAC but which are easier to describe and analyze. We refer to [15] for complete details about the $\mathrm{UHash}\,16^e$ family of hash functions. With these settings, we obtain

  $$\mathrm{FRMAC}_E(\mathcal{M}) = (E_{K \oplus R}(\mathrm{UH}\,16^e_k(\mathcal{M})), R) \quad ,$$

  where K and k are two secret keys of $32e$-bit and $16920e$-bit sizes respectively, E_K is a block cipher with $16e$-bit blocks and R is a randomly chosen block of size $16e$ bits, padded to $32e$ bits with zeroes.

– For small messages (few bytes), the costly part is the computation of π_R. The smaller the block size of this permutation is, the better the performance is. In our construction, the block size can be close to $\log_2(1/\varepsilon)$ (cf. Sect. 3).

For a detailed description of the UH 16^e hash functions see appendix A.

Proposition 1. *Hash family* UH 16^e *is* ε_m^e*-AU with* $\varepsilon_m = 1/2^{15} + 3/2^{24} + m/2^{142}$.

Remark 1. In some applications, the key size of the hash functions can be prohibitive. This size can be significantly decreased with other efficient ε_m-AU hash families, like the HASH127 [5] or POLYR [14] families. Otherwise, like UMAC, a Toeplitz approach, due to Krawczyk [13], enables to save key materials between iterations of the functions UH $16_k^1(\mathcal{M})$.

Security Results. For messages $\mathcal{M}$ of size m at most 2^{118} bits, since in this case $\varepsilon_m < 1/2^{15} + 1/2^{22}$, Proposition 1 and Theorem 2 show that codes $(T_{\mathcal{M}}, R)$ of size $32e$ computed with FRMAC$_E$, where E is a block cipher with a $16e$-bit block size and a $32e$-bit key size yields a security bound close to $L(2^e + 2^{e-7} + 1)(16e + 1)/2^{16e}$ (L is the total size, in bits, of the messages which form the queries of the attacker).
For $e = 4$, we have 64-bit tags and one can use an efficient 64-bit block cipher (for instance some of the 64-bit block ciphers considered by the NESSIE european project [23]) in order to have a security bound of $L/2^{53}$. For $e = 8$, one can use the 256-bit key size version of the 128-bit block cipher AES to obtain a security bound of $L/2^{112}$.

Performance Results in Software. In order to compare the Wegman-Carter's approach $E_K(R) \oplus h_k(\mathcal{M})$ versus the FRMAC approach $E_{K \oplus R}(h(\mathcal{M}))$, we have to compare the time needed for one encryption with a fast $2n$-bit block cipher versus one key setup for a $2n$-bit key and one encryption with a fast n-bit block cipher.
For 64-bit tags, if we refer to measurements done on pentium processors for the NESSIE european project [24], one of the fastest 128-bit block cipher is the AES algorithm [22] while one of the fastest 64-bit block cipher is the IDEA algorithm [16].

– From [18], the encryption of one 128-bit block with AES needs at best $16 \times 14.13 \simeq 226$ cycles.
– From [17], the encryption of one 64-bit block with IDEA needs on MMX processors (when four encryptions are done at the same time) at best $8 \times 14.13 \simeq 110$ cycles. However, IDEA does exhibit weak key classes of substantial size and this might yield flaws with FRMAC [23]. But a classical Feistel ladder built over the IDEA round function (initialized with constants instead of subkeys) provides an easy substitute for the weak IDEA key setup. In this case, it seems reasonable to assume that the Feistel key setup is not slower than one 64-bit IDEA encryption and this yields a total amount of 220 cycles.

More generally when the size n of the tags increases, it would be very surprising that two n-bit encryptions (among which the time for one encryption is the time of a Feistel $2n$-bit key setup) could be done faster than one $2n$-bit encryption. Indeed, the security level required in full generality for a $2n$-bit encryption is higher than the security reached by two n-bit encryptions, and higher security often results in a speed cost.

3 Provable Security

For increased generality, we analyze the security of the FRMAC construction for any ε_m-AU family of hash functions.

3.1 Universal Hashing

In order to obtain the security proof of Sect. 3.3, we need two more precise results about ε_m-AU families of hash functions.

Lemma 1 (Collision in a group of messages). *Let $\mathcal{M}_1, \ldots, \mathcal{M}_q$ be a group of q distinct messages of sizes $m_1, \ldots m_q$ bits, then*

$$\mathbf{Pr}[\exists i \neq j \text{ such that } h(\mathcal{M}_i) = h(\mathcal{M}_j) \mid h \xleftarrow{R} H] \leqslant q \sum_{i=1}^{q} \varepsilon_{m_i} \ .$$

Proof. Since it is always possible to number the messages differently, we can assume that the sizes m_i are in decreasing order. Let $P = \mathbf{Pr}[\exists i \neq j \text{ such that } h(\mathcal{M}_i) = h(\mathcal{M}_j) \mid h \xleftarrow{R} H]$, then

$$P \leqslant \sum_{i=1}^{q} \sum_{j>i} \mathbf{Pr}[h(\mathcal{M}_j) = h(\mathcal{M}_i) \mid h \xleftarrow{R} H] \leqslant \sum_{i=1}^{q} \sum_{j>i} \varepsilon_{\max(m_j, m_i)} \leqslant \ q \sum_{i=1}^{q} \varepsilon_{m_i} \quad \text{(Def. 1)}.$$

□

Lemma 2 (Collision with a reference message). *Let $\mathcal{M}_0, \mathcal{M}_1, \ldots, \mathcal{M}_q$ be a group of $q+1$ distinct messages of sizes $m_0, m_1, \ldots, m_q$ bits, then*

$$\mathbf{Pr}[\exists i \in \{1, \ldots, q\} \text{ such that } h(\mathcal{M}_i) = h(\mathcal{M}_0) \mid h \xleftarrow{R} H] \leqslant q \max_{0 \leqslant i \leqslant q} \varepsilon_{m_i} \ .$$

Proof. Let $P = \mathbf{Pr}[\exists i \in \{1, \ldots, q\} \text{ such that } h(\mathcal{M}_i) = h(\mathcal{M}_0) \mid h \xleftarrow{R} H]$, then

$$P \leqslant \sum_{i=1}^{q} \mathbf{Pr}[h(\mathcal{M}_i) = h(\mathcal{M}_0) \mid h \xleftarrow{R} H] \leqslant \sum_{i=1}^{q} \varepsilon_{\max(m_0, m_i)} \leqslant q \max_{0 \leqslant i \leqslant q} \varepsilon_{m_i} \quad \text{(Def. 1)} \ .$$

□

3.2 Adversary

Before giving the security result, we first define the adversary against the FRMAC construction.

Oracles. In the general case of the FRMAC$_\Pi$ construction, where Π is some family of permutations, the adversary can access the construction through two oracles.

- The generation oracle $\mathcal{O}_G$ chooses a random value $R \in \{0,1\}^r$ and computes the MAC $\mathcal{C} = (\pi_R(h(\mathcal{M})), R)$.
- The verification oracle $\mathcal{O}_V$ verifies whether $\pi_R(h(\mathcal{M})) = T$ for a given triplet $(\mathcal{M}, T, R)$.

The adversary is allowed to ask the generation of different MACs for a single message. In this case, since the adversary can always get rid of duplicates, we assume that a different R is chosen each time. We also assume that the adversary never asks a clearly invalid verification query. He thus never asks the verification of $(\mathcal{M}, T', R)$ where $T \neq T'$ and where (T, R) is a MAC generated by $\mathcal{O}_G$ for $\mathcal{M}$.

Parameters of the adversary. An adversary is defined through three parameters. The number L is the total size, in bits, of the messages which form the queries to both oracles. The number τ measures the maximal execution time of the adversary. The number μ represents a bound on the size of code of the adversary.

Goal of the adversary. The goal of the adversary is to successfully forge a valid MAC for a message $\mathcal{M}$. When this happens, we say that the adversary wins and the security of the construction is broken. Such an event happens when the verification oracle outputs 'MAC Valid' on a request $(\mathcal{M}, T, R)$ where (T, R) was not an output of the generation oracle for $\mathcal{M}$. The probability of success of the adversary $\mathcal{A}$ is denoted $\mathbf{Adv}^{\text{forge}}_{\text{FRMAC}}(\mathcal{A})$. As seen above, this is his probability of getting the answer 'MAC valid' with access to the two oracles $\mathcal{O}_G$ and $\mathcal{O}_V$:

$$\mathbf{Adv}^{\text{forge}}_{\text{FRMAC}}(\mathcal{A}) = \mathbf{Pr}[\mathcal{A}^{\mathcal{O}_G, \mathcal{O}_V} \text{ gets the answer 'MAC valid'}] \ .$$

We override this notation and we write $\mathbf{Adv}^{\text{forge}}_{\text{FRMAC}}(\tau, \mu, L)$ for the maximum value of the previous advantage over all adversaries of parameters (τ, μ, L).

3.3 Security in the Information Theoretic Model

Let FRMAC_{Π} be the construction defined in Sect. 2.1 by (1). We study, first, its security in the information theoretic model. That is, we assume that the adversary has unlimited computational power and that his only limiting parameter is L, the total number of bits in the queries to the oracles.

Security of FRMAC. The security of FRMAC_{Π} in the information theoretic model is given by Theorem 1.

Theorem 1. *Let Π be a family of 2^r random permutations of* $\text{Perm}(n)$. *If $r = n$ and if $n \geqslant 3$, we have, with* $\Sigma_L = \dfrac{1}{L} \displaystyle\max_{\substack{\forall \mathcal{M}_i \text{ s.t.} \\ \sum |\mathcal{M}_i| = L}} \sum_i \varepsilon_{m_i}$,

$$\mathbf{Adv}^{\text{forge}}_{\text{FRMAC}_{\Pi}}(\tau, \mu, L) \leqslant L \left(\frac{n+1}{2^n} + n\Sigma_L + \varepsilon_L \right) + \frac{1}{2^n} \ .$$

It is not difficult to notice that $\Sigma_L \leqslant \varepsilon_L$. Theorem 1 thus yields

$$\mathbf{Adv}^{\text{forge}}_{\text{FRMAC}_{\Pi}}(\tau, \mu, L) \leqslant L\,(n+1) \left(\varepsilon_L + \frac{1}{2^n} \right) + \frac{1}{2^n} \ . \tag{2}$$

In order to prove Theorem 1, we need three lemmas.

Lemma 3. *Let Π be a family of 2^r random permutations of* $\text{Perm}(n)$ *and let $\mathcal{A}$ be an adversary with access to the generation oracle of* FRMAC_{Π}. *Let $r = n$ and let* $\texttt{No}(R)$ *represents the number of times the value R is chosen by the oracle during the execution of the adversary. If $L \leqslant 2^n/4$, the probability that a single permutation is queried more than n times is bounded by*

$$\mathbf{Pr}[\exists R \text{ s.t. } \texttt{No}(R) \geqslant n] \leqslant \frac{1}{2^n} \ .$$

Lemma 4. *Let F be a family of 2^r random functions of* $\mathrm{Rand}(n,n)$*. Assume that $r = n$ and that no value R is chosen more than n times by the generation oracle, then we have, with* $\Sigma_L = \frac{1}{L} \max_{\substack{\forall \mathcal{M}_i \text{ s.t.} \\ \sum |\mathcal{M}_i| = L}} \sum_i \varepsilon_{m_i}$,

$$\mathbf{Adv}^{\mathrm{forge}}_{\mathrm{FRMAC}_F}(\tau, \mu, L) \leqslant L\,(n\Sigma_L + \varepsilon_L) \ .$$

Lemma 5. *Let F be a family of 2^r random functions of* $\mathrm{Rand}(n,n)$*, with $n \geqslant 2$, and let Π be a family of 2^r random permutations of* $\mathrm{Perm}(n)$*. Assume that $r = n$ and that no value R is chosen more than n times by the generation oracle, then we have*

$$\mathbf{Adv}^{\mathrm{FRMAC}_F}_{\mathrm{FRMAC}_\Pi}(\tau, \mu, L) \leqslant \ L\ \frac{n+1}{2^n} \ .$$

Here, the advantage $\mathbf{Adv}^{\mathrm{FRMAC}_F}_{\mathrm{FRMAC}_\Pi}(\mathcal{A})$ represents the probability, for an adversary $\mathcal{A}$, of distinguishing the construction FRMAC_F using functions, from the construction FRMAC_Π using permutations.

Proof of the theorem 1. The proof of Theorem 1 can then be easily deduced from Lemmas 3, 4 and 5. First, we observe that, since $n \geqslant 3$, the inequality of Theorem 1 is obviously true for $L > 2^n/4$. So, from now, we assume that $L \leqslant 2^n/4$.

If the generation oracle chooses a single value R more than n times, we say that the adversary wins and we stop. According to Lemma 3, this gives the adversary a probability $1/2^n$ of winning this way. If this event does not occur, we can apply Lemmas 5 and 4. In the following, we assume that we are in this case.

Let $\mathcal{A}$ be an adversary that forges against FRMAC_Π. He has a probability $\mathbf{Adv}^{\mathrm{forge}}_{\mathrm{FRMAC}_\Pi}(\mathcal{A})$ of success. If we execute $\mathcal{A}$ against FRMAC_F and if, according to his own criteria, he is successful, then $\mathcal{A}$ will either produce a valid forgery against FRMAC_F — and his probability of doing so is bounded by $\mathbf{Adv}^{\mathrm{forge}}_{\mathrm{FRMAC}_F}(\tau, \mu, L)$ — or he will have distinguished FRMAC_F from FRMAC_Π — and his probability of doing so is bounded by $\mathbf{Adv}^{\mathrm{FRMAC}_F}_{\mathrm{FRMAC}_\Pi}(\tau, \mu, L)$. Thus his probability of forging against FRMAC_Π cannot exceed the sum $\mathbf{Adv}^{\mathrm{forge}}_{\mathrm{FRMAC}_F}(\tau, \mu, L) + \mathbf{Adv}^{\mathrm{FRMAC}_F}_{\mathrm{FRMAC}_\Pi}(\tau, \mu, L)$.

Adding the probability $1/2^n$ of winning due to an excessive number of identical Rs, we finally obtain the probability that the adversary wins in the case $L \leqslant 2^n/4$:

$$\mathbf{Adv}^{\mathrm{forge}}_{\mathrm{FRMAC}_\Pi}(\mathcal{A}) \leqslant \frac{1}{2^n} + \mathbf{Adv}^{\mathrm{forge}}_{\mathrm{FRMAC}_F}(\tau, \mu, L) + \mathbf{Adv}^{\mathrm{FRMAC}_F}_{\mathrm{FRMAC}_\Pi}(\tau, \mu, L) \ .$$

Since this inequality stands for all adversaries of parameters (τ, μ, L), we get,

$$\begin{aligned} \mathbf{Adv}^{\mathrm{forge}}_{\mathrm{FRMAC}_\Pi}(\tau, \mu, L) &\leqslant \frac{1}{2^n} + \mathbf{Adv}^{\mathrm{forge}}_{\mathrm{FRMAC}_F}(\tau, \mu, L) + \mathbf{Adv}^{\mathrm{FRMAC}_F}_{\mathrm{FRMAC}_\Pi}(\tau, \mu, L) \ , \\ &\leqslant \frac{1}{2^n} + L\,(n\Sigma_L + \varepsilon_L) + L\,\frac{n+1}{2^n} \ . \end{aligned}$$

Since the inequality also stands for $L > 2^n/4$, this concludes the proof of Theorem 1.

□

Proof of Lemma 3. This lemma states that, with high probability, only a few messages will be computed using the same value of R.
We have $L \leqslant 2^r/4$. The probability of getting more than r identical values of R is bounded by:

$$\begin{aligned}\mathbf{Pr}[\exists R \text{ s.t. } \mathtt{No}(R) \geqslant r] &\leqslant \sum_R \left(\sum_{i_1<i_2<\cdots<i_r} \mathbf{Pr}[R_{i_1} = R] \cdots \mathbf{Pr}[R_{i_r} = R] \right) \ , \\ &\leqslant \sum_R \left(\binom{L}{r} \left(\frac{1}{2^r}\right)^r \right) \leqslant 2^r \left(L^r \left(\frac{1}{2^r}\right)^r \right) \ , \\ &\leqslant 2^r \left(\left(\frac{2^r}{4}\right)^r \left(\frac{1}{2^r}\right)^r \right) \leqslant \frac{1}{2^r} \ .\end{aligned}$$

With probability at most $1/2^r$, a value of R is shared among more than r messages. If $r = n$, we get $\mathbf{Pr}[\exists R \text{ s.t. } \mathtt{No}(R) \geqslant n] \leqslant \dfrac{1}{2^n}$. □

Proof of Lemma 4. We want to bound $\mathbf{Adv}^{\text{forge}}_{\text{FRMAC}_F}(\mathcal{A})$ for all adversaries $\mathcal{A}$ of parameters (τ, μ, L). The adversary wins when he gets the answer 'MAC valid' from the verification oracle. We also declare that the adversary wins when he gets a collision on the outputs of the generation oracle, that is, when he gets the same MAC for two different messages.

First, we observe that all the verification requests of an adversary can be sent at the end of his execution, without changing his probability of success. This is done as follows. When the adversary wants to call the verification oracle during his execution, he stores instead the query and assumes that the answer of the oracle was 'MAC invalid'. Then, when he reaches the end of his execution, the adversary sends all the stored queries to the verification oracle. This does not change the probability of success. Indeed, if the answer to the verification query was 'MAC invalid', nothing is changed from the adversary's viewpoint between the two executions. If the answer was instead 'MAC valid', it only delays the moment the adversary wins: the stored query is simply executed later, at the end of all the generation queries.
In the following, we thus assume that the adversary first do all his generation queries and then his verification queries.

We observe also that, while none of the two winning events has occurred, the adversary has only seen outputs of different random functions on different inputs. He has thus only seen random values. Any adaptive adversary can be replaced by a non adaptive adversary that performs as well. The non-adaptive adversary simply precomputes his queries assuming that the answers that he will receive are random values.
In the following, we thus assume that the adversary does not adapt his generation queries to the answers of the oracle.

Since the adversary $\mathcal{A}$ is not adaptive, we can bound his probability of success assuming the messages $\mathcal{M}$ are independent from h and F. We can apply Lemmas 1 and 2.
The adversary is successful if one of two events occurs.

- E1: The verification oracle outputs 'MAC valid'.
- E2: The generation oracle outputs two identical MACs.

There are two cases to consider with E1:

- The query of the adversary is a 'collision test', that is the adversary asks $(\mathcal{M}, T, R)$ where (T, R) was obtained for a different message $\mathcal{M}'$ through the generation oracle.

– The query of the adversary is a 'random guess', that is the adversary asks $(\mathcal{M}, T, R)$ where (T, R) was not obtained through the generation oracle.

In the case of 'random guesses', the probability of event E1 is the probability of successfully guessing the output of a random function. Since the output size is n bits and since the adversary gets at most L tries, we have:

$$\mathbf{Pr}[\text{E1 with 'random guesses'}] \leqslant \frac{L}{2^n} \ .$$

In the case of 'collision tests', let $\mathcal{M}$ be a message which got a MAC $\mathcal{C} = (T, R)$ through the generation oracle. And let $L_{\mathcal{C}}$ be the number of bits of messages tested with the tag $\mathcal{C}$ through the verification oracle. The probability that the MAC $\mathcal{C}$ is valid for one of those messages is equal to the probability of one of the messages having the same output than $\mathcal{M}$ through the hash function h.

$$\begin{aligned}\mathbf{Pr}[\text{E1 with 'collision tests'}] &\leqslant \sum_{\mathcal{C}} \mathbf{Pr}[\text{One of the } L_{\mathcal{C}} \text{ messages has same output than } \mathcal{M}] \ , \\ &\leqslant \sum_{\mathcal{C}} L_{\mathcal{C}} \max_{0 \leqslant i \leqslant L_{\mathcal{C}}} \varepsilon_{m_i} \leqslant \sum_{\mathcal{C}} L_{\mathcal{C}}\, \varepsilon_L \leqslant L\, \varepsilon_L \quad \text{(Lemma 2)} \ .\end{aligned}$$

Now we evaluate the probability of event E2. The probability of getting two identical MACs is equal to the probability of simultaneously getting the same R and the same output of the hash function. Recall that no value of R is shared among more than n messages. We get:

$$\begin{aligned}\mathbf{Pr}[\text{E2}] &= \sum_{R} \mathbf{Pr}[\exists \text{ collisions within a group of messages indexed by } R] \ , \\ &\leqslant \sum_{R} n \sum_{\substack{\text{messages in} \\ \text{the group}}} \varepsilon_{m_i} \leqslant L\, n \Sigma_L \quad \text{(Lemma 1)} \ .\end{aligned}$$

Finally, adding the probabilities of the two events, we obtain:

$$\mathbf{Adv}^{\text{forge}}_{\text{FRMAC}_F}(\mathcal{A}) \leqslant L\,(n\Sigma_L + \varepsilon_L) \ .$$

□

Proof of Lemma 5. We want to bound $\mathbf{Adv}^{\text{FRMAC}_F}_{\text{FRMAC}_\Pi}(\mathcal{A})$ for all adversaries $\mathcal{A}$ of parameters (τ, μ, L).
The adversary tries to distinguish the FRMAC construction using functions, from the FRMAC construction using permutations. We also say that the adversary wins if the verification oracle outputs 'MAC valid'. Thus, with the same arguments as in the proof of Lemma 4, we can assume that the verification queries are all done at the end of the execution. We separately study the probability of success of the adversary with the generation queries and with the verification queries.
During the generation phase, the adversary tries to distinguish functions from permutations. Each of them is called at most n times. We can apply the PRF/PRP switching lemma of [2]. Let q_R denote the number of calls to either f_R or π_R. We get:

$$\mathbf{Pr}[\mathcal{A} \text{ wins with generation queries}] \leqslant \sum_{R} \frac{q_R^2}{2^{n+1}} \leqslant \sum_{R} \frac{n \cdot q_R}{2^{n+1}} \leqslant \frac{n}{2^{n+1}} \times \sum_{R} q_R \leqslant \frac{nL}{2^{n+1}} \ .$$

During the verification phase, the adversary can do either 'collision tests' (he verifies a MAC previously generated for another message $\mathcal{M}$) or 'random guesses' (he verifies a completely new MAC).
In the 'collision tests', the difference between permutations and functions appears when the function gives the same output on two different points. This happens with probability $1/2^n$ for each try. The advantage gained after at most L queries is thus bounded by $L/2^n$.
In the 'random guesses', the difference between permutations and functions is that permutations cannot give the same output on two different points. A correct guess happens with probability $1/2^n$ for a function and with probability $1/(2^n - q_R)$ for a permutation, where q_R represents the number of known values of the permutation π_R after the generation phase. Each try thus gives an advantage $1/(2^n - q_R) - 1/2^n$. Since $q_R \leqslant n$, the advantage gained after at most L queries is bounded by $L/(2^n - n) - L/2^n$.
Summing the two contributions, we find that the adversary gains an advantage $L/(2^n - n)$ in the verification phase.
Finally, the advantages of the generation and verification queries sum up to:

$$\mathbf{Adv}^{\mathrm{FRMAC}_F}_{\mathrm{FRMAC}_\Pi}(\mathcal{A}) \leqslant \frac{L}{2^n - n} + \frac{nL}{2^{n+1}} \leqslant L\ \frac{n+1}{2^n}\ ,$$

for $n \geqslant 2$. □

3.4 Security in the Standard Model

Now we study the security of the FRMAC_E construction in the standard model for a block cipher E. The adversary no longer has unlimited computational power but he can also win by distinguishing the block cipher E from a random family of permutations.

MAC Definition. Let H be an ε_m-AU family of hash functions. Let E be a block cipher with blocks of size n and keys of size k, with $n \leqslant k$. The MAC that we consider in this model is the following:

$$\mathrm{FRMAC}_E(\mathcal{M}) = (E_{K \oplus R}(h(\mathcal{M})), R)\ ,$$

where R is an n-bit block, randomly chosen for the MAC generation. The random value R is of size n. If $k > n$, the block R is padded with zeroes to perform the `XOR` operation.

The PRP-RKA Model. Since the adversary has access to the block cipher with different keys, we need to use the model proposed by Bellare and Kohno in [3]. This model allows proofs in the standard model in the case of related keys attacks:

$$\mathbf{Adv}^{\textbf{prp-rka}}_{\mathtt{XOR}_r,E}(\mathcal{A}) = \Big|\mathbf{Pr}[K \xleftarrow{\mathrm{R}} \{0,1\}^k\ :\ \mathcal{A}^{E_{\cdot \oplus K}(\cdot)} = 1] - \mathbf{Pr}[K \xleftarrow{\mathrm{R}} \{0,1\}^k, \Pi \xleftarrow{\mathrm{R}} \mathrm{FPerm}(k,n)\ :\ \mathcal{A}^{\Pi_{\cdot \oplus K}(\cdot)} = 1]\Big|\ .$$

The attack model allows the adversary to choose an r-bit number R and then obtain the value of the block cipher, on an input of his choice, under the key $R \oplus K$ (if $r < k$, the value R is padded with zeroes for this operation). This advantage measures the success of the adversary in determining whether the oracle uses the block cipher E or a random family of permutations. As before, we override this notation for all adversaries of parameters τ, μ, L. Theorem 2 gives the security of the construction in the computational model.

Theorem 2. *Let E be a block cipher with blocks of size n and keys of size k, with $n \leqslant k$. If $n \geqslant 3$, we have,*

$$\mathbf{Adv}^{\text{forge}}_{\text{FRMAC}_E}(\tau, \mu, L) \leqslant L\,(n+1)\left(\frac{1}{2^n} + \varepsilon_L\right) + \frac{1}{2^n} + \mathbf{Adv}^{\mathbf{prp\text{-}rka}}_{\mathtt{XOR}_n, E}(\tau, \mu, L) \ . \tag{3}$$

The proof of this theorem derives immediately from Theorem 1 and the definition of the advantage $\mathbf{Adv}^{\mathbf{prp\text{-}rka}}_{\mathtt{XOR}_n, E}(\tau, \mu, L)$. Indeed, if an adversary is able to forge a valid MAC against FRMAC_E, he is either able to forge directly against FRMAC_Π or he provides a way to distinguish between the two situations.
Most modern concrete block ciphers, including the AES, are designed with the goal of resisting $\mathtt{XOR}_n$ related key attacks. Especially, for block ciphers with keys of size k two times larger than their n-bit block, it seems reasonable to assume that $\mathbf{Adv}^{\mathbf{prp\text{-}rka}}_{\mathtt{XOR}_n, E}(\tau, \mu, L) \simeq \mathbf{Adv}^{\mathbf{prp}}_{E}(\tau, \mu, L) \simeq L/2^n$.

4 Extensions

As noticed by Bernstein at the end of [5], many efficient ε_m-AU families of hash functions can be constructed using reductions modulo secret ideals in rings.
More precisely, let $\mathcal{R}$ be a commutative ring with an identity element and let p be an arbitrary injective function from a set of messages $\{0,1\}^*$ to a subset S of $\mathcal{R}$ such that, for any messages $\mathcal{M}$ and $\mathcal{N}$, $p(\mathcal{M}) - p(\mathcal{N}) \in S$. We denote by S_m the finite subsets of S such that for any messages $\mathcal{M}$ and $\mathcal{N}$ in $\{0,1\}^m$, $p(\mathcal{M}), p(\mathcal{N}), p(\mathcal{M}) - p(\mathcal{N}) \in S_m$ and such that $S_1 \subset S_2 \subset \cdots \subset S$. Moreover, let $I = (\mathcal{I}_i)_{1,\ldots,2^d}$ be a set of 2^d distinct ideals of $\mathcal{R}$. For our cryptographic purposes, these ideals must satisfy two conditions.

Cond. 1: For any element $\mathcal{I}_i$ of I, the domain $\mathcal{R}/\mathcal{I}_i$ is finite. We denote by 2^n the power of two greater than the cardinality of the largest such domain and by q_i an arbitrary injective function from $\mathcal{R}/\mathcal{I}_i$ to $\{0,1\}^n$.
Cond. 2: There exists a monotonic increasing function ε_m such that,

$$\forall z \in S_m,\ |\{\mathcal{I} \in I \mid z \bmod \mathcal{I} = 0\}| < \varepsilon_m 2^d \ .$$

With these notations, we consider the family H of hash functions $(h_i)_{1,\ldots,2^d}$ defined by,

$$h_i : \{0,1\}^* \longrightarrow \{0,1\}^n,\ \mathcal{M} \longrightarrow q_i(p(\mathcal{M}) \bmod \mathcal{I}_i) \ .$$

Lemma 6. *H is an ε_m-AU family of hash functions.*

Proof. Let $\mathcal{M}$ and $\mathcal{N}$ be two distinct messages of $\{0,1\}^m$, then

$$\begin{aligned}
|\{h \in H \mid h(\mathcal{M}) = h(\mathcal{N})\}| &= |\{i \mid q_i(p(\mathcal{M}) \bmod \mathcal{I}_i) = q_i(p(\mathcal{N}) \bmod \mathcal{I}_i)\}| \ , \\
&= |\{i \mid p(\mathcal{M}) - p(\mathcal{N}) \bmod \mathcal{I}_i = 0\}| \quad (q_i \text{ injective}) \ , \\
&\leqslant \max_{z \in S_m} |\{i \mid z \bmod \mathcal{I}_i = 0\}| \quad (p \text{ injective}) \ , \\
&\leqslant \varepsilon_m 2^d \quad \text{(Cond. 2)} \ .
\end{aligned}$$

□

Conditions 1 and 2 can seem, at a first glance, a little obscure. They are consequences of mathematical properties. Cond. 1 is, for instance, true for maximal ideals in rings such that their additive group is a finitely generated abelian group [9, Exercise 4.26 p.142], which is generally true for usual mathematic constructions over finite fields. Cond. 2 is a corollary of the uniqueness and the finiteness of the ideal factorization for a large class of rings. It is well known that in Dedekind rings which occurs, for instance, in the context of the global fields (i.e, number fields or function fields), we have unique factorizations in prime ideals [20]. For the more general case of the noetherian rings (like multivariate polynomial algebras), we have a similar result for primary decomposition in maximal ideals [19].

Let $\mathbb{F}_q$ be a finite field and $\log_2 q = n$, we give examples of such a construction in Table 1 and, for each of them, an approximation $\tilde{\varepsilon}_m$ of the corresponding collision bound ε_m.

Table 1. Examples of ε_m-almost-universal hash families

$\mathcal{R}$	I	S_m	$p(\mathcal{M})$	$\tilde{\varepsilon}_m$
$\mathbb{Z}$	$\{(\pi) \mid \pi \text{ prime}, 2^{n-1} < \pi < 2^n\}$	$\{z \in \mathcal{R} \mid \lvert z\rvert \leqslant 2^m\}$	$\sum_{\lvert M_i\rvert=1} M_i 2^i$	$\frac{m}{2^n}$
$\mathbb{F}_2[X]$	$\{(\pi(X)) \mid \pi(X) \text{ irreducible}, \deg \pi(X) = n\}$	$\{z \in \mathcal{R} \mid \deg(z) \leqslant m\}$	$\sum_{\lvert M_i\rvert=1} M_i X^i$	$\frac{m}{2^n}$
$\mathbb{F}_q[X]$	$\{(X-\alpha) \mid \alpha \in \mathbb{F}_q\}$	$\{z \in \mathcal{R} \mid \deg(z) \leqslant \frac{m}{n}\}$	$\sum_{\lvert M_i\rvert=n} M_i X^i$	$\frac{m}{n\,2^n}$
$\mathbb{F}_q[X,Y]$	$\{(X-\alpha, Y-\beta) \mid \alpha,\beta \in \mathbb{F}_q\}$	$\left\{z \in \mathcal{R} \mid \deg(z) \leqslant \frac{m}{n}\right\}$	$\sum_{\lvert M_{ij}\rvert=n} M_{ij} X^i Y^j$	$\frac{\sqrt{m}}{\sqrt{n}\,2^n}$

Such a generalization combined with the security results of Sect. 3 enables the design of FRMAC constructions which is adapted to non software targets. The reduction modulo an ideal of the form $(X-\alpha)$ in $\mathbb{F}_{2^{128}}$, for instance, yields a $(m/128 \times 1/2^{128})$-AU linear hash family which is well suited for hardware applications.

5 Conclusion

In this paper, we studied the FRMAC message authentication code, a RMAC construction in which an ε-almost universal hash family is used instead of a CBC chain. This construction enables random nonces. When tags and nonces are of size n bits, we show as a consequence that the FRMAC security bound in the information theoretic model (2) is only $n+1$ times larger than the classical bound for Wegman-Carter's MACs. Furthermore, since FRMAC takes advantage of the resistance of most of the block ciphers to `XOR`-related key attacks, the security bound in the computational model (3) is not penalized by the PRF/PRP switching overhead. This finally yields a bound which is, as in RMAC, beyond the birthday paradox. Thus, for a given security level, FRMAC is able to use block ciphers with smaller block sizes, without any unicity requirement on the nonces.
In return, implementing it in real applications tends to be easier. Used with efficient hash functions in software, FRMAC reaches a performance for large messages similar to the fastest previously known constructions that achieve the same security level. For small messages, FRMAC might be even more efficient and instances with the AES yield a very secure design with a security bound close to $L/2^{120}$.

References

1. M. Bellare, O. Goldreich, and H. Krawzyck. Stateless Evaluation of Pseudorandom Functions: Security beyond the Birthday Barrier. In *Advances in Cryptology – CRYPTO '99*. Springer-Verlag, August 1999.
2. M. Bellare, J. Killian, and P. Rogaway. The Security of the Cipher Block Chaining Message Authentication Code. In *Advances in Cryptology — CRYPTO'94*, volume 839 of *Lecture Notes in Computer Science*, pages 341–358. Springer, 1994.
3. M. Bellare and T. Kohno. A Theoretical Treatment of Related-Key Attacks: RKA–PRPs, RKA–PRFs, and Applications. In E. Biham, editor, *Advances in Cryptology — Proceedings of EUROCRYPT 2003*, Lecture Notes in Computer Science. Springer, 2003.
4. Mihir Bellare, Ted Krovetz, and Phillip Rogaway. Luby-Rackoff Backwards: Increasing Security by Making Block Ciphers Non-Invertible. In K. Nyberg, editor, *Advances in Cryptology - EuroCrypt '98*, volume 1403, Berlin, october 1998. Springer-Verlag,. Lecture Notes in Computer Science.
5. D.J. Bernstein. Floating-Point Arithmetic and Message Authentication, March 2000. Submitted for publication.
6. J. Black, S. Halevi, H. Krawczyk, T. Krovetz, and P. Rogaway. UMAC: Fast and Secure Message Authentication. In *Advances in Cryptology – CRYPTO '99*. Springer-Verlag, August 1999.
7. J.L. Carter and M.N. Wegman. Universal classes of hash functions. *Journal of computer and system sciences*, 18:143–154, 1979.
8. J.L. Carter and M.N. Wegman. New hash functions and their use in authentication and set equality. *Journal of computer and system sciences*, 22(18):265–279, 1981.
9. David Eisenbud. *Commutative Algebra with a View Toward Algebraic Geometry*. Number 150 in Graduate Texts in Mathematics. Springer, 1995.
10. Éliane Jaulmes. *Analyse de sécurité de schémas cryptographiques*. PhD thesis, École Polytechnique, June 2003.
11. Éliane Jaulmes, Antoine Joux, and Frédéric Valette. On the Security of Randomized CBC-MAC beyond the Birthday Paradox Limit. A New Construction. In Joan Daemen and Vincent Rijmen, editors, *Fast Software Encryption 2002*, volume 2365 of *Lecture Notes in Computer Science*, pages 237–251. Springer Verlag, February 2002.
12. T. Kohno, J. Viega, and D. Whiting. CWC: A high-performance conventional authenticated encryption mode. In *Fast Software Encryption*, Lecture Notes in Computer Science. Springer-Verlag, 2004. To appear.
13. H. Krawczyk. LFSR-Based Hashing and Authentication. In Y. Desmedt, editor, *Advances in Cryptology — CRYPTO '94*, volume 839 of *Lecture Notes in Computer Science*, pages 129–139. Springer, 1994.
14. T. Krovetz and P. Rogaway. Fast Universal Hashing with Small Keys and no Preprocessing: The PolyR Construction. In D.H. Won, editor, *ICIS 2000*, volume 2015 of *Lecture Notes in Computer Science*, pages 73–89. Springer-Verlag, 2000.
15. Theodore T. Krovetz. *Software-Optimized Universal Hashing and Message Authentication*. PhD thesis, University of California Davis, September 2000.
16. X. Lai and J. L. Massey. A proposal for a new block encryption standard. In Ivan B. Damgård, editor, *Advances in Cryptology - EuroCrypt '90*, pages 389–404, Berlin, 1990. Springer-Verlag. Lecture Notes in Computer Science Volume 473.
17. Helger Lipmaa. IDEA, a cipher for multimedia architecture ? In *Selected Areas in Cryptology'99*, volume 1556 of *Lecture Notes in Computer Science*, pages 256–268. Springer-Verlag, 1999.
18. Helger Lipmaa. Fast software implementation of SC2000. In *Information Security Conference 2002*, volume 2433 of *Lecture Notes in Computer Science*, pages 63–74. Springer-Verlag, 2002.
19. Hideyuki Matsumura. *Commutative Ring Theory*, volume 8 of *Cambridge studies in advanced mathematics*. Cambridge University Press, 1986.
20. Jürgen Neukirch. *Algebraic Number Theory*, volume 322 of *Grundlehren der mathematischen Wissenschaften*. Springer, 1999.

21. Wim Nevelsteen and Bart Preneel. Software Performances of Universal Hash Functions. In J. Stern, editor, *Advances in Cryptology — EUROCRYPT '99*, volume 1592 of *Lecture Notes in Computer Science*, pages 24–41. Springer, 1999.
22. NIST. *Specification for the Advanced Encryption Standard (AES)*, November 2001. Federal Information Processing Standards publication 197.
23. B. Preneel, A. Biryukov, E. Oswald, B. van Rompay, L. Granboulan, E. Dottax, S. Murphy, A. Dent, J. White, M. Dichtl, S. Pyka, Schafheutle, P. Serf, E. Biham, E. Barkan, O. Dunkelman, M. Ciet, F. Sica, and H. Raddum L. Knudsen. Nessie security report. Technical report, NESSIE, October 2002. NES/DOC/ENS/WP5/D20/1.
24. B. Preneel, B. Van Rompay, S.B. Ors, A. Biryukov, L. Granboulan, E. Dottax, M. Dichtl, M. Schafheutle, P. Serf, S. Pyka, E. Biham, E. Barkan, O. Dunkelman, J. Stolin, M. Ciet, J-J. Quisquater, F. Sica 5 H.Raddum, and M. Parker. Performance of optimized implementations of the nessie primitives. Technical report, NESSIE, October 2002. NES/DOC/TEC/WP6/D21/1.

A UH 16^e Hash Functions

The functions $\mathrm{UH}\,16^e_k$ are defined from $\{0,1\}^*$ to $\{0,1\}^{16e}$, for $e = 1, \ldots, 16$ and $k = (k_1, \ldots, k_e)$, by $\mathrm{UH}\,16^1_{k_1}(\mathcal{M})||\ldots||\,\mathrm{UH}\,16^1_{k_e}(\mathcal{M})$ where, for $\kappa = (\kappa_1, \kappa_2, \kappa_3)$,

$$\mathrm{UH}\,16^1_\kappa(\mathcal{M}) = \mathrm{IPHash}\,16_{\kappa_3}(\mathrm{PHash}\,128_{\kappa_2}(\mathrm{NHHash}\,16_{\kappa_1}(\mathcal{M})))\ .$$

The functions IPHash 16, PHash 128 and NHHash 16 are defined as follows.

– The function $\mathrm{NHHash}\,16_\kappa$ is based on the NHS 16 function. Mainly, let $\kappa = (\kappa_1, \ldots, \kappa_{1024})$ be a 2^{14}-bit key, let $\mathcal{M} = \mathcal{M}_1||\ldots||\mathcal{M}_b$ where $|\mathcal{M}_1| = \cdots = |\mathcal{M}_{b-1}| = 2^{14}$, let $+_{16}$ (resp. $\times_{32}$) denote the signed addition (resp. multiplication) modulo 2^{16} (resp. modulo 2^{32}), then

$$\mathrm{NHHash}\,16_\kappa(\mathcal{M}) = \mathrm{NHS}\,16_\kappa(\mathcal{M}_1)||\ldots||\,\mathrm{NHS}\,16_\kappa(\mathcal{M}_{b-1})||\,\mathrm{NHS}\,16_\kappa(\mathcal{M}_b)\ ,$$

where, with $M = M_1||\ldots||M_b$ such that M_i is considered as an integer in $[-2^{15}, 2^{15}-1]$ for $1024 \geqslant b \geqslant i \geqslant 1$ and with $M_i = 0$ for $i > b$,

$$\mathrm{NHS}\,16_\kappa(M) = (M_1 +_{16} \kappa_1) \times_{32} (M_2 +_{16} \kappa_2) + \cdots + (M_{1023} +_{16} \kappa_{1023}) \times_{32} (M_{1024} +_{16} \kappa_{1024}) +_{32} |M|\ .$$

– The function $\mathrm{PHash}\,128_\kappa$ is more classically based on the reduction modulo the prime p_{128} equal to $2^{128} - 159$. Let κ be a 128-bit key, let $\mathcal{M}||1 = \mathcal{M}_1||\ldots||\mathcal{M}_b$ where $|\mathcal{M}_1| = \cdots = |\mathcal{M}_{b-1}| = 120$, then,

$$\mathrm{PHash}\,128_\kappa(\mathcal{M}) = \kappa^b + \sum_{i=0}^{b-1} \kappa^i \mathcal{M}_{b-i} \bmod p_{128}\ .$$

– The function $\mathrm{IPHash}\,16_\kappa$ folds its 128-bit input into a 16-bit output. Again, let p_{24} be the prime equal $2^{24} - 3$, let $\kappa = (\kappa_0, \ldots, \kappa_{16})$ be a 17×24-bit key, let $\mathcal{M} = \mathcal{M}_1||\ldots||\mathcal{M}_{16}$ where $|\mathcal{M}_i| = 16$, then

$$\mathrm{IPHash}\,16_\kappa(\mathcal{M}) = (\kappa_0 + \left(\sum_{i=1}^{16} \kappa_i \mathcal{M}_i \bmod p_{24}\right) \bmod 2^{24}) \div 2^8\ .$$

From [21, 15], we have the following results.

Lemma 7. *Hash family* NHHash 16 *is* ε_m*-AU with* $\varepsilon_m = 1/2^{15}$.

Lemma 8. *Hash family* PHash 128 *is* ε_m*-AU with* $\varepsilon_m = m/2^{133}$.

Lemma 9. *Hash family* PHash 128 $\circ$ NHHash 16 *is* ε_m*-AU with* $\varepsilon_m = 1/2^{15} + m/2^{142}$.

Proposition 2. *Hash family* UH 16^e *is* ε_m^e*-AU with* $\varepsilon_m = 1/2^{15} + 3/2^{24} + m/2^{142}$.

Remark 2. The main difference between the UH 16 and the UHash 16 hash families is the PHash 128 hash family. Krovetz et al. [15, 23] propose a "ramped" strategy with, instead of a single 128-bit prime p_{128}, intermediate 32-bit and 64-bit primes in order to have slightly better performance results for messages of small size.