Authenticated Encryption: How Reordering can Impact Performance

Basel Alomair
Network Security Lab (NSL)
University of Washington
alomair@uw.edu

Abstract. In this work, we look at authenticated encryption schemes from a new perspective. As opposed to focusing solely on the *"security"* implications of the different methods for constructing authenticated encryption schemes, we investigate the effect of the method used to construct an authenticated encryption scheme on the *"performance"* of the construction. We show that, as opposed to the current NIST standard, by performing the authentication operation before the encryption operation, the computational efficiency of the construction can be increased, without affecting the security of the overall construction. In fact, we show that the proposed construction is even more secure than standard authentication based on universal hashing in the sense that the hashing key is resilient to key recovery attacks.

1 Introduction

There are three different methods to generically compose an authenticated encryption scheme by combining an encryption algorithm with a message authentication code (MAC) algorithm: Encrypt-and-MAC ($E\&M$), Encrypt-then-MAC (EtM), or MAC-then-Encrypt (MtE). Although significant efforts have been devoted to analyzing the security implications of different generic compositions (see, e.g., [11, 12, 22, 27, 49, 52]), little effort has been devoted to the study of the performance implications of different generic compositions [7]. Of particular interest to this work is the performance aspect of generic compositions when the encryption algorithm is blockcipher based and the MAC algorithm is universal hash-function family based. (We focus on such constructions since blockciphers are the building blocks of choice for constructing secure encryption algorithms [44] and since universal hash families based MACs are the fastest method for message authentication [66]).

In a typical EtM composition, the plaintext is broken into blocks. Each block is processed with a blockcipher, resulting in a ciphertext block. The resulting ciphertext blocks are then authenticated using a MAC based on a universal hash-function family (in the Carter-Wegman style [23, 24]). One of the most recent authenticated encryption schemes is the current National Institute of Standards and Technology (NIST) standardized Galois/Counter Mode (GCM) of authenticated encryption [30]. The GCM standard is based on the Carter-Wegman Counter (CWC) blockcipher mode of authenticated encryption proposed by Kohno et al. in [46]. The GCM and CWC modes of operations give high-performance authenticated encryption by combining the counter mode of encryption with a universal hash-function family for authentication.

Recently, there has been increasing attention in the design of secure and efficient authenticated encryption algorithms. Following a long tradition of focused competitions in symmetric cryptography,[1] a new NIST-funded competition, CAESAR, is now calling for standardized authenticated ciphers. The AES competition is generally viewed as having provided a tremendous boost to the cryptographic research community's understanding of blockciphers, and a tremendous increase in confidence in the security of some blockciphers. Similar comments apply to eSTREAM and to the SHA-3 competition, and are also expected to apply to CAESAR [16].

The OKH solution. In this work, we investigate the performance implications of the order in which the two

[1] The 1997 NIST's open competition for a new Advanced Encryption Standard (AES), the 2004 ECRYPT's open competition for Stream Ciphers (eSTREAM), and the 2007 NIST's open competition for a new Hash Standard (SHA-3).

operations, encryption and authentication, are performed. We describe the Odd Key Hashing (OKH) mode of authenticated encryption. The OKH mode is motivated by the current NIST standard (the Galois/Counter Mode (GCM) of authenticated encryption [30]) and the CWC mode of authenticated encryption proposed by Kohno et al. [46]. However, unlike the GCM and the CWC schemes, the order of encrypt-then-authenticate is reversed in the OKH mode. That is, as opposed to applying the hashing operation on the ciphertext, it is applied on the plaintext, before blockcipher encryption.

Earlier results in the security of authenticated encryption systems show that, when combining a secure encryption algorithm[2] and a secure MAC algorithm[3], only the EtM composition guarantees the construction of a secure authenticated encryption system [12, 22, 49]. Such early results are perhaps the main reason for adopting the EtM composition as a standard. However, more recent results show that the MtE and the $E\&M$ can indeed construct secure authenticated encryption systems [7, 52]. The main objective of this work is to highlight the possible performance advantages of the MtE and the $E\&M$ compositions, hoping to pave the road for more efficient standards (especially with the ongoing NIST-funded competition to standardize authenticated encryption algorithms [16]).

The main result of this study is to show that, while the hash family used to construct a MAC in the EtM composition *must be universal*, this need not be the case in the MtE composition.[4] The performance implication of this result is that, since the hash family need not be universal, it can be computed faster than the fastest universal hash family in the cryptographic literature. The theoretical significance of this result is that relaxing the security requirements on the MAC algorithm does not affect the provable security of the overall authenticated encryption composition. In fact, we show that MtE compositions are even more secure than their EtM counterparts in the sense that they are secure against the key-recovery attacks discovered in [39]. We emphasize that the main purpose of this work is not the design of most efficient authenticated encryption algorithm. Rather, laying down the theoretical basis for the design of more efficient authenticated encryption algorithms is the main purpose of this work.[5]

Versions. An extended abstract of this paper appeared in the 10th International Conference on Applied Cryptography and Network Security–ACNS'12 [2]; this is the full version. We note here that the preliminary version of OKH appeared in [2] has a vulnerability when the integer representation of the last blocks in the forgery attempt are equal to 2^{n-1}. This vulnerability has appeared in [67] and it has been fixed in this manuscript.

Organization. In Section 2, we give a brief background and discuss related work. In Section 3, we give some preliminaries. In Section 4, we describe the OK hash family. In Section 5, we describe the construction of the proposed OKH authenticated encryption scheme. In Section 6, we state and prove the authenticity and privacy theorems of the proposed scheme. In Section 7, we summarize the basic ideas behind the proposed mode of operation and provide performance discussions. In Section 8, we conclude the paper.

2 Background and related work

There are two main approaches to design an authenticated encryption system: a generic approach and a dedicated approach. In the generic approach, an encryption primitive for data privacy is combined with a MAC primitive for data integrity to construct an authenticated encryption system. Examples of the generic constructions include, but are not limited to, SSH [72], IPsec [28], SSL [33], CWC [46], and GCM [30]. In the

[2] Secure in the sense that it provides indistinguishability under chosen plaintext attacks

[3] Secure in the sense that it provides unforgeability under chosen message attacks

[4] Although the same result can be shown for the $E\&M$ composition, we restrict our discussion to the MtE composition for brevity.

[5] It might be worth mentioning here that the ideas presented in this work are the basis for the design of AVALANCHE [3], one of the candidates for the CAESAR competition.

dedicated approach, an authenticated encryption primitive is designed as a standalone system. The first dedicated scheme is the PCBC of Gligor and Donescu [34]. The formal notion of authenticated encryption systems was introduced independently by Katz and Yung in [45], and by Bellare and Rogaway in [13]. Since then, many dedicated authenticated encryption schemes have been proposed, such as, RPC of Katz and Yung [45], XECB of Gligor and Donescu [35], IAPM of Jutla [43], OCB of Rogaway et al. [60], and EAX of Bellare et al. [14]. A noteworthy point is that all secure dedicated authenticated encryption primitives are blockcipher based. That is, although stream cipher based authenticated encryption primitives have been proposed, e.g., in [32,69], such stream cipher based proposals have been analyzed and shown to be vulnerable to differential cryptanalysis [54,56,57,70].

The use of universal hash-function families to construct MAC algorithms is due to Carter and Wegman [23, 24]. Compared to blockcipher based MACs, such as [9, 18, 29, 41], and cryptographic hash function based MACs, such as [8, 21, 58, 65], universal hashing based MACs lead to faster message authentication [17, 38, 47, 59]. The security of MACs based on universal hashing has been extensively studied (see, e.g., [4,5,39]

The speed of a universal hash family based MAC relies mainly on the speed of the used universal hash family. Consequently, significant efforts have been devoted to the design of fast universal hash families. In [47], Krawczyk introduced the cryptographic CRC which hashes in about 6 cycles/byte, as shown by Shoup in [62]. In [59], Rogaway proposed the bucket hashing which was the first hash family explicitly targeted for fast software implementation; it runs in about $1.5 - 2.5$ cycles/byte [17]. In [42], Johansson described bucket hashing with smaller key size. In [38], Halevi and Krawczyk proposed the MMH family, which hashes at about $1.2 - 3$ cycles/byte. In [31], Etzel et al. proposed the square hash, an MMH-variant that can be more efficient than MMH in certain settings [17]. In [15], Bernstein proposed floating-point arithmetic based hash function that achieves a peak speed of 2.4 cycles/byte. In [1], Afanassiev et al. described an application of hashing based on polynomial evaluation over finite fields. In [55], Nevelsteen and Preneel study the performance of several universal hash functions proposed for MACs. The speed champion of universal hash functions directed for software implementation is the NH family of Black et al. proposed in [17]. The NH family is an extension to the MMH family of [38]. The speed improvement comes from eliminating the non-trivial modular reduction required by the MMH family. The novelty of NH family is that it uses arithmetic modulo powers of two or, as the authors call it, "computations that computers like to do [17]." The NH family hashes at about 0.34 cycles/byte for 2^{-32} probability of message collision.

Recently, Alomair and Poovendran showed that, in contrast to the EtM composition, one can utilize the $E\&M$ and MtE compositions to increase the efficiency of the system by eliminating the last blockcipher call [6, 7]. In this work, we take the idea of utilizing the $E\&M$ and MtE compositions one step further. That is, while the use of universal hash functions was necessary for the security of the schemes in [6, 7], the scheme proposed here need not use universal hash functions, thus increasing the speed of the system, while maintaining the required provable security.

3 Notations and Preliminaries

In this section we state our assumptions and describe the notations and definitions that will be used for the rest of the paper.

3.1 Notations

The following notations will be used throughout the rest of the paper.

- If s is a binary string, $|s|$ denotes the length of s in bits.
- For a positive integer β, $\{0,1\}^\beta$ denotes a binary string of length β-bits, and $\{0,1\}^*$ denotes a binary string of arbitrary length.

- If b is a bit and β is a positive integer, we denote by b^{β} the concatenation of b with itself β times.
- For a non-empty set $\mathcal{H}$, we denote by $h \overset{\$}{\leftarrow} \mathcal{H}$ the selection of a member of $\mathcal{H}$ uniformly at random and assigning it to h.
- If x and n are positive integers so that $0 \leq x < 2^n$, we denote by $\mathsf{tostr}(x, n)$ the binary representation of x as an n-bit string (in a big-endian format).
- If s is a binary string, we denote by $\mathsf{toint}(s)$ the unsigned integer representation of s (in a big-endian format).
- If s is a binary string and ℓ is a positive integer, we denote by $\mathsf{setlen}(s, \ell)$ the truncation of s into its ℓ most significant bits. If $|s| < \ell$, then $\mathsf{setlen}(s, \ell)$ denotes the ℓ-bit long string $s||0^{\ell-|s|}$.

3.2 Universal Hash-Function Families

A family of hash functions $\mathcal{H}$ is specified by a finite set of keys $\mathcal{K}$. Each key $k \in \mathcal{K}$ defines a member of the family $\mathcal{H}_k \in \mathcal{H}$. As opposed to thinking of $\mathcal{H}$ as a set of functions from $\mathcal{D}$ to $\mathcal{R}$, it can be viewed as a single function $\mathcal{H} : \mathcal{K} \times \mathcal{D} \to \mathcal{R}$, whose first argument is usually written as a subscript. A random element $h \overset{\$}{\leftarrow} \mathcal{H}$ is determined by selecting a $k \overset{\$}{\leftarrow} \mathcal{K}$ uniformly at random and setting $h \leftarrow \mathcal{H}_k$. There are many classes of universal hash families, depending on their probability of message collision (see, e.g., [23,38,47,48,63,68]). We give below a formal definition of one class of universal hash families called ϵ-almost universal.

Definition 1. *Let $\mathcal{H} = \{h : \mathcal{D} \to \mathcal{R}\}$ be a family of hash functions and let $\epsilon \geq 0$ be a real number. $\mathcal{H}$ is said to be ϵ-almost universal, denoted ϵ-AU, if for all distinct $M, M' \in \mathcal{D}$, we have that $\Pr_{h \leftarrow \mathcal{H}}\Big[h(M) = h(M')\Big] \leq \epsilon$. $\mathcal{H}$ is said to be ϵ-almost universal on equal-length strings if for all distinct, equal-length strings $M, M' \in \mathcal{D}$, we have that $\Pr_{h \leftarrow \mathcal{H}}\Big[h(M) = h(M')\Big] \leq \epsilon$.*

3.3 Blockciphers

Let $F : \mathcal{K} \times \mathcal{D} \to \mathcal{R}$ be a family of functions from $\mathcal{D}$ to $\mathcal{R}$ indexed by keys $\mathcal{K}$. We use $F_K(D)$ as shorthand for $F(K, D)$. F is a family of permutations (i.e. a blockcipher), if $\mathcal{D} = \mathcal{R}$ and $F_K(\cdot)$ is a permutation on $\mathcal{D}$ for each $K \in \mathcal{K}$. If F is a family of permutations, we use $F_K^{-1}(\cdot)$ to denote the inverse of $F_K(\cdot)$ and we use $F^{-1}(\cdot, \cdot)$ to denote the function that takes as input (K, D) and computes $F_K^{-1}(D)$.

We adopt the notion of security for blockciphers formalized in [10]. Let $\mathsf{Perm}(\mathcal{K}, \mathcal{D})$ denote the set of all possible blockciphers (permutations) with key space $\mathcal{K}$ and domain $\mathcal{D}$. Then, the notation $\pi \overset{\$}{\leftarrow} \mathsf{Perm}(\mathcal{K}, \mathcal{D})$ corresponds to selecting a random block-cipher. Given a family of functions $E : \mathcal{K} \times \mathcal{D} \to \mathcal{D}$ and a key $K \in \mathcal{K}$, define the related-key oracle $E_{RK(\cdot,K)}(\cdot)$ as an oracle that takes two arguments, a function $\phi : \mathcal{K} \to \mathcal{K}$ and an element $M \in \mathcal{D}$, and that returns $E_{\phi(K)}(M)$.

The function ϕ is the related-key-deriving (RKD) function or the key transformation function. Let Φ be a set of functions mapping $\mathcal{K}$ to $\mathcal{K}$. Then Φ is called the set of allowed RKD functions, or allowed key-transformations.

Let $E : \mathcal{K} \times \mathcal{D} \to \mathcal{D}$ be a family of functions and let Φ be a set of RKD functions over $\mathcal{K}$. Let $\mathcal{A}$ be an adversary with access to a related-key oracle, and restricted to queries of the form (ϕ, m) in which $\phi \in \Phi$ and $m \in \mathcal{D}$. Then,

$$\begin{aligned}\mathsf{Adv}^{\text{prp-rka}}_{\Phi,E}(\mathcal{A}) = \Pr\Big[K \overset{\$}{\leftarrow} \mathcal{K} : \mathcal{A}^{E_{RK(\cdot,K)}(\cdot)} = 1\Big] \\ - \Pr\Big[K \overset{\$}{\leftarrow} \mathcal{K}; \pi \overset{\$}{\leftarrow} \mathsf{Perm}(\mathcal{K}, \mathcal{D}) : \mathcal{A}^{\pi_{RK(\cdot,K)}(\cdot)} = 1\Big] \qquad (1)\end{aligned}$$

denotes the prp-rka-advantage of $\mathcal{A}$ in distinguishing a random instance of E from a random permutation using Φ-restricted related keys. We say that $\mathcal{E}$ is a secure pseudorandom permutation (prp) against related-key-attacks (rka) if the prp-rka-advantages of all adversaries using reasonable resources is small.

A blockcipher is said to be strong pseudorandom permutation (sprp) if it is indistinguishable from a random permutation even if the adversary is given an oracle access to the inverse function. Then,

$$\begin{aligned}\mathsf{Adv}^{\text{sprp-rka}}_{\Phi,E}(\mathcal{A}) = \Pr\left[K \xleftarrow{\$} \mathcal{K} : \mathcal{A}^{E_{RK(\cdot,K)}(\cdot), E^{-1}_{RK(\cdot,K)}(\cdot)} = 1\right] \\ - \Pr\left[K \xleftarrow{\$} \mathcal{K}; \pi \xleftarrow{\$} \mathsf{Perm}(\mathcal{K},\mathcal{D}) : \mathcal{A}^{\pi_{RK(\cdot,K)}, \pi^{-1}_{RK(\cdot,K)}} = 1\right] \qquad (2)\end{aligned}$$

denotes the sprp-rka-advantage of $\mathcal{A}$ in distinguishing a random instance of E from a random permutation using Φ-restricted related keys.

3.4 Authenticated Encryption Schemes

The authenticated encryption model that we use is similar to the one in [46, 60]. A nonce-using, symmetric authenticated encryption scheme $\mathcal{AE} = (\mathcal{K}, \mathcal{SE}, \mathcal{VD})$ consists of three algorithms: the key generation algorithm ($\mathcal{K}$), the signed encryption algorithm ($\mathcal{SE}$), and the verified decryption algorithm ($\mathcal{VD}$). $\mathcal{AE}$ is defined over some key space KeySp, some nonce space $\mathsf{NonceSp} = \{0,1\}^{\mathsf{nl}}$, for a positive integer nl, and some message space $\mathsf{MsgSp} = \{0,1\}^*$. We require that membership in MsgSp can be efficiently tested and that if M, M' are two strings such that $M \in \mathsf{MsgSp}$ and $|M| = |M'|$, then $M' \in \mathsf{MsgSp}$.

The randomized key generation algorithm $\mathcal{K}$ returns a key $K \in \mathsf{KeySp}$. The deterministic signed encryption algorithm $\mathcal{SE}$ takes as input a key $K \in \mathsf{KeySp}$, a nonce $N \in \mathsf{NonceSp}$, and a payload message $M \in \mathsf{MsgSp}$, and returns a ciphertext $\sigma \in \{0,1\}^*$. The deterministic verified decryption algorithm $\mathcal{VD}$ takes as input a key $K \in \mathsf{KeySp}$, a nonce $N \in \mathsf{NonceSp}$, a string $\sigma \in \{0,1\}^*$, and outputs a message $M \in \mathsf{MsgSp}$ or the special symbol $\mathsf{INVALID}$ on error. We ask for the basic validity requirement that if $\sigma = \mathcal{SE}_K(N, M)$ then it must be the case that $\mathcal{VD}_K(N, \sigma) = M$.

3.5 Adversarial Model

We adopt the standard adversarial model used in authenticated encryption schemes. The adversary is given oracle access to the signed encryption algorithm $\mathcal{SE}_K(N, M)$. The adversary can call the $\mathcal{SE}$ oracle on nonce-message pairs (N, M) of her choice and observing the outputs. After calling the $\mathcal{SE}$ oracle for q times, the adversary attempts a forgery by calling the verified decryption algorithm $\mathcal{VD}_K(N, \sigma)$ for an (N, σ) pair of her choice. Note that the adversary does not see the secret key K. If the verified decryption oracle returns the $\mathsf{INVALID}$ symbol, the adversary is considered unsuccessful; otherwise, the forgery attempt is said to be successful.

A standard assumption in authenticated encryption schemes is that the adversary is nonce-respecting. An adversary is said to be nonce-respecting if she never repeats a nonce. That is, after calling the signed encryption oracle on (N, M), the adversary never asks its oracle a query (N, M'), regardless of the oracle responses. We emphasize, however, that the nonce used in the forgery attempt may coincide with a nonce used in one of the adversary's queries.

To model the privacy of the authenticated encryption scheme, consider an adversary $\mathcal{A}$ who has one of two types of oracles: a real encryption oracle and a fake encryption oracle. The real encryption oracle $\mathcal{E}_K(\cdot,\cdot)$ takes as input a pair (N, M) and returns a ciphertext $C \leftarrow \mathcal{E}_K(N, M)$. Assume that the length of the ciphertext depends only on the length of the plaintext, that is, $|C| = l(|M|)$. The fake encryption oracle, $\$(\cdot,\cdot)$, takes as input a pair (N, M) and returns a random string $C \xleftarrow{\$} \{0,1\}^{l(|M|)}$. Given adversary $\mathcal{A}$ and authenticated encryption scheme OKH $= (\mathcal{K}, \mathcal{SE}, \mathcal{VD})$, define

$$\mathsf{Adv}^{\mathrm{priv}}_{\mathrm{OKH[BC,OK]}}(\mathcal{A}) = \Pr\left[K \xleftarrow{\$} \mathcal{K} : \mathcal{A}^{\mathcal{SE}_K(\cdot,\cdot)} = 1\right] - \Pr\left[\mathcal{A}^{\$(\cdot,\cdot)} = 1\right] \tag{3}$$

to be $\mathcal{A}$'s advantage of breaking the privacy of the authenticated encryption scheme using BC as a blockcipher and OK for hashing.

To analyze the integrity of the proposed system, fix an authenticated encryption scheme OKH $= (\mathcal{K}, \mathcal{SE}, \mathcal{VD})$ and run an adversary $\mathcal{A}$ with an oracle $\mathcal{SE}_K(\cdot,\cdot)$ for some key K. We adopt the standard definition of existential unforgeability under chosen message attack, modeled by the following game

Game 1 (EU-CMA game)

1. *A random string, K, is selected as the shared secret.*
2. *Suppose $\mathcal{A}$ makes a signed encryption query on a nonce-message pair (N, M). Then the oracle computes the integrity aware ciphertext $\sigma = \mathcal{SE}_K(N, M)$ and returns it to $\mathcal{A}$.*
3. *After calling the signed encryption oracle a polynomial number of times, $\mathcal{A}$ makes a verified decryption query (N, σ). The oracle computes the decision $d = \mathcal{VD}_K(N, \sigma)$ and returns it to $\mathcal{A}$.*

The adversary, $\mathcal{A}$, successfully forges if $\mathcal{A}$ is nonce-respecting, $\mathcal{A}$ outputs a pair (N, σ) where $\mathcal{VD}_K(N, \sigma) \neq$ $\mathsf{INVALID}$, and $\mathcal{A}$ made no earlier query (N, M) which resulted in the response σ. Let

$$\mathsf{Adv}^{\mathrm{auth}}_{\mathrm{OKH[BC,OK]}}(\mathcal{A}) = \Pr\left[K \xleftarrow{\$} \mathcal{K} : \mathcal{A}^{\mathcal{SE}_K(\cdot,\cdot)} \text{ forges }\right] \tag{4}$$

be $\mathcal{A}$'s advantage of successful forgery against the scheme OKH that uses BC as a blockcipher for encryption and the OK family for hashing when given oracle access to the $\mathcal{SE}_K(\cdot,\cdot)$ algorithm.

3.6 Properties of Odd Integers

We state here two lemmas about odd integers in the finite integer ring $\mathbb{Z}_{2^n}$ that will be used for the remainder of the paper.

Lemma 1. *For any nonzero integers α and β in $\mathbb{Z}_{2^n}$, 2^n divides $\alpha\beta$ only if both α and β are even integers. Formally, the following one-way implication must hold:*

$$\alpha\beta \equiv 0 \quad \Rightarrow \quad \alpha \equiv \beta \equiv 0 \mod 2. \tag{5}$$

Proof. Observe that for any odd integer d, $\gcd(d, 2^n) = 1$; and for any integer d such that $\gcd(d, 2^n) = 1$, d must be odd. Therefore, for any integer $d \in \mathbb{Z}_{2^n}$, $d \in \mathbb{Z}^*_{2^n}$ if and only if d is odd. Now, let α and β be nonzero elements and let

$$\alpha\beta \equiv 0 \tag{6}$$

but assume, without loss of generality, that α is an odd integer. Then, the multiplicative inverse of α in the ring $\mathbb{Z}_{2^n}$ does exist. Multiplying both sides of equation (6) by α^{-1} implies that β is the zero element in $\mathbb{Z}_{2^n}$, and the lemma follows by contradiction.

Lemma 2. *Let $\boldsymbol{X}$ be the random variable representing the experiment of drawing a number x from the set of integers $\{0, 1, 2, \cdots, 2^n - 1\}$ uniformly at random. Then, for any odd integer $k \in \mathbb{Z}_{2^n}$, the random variable $\boldsymbol{Y} = k \cdot \boldsymbol{X} \mod 2^n$ is uniformly distributed over the set $\{0, 1, 2, \cdots, 2^n - 1\}$.*

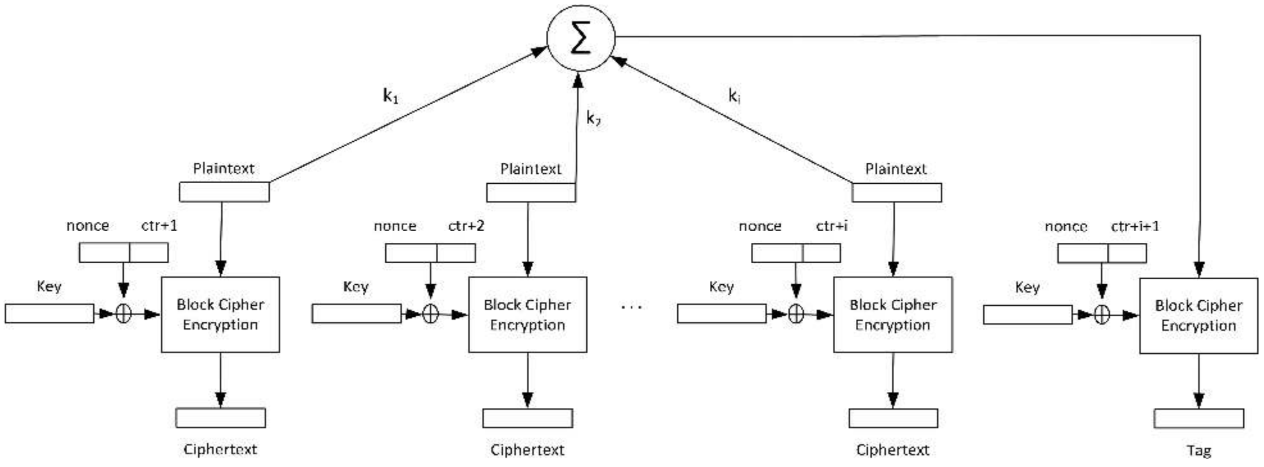

Fig. 1. A block diagram depicting the proposed OKH authenticated encryption. Plaintext blocks are multiplied by the keys of the OK hash family, the k_i's; the resulting products are summed modulo 2^n; and the result is processed with the block cipher to produce the tag. Note that the nonce-counter concatenation is XORed with the key, not the plaintext block. Therefore, given a nonce-respecting adversary, the encryption key of each block is different than all other blocks of the same message and different than all other blocks of different messages due to the use of nonce-counter concatenation. This observation is critical for the security of the proposed scheme.

Proof. This is a direct consequence of the fact that every odd integer is invertible in $\mathbb{Z}_{2^n}$. To formally prove the lemma, it suffices to show that for every $y \in \{0, 1, 2, \cdots, 2^n - 1\}$, there exists an $x \in \{0, 1, 2, \cdots, 2^n - 1\}$ that satisfies the equation

$$y = k \cdot x \mod 2^n, \tag{7}$$

and that this x is unique.

Fix any $y \in \{0, 1, 2, \cdots, 2^n - 1\}$ and any $k \in \mathbb{Z}^*_{2^n}$. Since $k \in \mathbb{Z}^*_{2^n}$, by Bézout's lemma [64], there exists $k^{-1} \in \mathbb{Z}^*_{2^n}$ so that $k^{-1}k \equiv 1 \mod 2^n$, and multiplying both sides by y gives $(yk^{-1})k \equiv y \mod 2^n$. Hence, $x = yk^{-1} \mod 2^n$ satisfies equation (7). Therefore, there exists an $x \in \{0, 1, 2, \cdots, 2^n - 1\}$ that satisfies equation (7).

To show that this x is unique, let $x' \neq x$ also satisfies equation (7). Then,

$$x'k \equiv y \mod 2^n. \tag{8}$$

Multiplying both equations (7) and (8) by k^{-1} gives $x \equiv yk^{-1} \mod 2^n$, and $x' \equiv yk^{-1} \mod 2^n$. Therefore, $x \equiv x' \mod 2^n$ and, hence, the x in $\{0, 1, 2, \cdots, 2^n - 1\}$ that satisfies equation (7) for any fixed $y \in \{0, 1, 2, \cdots, 2^n - 1\}$ is unique, and the lemma follows.

Lemmas 1 and 2, along with the fact that there is a one-to-one correspondence between n-bit strings and the integer ring $\mathbb{Z}_{2^n}$, will be used to establish the results of this paper.

4 The Odd Key Hash Family

In this section, we give a description of the OK hash family that will be used in the construction of our OKH authenticated encryption. Fix an integer $n \geq 1$ (the "block size") and an integer $b \geq 1$ (the "number of blocks"). We define the family of functions $\mathsf{OK}[n, b]$ as follows. The domain is $\mathcal{D} = \{0, 1\}^n \cup \{0, 1\}^{2n} \cup \cdots \cup \{0, 1\}^{bn}$ and the range is $\mathcal{R} = \{0, 1\}^n$. Each function in $\mathsf{OK}[n, b]$ is defined by the b-tuple $K = (k_1, \cdots, k_b)$, where $k_i \in \mathbb{Z}^*_{2^n}$ for $i = 1, \cdots, b$. A random function in $\mathsf{OK}[n, b]$ is given by drawing the k_i's at random from the multiplicative group $\mathbb{Z}^*_{2^n}$. The function determined by K is written as $\mathsf{OK}_K(\cdot)$.

```
Algorithm $\mathcal{K}$
    $K_e \xleftarrow{\$} \{0,1\}^{\mathsf{kl}}$
    $K_h \leftarrow \mathsf{KeyGenOK}$
    return $K = K_e \parallel K_h$
```

```
Algorithm $\mathcal{SE}_K(N, M)$
    $\ell \leftarrow |M| \mod n$
    $M \leftarrow M||\{1||0^{(n-2-\ell) \mod n}||1\}$
    $C \longleftarrow \mathcal{E}_{K_e}(N, M)$
    $\tau \leftarrow \mathsf{MAC}_{K_h}(N, M)$
    return $\sigma = C \parallel \tau$
```

```
Algorithm $\mathcal{VD}_K(N, \sigma)$
    if $|\sigma| \leq \mathsf{tl}$ then return INVALID
    parse $\sigma$ as $C||\tau$ where $|\tau| = \mathsf{tl}$
    if $C \notin \mathsf{MsgSp}$ then return INVALID
    $M \longleftarrow \mathcal{D}_{K_e}(N, C)$
    if $\tau \neq \mathsf{MAC}_{K_h}(N, M)$ then return INVALID
    truncate the least significant $1||0^*||1$ bits of $M$
    return $M$
```

Fig. 2. Pseudocodes of the key generation ($\mathcal{K}$), signed encryption ($\mathcal{SE}$), and verified decryption ($\mathcal{VD}$) algorithms.

For an input message $M \in \mathcal{D}$, view M as a sequence of n-bit blocks, i.e., $M = m_1 \cdots m_\ell$, where $\ell \leq b$, and write each block in its unsigned integer representation in $\mathbb{Z}_{2^n}$ (in a big-endian format). Then, the compressed image of M is given by

$$\mathsf{OK}_K(M) = \sum_{i=1}^{\ell} k_i m_i \mod 2^n. \tag{9}$$

When the values of n and b are known, we will write OK instead of $\mathsf{OK}[n, b]$ to simplify the notations.

5 Description of the OKH Authenticated Encryption

As mentioned earlier, the key idea allowing for more efficient authentication over the GCM and CWC modes of operation is advancing the hashing phase to be applied on the plaintext instead of the ciphertext. The mode of operation used for encryption is similar to the counter mode (CTR) but with the the requirement that the plaintext is to be processed by the blockcipher. A block diagram depicting the proposed OKH mode of authenticated encryption is shown in Figure 1. Plaintext blocks are multiplied by the keys of the OK hash family, the k_i's; the resulting products are summed modulo 2^n; and the result is processed with the blockcipher to produce the tag. Note that the nonce-counter concatenation is XORed with the key, not the plaintext block. Therefore, given a nonce-respecting adversary, the encryption key of each block is different than all other blocks of the same message and different than all other blocks of different messages due to the use of nonce-counter concatenation.[6] This observation is critical for the security of the proposed scheme.

As in previous authenticated encryption schemes, such as IAPM [43] and OCB [60], we require the blockcipher BC to be a strong pseudorandom permutation (many blockciphers, such as AES [25], are believed to be strong pseudorandom permutations [44]). Let BC: $\{0,1\}^{\mathsf{kl}} \times \{0,1\}^{\mathsf{bs}} \rightarrow \{0,1\}^{\mathsf{bs}}$ be the used blockcipher, where kl and bs are the key length and block size of the blockcipher, respectively. The authenticated encryption using BC for encryption and the OK family for hashing, OKH $= (\mathcal{K}, \mathcal{SE}, \mathcal{VD})$, is defined as follows. The message spaces are

$$\mathsf{MsgSp} = \Big\{ M \in \{0,1\}^* : |M| \leq \mathsf{MaxLen} \Big\}, \tag{10}$$

$$\mathsf{KeySp} = \Big\{ (K_e, K_h) \in \{0,1\}^{\mathsf{kl}} \times \{0,1\}^{\mathsf{MaxLen}} \Big\}, \tag{11}$$

$$\mathsf{NonceSp} = \Big\{ N \in \{0,1\}^* : |N| \leq \mathsf{kl} - \log_2 \mathsf{MaxLen} \Big\}, \tag{12}$$

where MaxLen is the message maximum length. The size of the counter, CtrLen, in the mode of operation of Figure 1 is at least $\log_2 \mathsf{MaxLen}$. The concatenation of the nonce and the counter is of length kl-bits.

[6] A well-known fact is that rekeying can hurt the performance of AES-like blockciphers. Another alternative is to replace rekeying with a tweakable blockcipher [51]. Since the construction of particular authenticated encryption algorithm is not the main focus of this work, however, we focus only on one of the two alternatives and emphasize that they both provide the required security.

```
Algorithm KeyGenOK
  Key $\xleftarrow{\$} \{0,1\}^{\mathsf{MaxLen}}$
  $\alpha \leftarrow |\mathsf{Key}|/\mathsf{tl}$
  break Key into tl-bit chunks $K_i$'s
  for $i = 1$ to $\alpha$ do
    $K_i \leftarrow K_i \vee \mathsf{tostr}(1, \mathsf{tl})$
  return $K_h = K_1 || \cdots || K_\alpha$
```

```
Algorithm $\mathcal{E}_{K_e}(N, M)$
  $\ell \leftarrow$ min int so that bs divides $|M||0^\ell|$
  $M \leftarrow M||0^\ell$
  $\alpha \leftarrow |M|/\mathsf{bs}$
  break $M$ into bs-bit chunks $M_i$'s
  for $i = 1$ to $\alpha$ do
    $K_{e_i} \leftarrow K_e \oplus (N||\mathsf{tostr}(i, \mathsf{CtrLen}))$
    $c_i \leftarrow \mathsf{BC}_{K_{e_i}}(M_i)$
  return $C = c_1 || \cdots || c_\alpha$
```

```
Algorithm $\mathcal{D}_{K_e}(N, C)$
  $\alpha \leftarrow |C|/\mathsf{bs}$
  break $C$ into bs-bit chunks $c_i$'s
  for $i = 1$ to $\alpha$ do
    $K_{e_i} \leftarrow K_e \oplus (N||\mathsf{tostr}(i, \mathsf{CtrLen}))$
    $M_i \leftarrow \mathsf{BC}^{-1}_{K_{e_i}}(c_i)$
  return $M = M_1 \, || \cdots || M_\alpha$
```

```
Algorithm $\mathsf{MAC}_{K_h}(N, M)$
  $\gamma' \leftarrow \mathsf{OK\text{-}HASH}_{K_h}(M)$
  $\ell \leftarrow \mathsf{bs} - n$
  $\gamma \leftarrow \gamma'||0^\ell$
  $\alpha \leftarrow |M|/n$
  $K_\alpha \leftarrow K_e \oplus (N||\mathsf{tostr}(\alpha + 1, \mathsf{CtrLen}))$
  $\tau' \leftarrow \mathsf{BC}_{K_\alpha}(\gamma)$
  $\tau \leftarrow \mathsf{setlen}(\tau', \mathsf{tl})$
  return $\tau$
```

```
Algorithm $\mathsf{OK\text{-}HASH}_{K_h}(M)$
  $\alpha \leftarrow |M|/n$
  break $M$ into $n$-bit chunks $M_i$'s
  for $i = 1$ to $\alpha$ do
    $m_i \leftarrow \mathsf{toint}(M_i)$
    $k_i \leftarrow \mathsf{toint}(K_i)$
  $\gamma \leftarrow \sum_{i=1}^{\alpha} k_i m_i \mod 2^n$
  return $\gamma$
```

Fig. 3. Pseudocodes of the KeyGenOK, $\mathcal{E}$, $\mathcal{D}$, MAC, and OK-HASH algorithms.

First, the plaintext message is appended with $1||0^*||1$ as its least significant bits. As can be seen in Algorithm $\mathcal{SE}_K(\cdot, \cdot)$ in Figure 2, the number of zeros will depend on the length of the message and is chosen so that the message length is divisible by n and the least significant bit of the last message block is '1'. Informally speaking, the authentication tag is computed by dividing the plaintext message into blocks of n-bit long, hash it according to equation (9) using a member of the OK family, and encrypt the resulting n-bit hashed image (as shown in the MAC algorithm of Figure 3). The size of the hashing images, n, is less than or equal to the blockcipher size, bs. The encryption part is done the natural way (as shown in Figure 3).

Remark 1. There are two important points to note about the OK family. First, the OK family is defined over the domain $\mathcal{D}$ only. This issue, however, can be easily solved with an appropriate padding. Second, and more important, as in universal hash families, the OK family as described in Section 4 can only be used to authenticate equal-length messages. For example, a message block consisting of all zeros will not contribute to the value of the hashed image. Hence, it is easy to come up with two distinct messages that collide and, eventually, achieve a successful forgery. However, there are known techniques to make the hash function applicable to arbitrary-length messages. For instance, in [17] the authors proposed appending the length of the message at the end. In our case, it suffices to append the sequence '$1||0^*||1$' as an end-of-message (EOM) sequence (both the most significant '1' and the least significant '1' of EOM will play pivotal roles in the security of the algorithm, as will be detailed in the proof of Theorem 2).

Another important remark is related to the message length. For a message that is longer than MaxLen, it is treated as multiple chunks of length MaxLen or less and the corresponding tags are concatenated. That is, arbitrary long messages can be authenticated using the same fixed-length hashing key (this is actually the case for any universal hashing based MAC, not just the proposed one [17]). Typically, MaxLen is set to some proper value that is not too short for maximum message length and not too long for the key to be infeasible. Note, however, that this is not a major design challenge as one is often dealing with reasonably short messages. For instance, about one-third of the messages on the backbone of the Internet are only 43 bytes [60]. For the rest of the paper, we will assume messages of length MaxLen or less for simplicity.

Formally, The OKH's key generation ($\mathcal{K}$), signed encryption ($\mathcal{SE}$), and verified decryption ($\mathcal{VD}$) algorithms are as defined in Figure 2.

The rest of the algorithms (KeyGenOK, $\mathcal{E}$, $\mathcal{D}$, MAC, OK-HASH) are defined in Figure 3. Algorithm KeyGenOK handles the generation of the key that defines the used member of the OK hashing family. Algorithms $\mathcal{E}$ and $\mathcal{D}$ handle the encryption and decryption operations. Algorithm MAC handles the generation of authentication tags, which calls algorithm OK-HASH to compress the message.

6 Theorem Statements and Proofs

6.1 Security of Encryption

In this section, we show that the privacy of the proposed scheme is provably secure assuming the used blockcipher is a pseudorandom permutation secure against related-key attacks.

Theorem 1. *Let* OKH[BC;OK] *be the authenticated encryption scheme described in Section 5 using the* OK *hash family for compression and the blockcipher* BC *for encryption. Then given a nonce-respecting adversary, $\mathcal{A}$, against* OKH[BC;OK], *one can construct an adversary $\mathcal{B}$ against* BC *such that*

$$\mathsf{Adv}^{\text{priv}}_{\mathsf{OKH[BC;OK]}}(\mathcal{A}) \leq \mathsf{Adv}^{\text{prp-rka}}_{\oplus,\mathsf{BC}}(\mathcal{B}), \tag{13}$$

where $\mathsf{Adv}^{\text{priv}}_{\mathsf{OKH[BC;OK]}}(\mathcal{A})$ *is as defined in equation (3) and* $\mathsf{Adv}^{\text{prp-rka}}_{\oplus,\mathsf{BC}}(\mathcal{B})$ *denotes the adversary's advantage of breaking the prp-rka-security of the blockcipher when given access to oracle with keys related by the xor operation. Furthermore, the experiment for $\mathcal{B}$ takes the same time as the experiment for $\mathcal{A}$ and, if $\mathcal{A}$ makes at most q oracle queries totaling at most μ bits of payload data, then $\mathcal{B}$ makes at most $\mu/\ell + q$ oracle queries.*

Theorem 1 states that, if BC is a pseudorandom permutation secure against related-key attacks, then the proposed authenticated encryption scheme provides data privacy.

Proof (of Theorem 1). Let $\mathcal{B}$ be an adversary against BC that uses adversary $\mathcal{A}$ and that has oracle access to the blockcipher. Adversary $\mathcal{B}$ runs $\mathcal{A}$ and replies to $\mathcal{A}$'s encryption oracle queries using its own oracle $\mathsf{BC}_{RK(\cdot,K)}(\cdot)$ for the blockcipher used in the construction of the authenticated encryption scheme describe in Section 5. Adversary $\mathcal{B}$ returns the same bit that $\mathcal{A}$ returns. Then,

$$\Pr\left[K \xleftarrow{\$} \mathcal{K} : \mathcal{A}^{\mathcal{SE}_K(\cdot,\cdot)} = 1\right] = \Pr\left[K \xleftarrow{\$} \mathcal{K} : \mathcal{B}^{\mathsf{BC}_{RK(\cdot,K)}(\cdot)} = 1\right], \tag{14}$$

since each block in the mode of operation of Figure 1 is encrypted with a different key. Furthermore,

$$\Pr\left[\mathcal{A}^{\$(\cdot,\cdot)} = 1\right] = \Pr\left[K \xleftarrow{\$} \mathcal{K}; \pi \xleftarrow{\$} \mathsf{Perm}(\mathcal{K},\mathcal{D}) : \mathcal{B}^{\pi_{RK(\cdot,K)}(\cdot)} = 1\right] \tag{15}$$

since $\mathcal{B}$ replies to all of $\mathcal{A}$'s oracle queries with independently selected random strings. Consequently,

$$\begin{aligned}
\mathsf{Adv}^{\text{priv}}_{\mathsf{OKH[BC;OK]}}(\mathcal{A}) &= \Pr\left[K \xleftarrow{\$} \mathcal{K} : \mathcal{A}^{\mathcal{SE}_K(\cdot,\cdot)} = 1\right] - \Pr\left[\mathcal{A}^{\$(\cdot,\cdot)} = 1\right] \\
&= \Pr\left[K \xleftarrow{\$} \mathcal{K} : \mathcal{B}^{\mathsf{BC}_{RK(\cdot,K)}(\cdot)} = 1\right] \\
&\qquad - \Pr\left[K \xleftarrow{\$} \mathcal{K}; \pi \xleftarrow{\$} \mathsf{Perm}(\mathcal{K},\mathcal{D}) : \mathcal{B}^{\pi_{RK(\cdot,K)}(\cdot)} = 1\right] \\
&= \mathsf{Adv}^{\text{prp-rka}}_{\oplus,\mathsf{BC}}(\mathcal{B}),
\end{aligned}$$

and the theorem follows.

Remark 2. One may prefer the privacy notion of indistinguishability under chosen plaintext attacks (IND-CPA) [36], which captures the adversary's inability to distinguish ciphertexts for a pair of adversary-selected plaintexts of equal length. However, the used privacy definition, which captures the adversary's inability to distinguish equal length ciphertexts from random strings, implies the notion of IND-CPA, but the converse is not true [60]. For the proposed OKH and a nonce-respecting adversary, the proof of IND-CPA follows directly from the fact that the blockcipher is a prp-rka-secure and that, for different nonces, each block is encrypted with a different key.

6.2 Security of Authentication

We give here information-theoretic bounds on the authenticity of the scheme of Section 5 assuming the use of a true random permutation, $\mathsf{Perm}(\ell)$, for encryption.

Theorem 2. *Fix an* $\mathsf{OK}[n, b]$ *hash family and let* $\mathsf{Perm}(\ell) : \{0,1\}^\ell \to \{0,1\}^\ell$ *be a true random permutation and let* tl *be the desired tag length. Let* $\mathcal{A}$ *be a nonce-respecting adversary that asks* q *queries and then makes its forgery attempt against the* OKH *of Section 5. Then,* $\mathcal{A}$*'s advantage of successful forgery is bounded by*

$$\mathsf{Adv}^{\text{auth}}_{\text{OKH}[\mathsf{Perm}(\ell),\mathsf{OK}_{[n,b]}]}(\mathcal{A}) \leq 2^{-n} + 2^{-\mathsf{tl}},$$

where $\mathsf{Adv}^{\text{auth}}_{\text{OKH}[\mathsf{Perm}(\ell),\mathsf{OK}_{[n,b]}]}(\mathcal{A})$ *is as defined in equation (4).*

It is standard to pass a complexity-theoretic analog of Theorem 2, but in doing this one will need access to a BC^{-1} oracle in order to verify a forgery attempt, which translates into needing the strong pseudorandom permutation assumption. One gets the following. Fix an $\mathsf{OK}[n, b]$ hash family and a blockcipher $\mathsf{BC} : \mathcal{K} \times \{0,1\}^\ell \to \{0,1\}^\ell$. Let $\mathcal{A}$ be a nonce-respecting adversary that asks q queries totaling at most λ bits of payload and then makes its forgery attempt. Then, there is an adversary $\mathcal{B}$ attacking the blockcipher in which

$$\mathsf{Adv}^{\text{auth}}_{\text{OKH}[\mathsf{BC},\mathsf{OK}]}(\mathcal{A}) \leq \mathsf{Adv}^{\text{sprp-rka}}_{\oplus,\mathsf{BC}}(\mathcal{B}) + 2^{-n} + 2^{-\mathsf{tl}}.$$

Furthermore, adversary $\mathcal{B}$ takes the same time adversary $\mathcal{A}$ takes and makes at most $\lambda/\ell + q$ oracle queries.

Before we proceed with the security analysis, we give the intuition behind the choices of the OK hash family and the mode of encryption in Figure 1. First, observe that the OK family is not a universal hash family. To see this, let the OK family be defined over the ring $\mathbb{Z}_{2^n}$. Let $M = m_1||m_2$ be a message of two n-bit blocks (for the rest of the paper, we overload m_i to denote both the n-bit binary string of the ith message block and its integer representation as an element of $\mathbb{Z}_{2^n}$ in a big-endian format; the distinction between the two representations will be omitted as long as it is clear from the context). Consider now the message $M' = m'_1||m'_2 = m_1 + 2^{n-1}||m_2 + 2^{n-1} \neq M$. Then,

$$\begin{align} k_1 m'_1 + k_2 m'_2 &= k_1\big(m_1 + 2^{n-1}\big) + k_2\big(m_2 + 2^{n-1}\big) \tag{16}\\ &\equiv k_1 m_1 + k_2 m_2 \mod 2^n, \tag{17}\end{align}$$

where equation (17) holds since both k_1 and k_2 are odd integers by design.

Remark 3. Equation (17) implies that the OK family, unlike universal hash families, cannot be used to construct standard MACs since forgers can easily find two colliding messages. However, one key idea of the proposed OKH is that finding two messages that collide does not translate into a successful forgery (since the adversary must also predict the correct ciphertext corresponding to the colliding messages). The other key idea behind the design of OKH is that the effect of modifying an observed ciphertext block will result in modifying its corresponding plaintext block randomly (assuming the blockcipher is a strong pseudorandom permutation).[7] For that, the following lemma addresses the adversary's chances of causing a collision in the hashing phase by modifying the ciphertext.

Lemma 3. *Fix an* $\mathsf{OK}[n, b]$ *hash family and let* $\mathsf{Perm}(\ell) : \{0,1\}^\ell \to \{0,1\}^\ell$ *be a true random permutation. Let* $C \neq C'$ *be any two distinct ciphertexts and let* $M \neq M'$ *be the plaintext messages corresponding to* C *and* C'*, respectively. Then, assuming the use of* $\mathsf{Perm}(\ell)$ *for block encryption,* $\Pr_{h \xleftarrow{\$} \mathsf{OK}[n,b]}\Big[h(M) = h(M')\Big] \leq 2^{-n}$.

[7] These two key ideas are the main ingredients of the formal security proof of Theorem 2.

Proof. Let c_i and c'_i denote the i^{th} blocks of C and C', respectively. Similarly, let m_i and m'_i denote the i^{th} blocks of M and M', respectively. Since $M \neq M'$, they must be different in at least one block. Assume M and M' differ in a single block only. Without loss of generality, let $m'_1 = m_1 + \epsilon \neq m_1$ and the rest of the blocks are the same. Then, since k_1 is an odd integer, by Lemma 1, $k_1 \cdot \epsilon \not\equiv 0 \mod 2^n$ and the probability $\Pr_{h \xleftarrow{\$} \mathsf{OK}[n,b]} \Big[h(M) = h(M') \Big] = 0$.

Assume now that M and M' differ by more than one block. Write $m'_i = m_i + \epsilon_i \neq m_i$ for each block i in which the two messages differ. Since $\mathsf{Perm}(\ell)$ is used for encryption, even if c'_i differs with c_i by a known constant, ϵ_i will be a random element of $\mathbb{Z}_{2^n}$. Therefore, by Lemma 2, $\Pr_{h \xleftarrow{\$} \mathsf{OK}[n,b]} \Big[h(M) = h(M') \Big] = 2^{-n}$, and the lemma follows.

Remark 4. Lemma 3 illustrates the significance of restricting the hashing keys to the set of odd integers. To see this, let the keys be drawn from $\mathbb{Z}_{2^n}$ as opposed to $\mathbb{Z}^*_{2^n}$. Assume that k_i, for some i, happened to be equal to 2^{n-1}. Then, using the fact that the used blockcipher can be modeled as a strong pseudorandom permutation, any modification of the i^{th} ciphertext block will go undetected with a probability $1/2$. This is because $\epsilon \cdot 2^{n-1}$ is congruent to zero modulo 2^n for any even ϵ. Similarly, if k_i, for some i, happened to be equal to 2^{n-2} then any modification of the i^{th} ciphertext block will go undetected with a probability $1/4$. This is because $\epsilon \cdot 2^{n-2}$ is congruent to zero modulo 2^n for any ϵ that is divisible by 4. In general, if k_i is equal to $2^{n-\ell}$, for any positive integer $\ell < n$, then any modification of the i^{th} ciphertext block will go undetected with a probability $1/2^{\ell}$.

With Remark 3 and Lemma 3 in hand, we proceed with the formal proof of Theorem 2. But before we do so, we define the two possible adversarial strategies for successful forgery.

- ForgeM: Given a plaintext message M, an adversary may attempt to produce a valid $\big(f(M), \tau\big)$ pair, where f is a function of the adversary's choice and τ is the authentication tag.
- ForgeC: An adversary may attempt to forge a valid (C, τ) pair, where C is a ciphertext of the adversary's choice and τ is the authentication tag, regardless of the plaintext corresponding to C.

ForgeM implies that the adversary is attempting to produce a valid authentication tag for a specific plaintext message that is meaningful to the adversary. ForgeC implies that the adversary is attempting to produce a valid authentication tag for some ciphertext regardless of its corresponding plaintext, even if the plaintext turns out to be gibberish. Note that an adversary may either attempt to forge messages of her choice (i.e., ForgeM), or some random messages (i.e., ForgeC). Hence, these two strategies span the entire attack vector.[8]

Proof (of Theorem 2). Assume $\mathcal{A}$ has made q queries $\Big\{ \big(N_1, M_1\big), \cdots, \big(N_q, M_q\big) \Big\}$, where $N_i \neq N_j$ for any $i \neq j$, and recorded the sequence

$$S = \Big\{ \big(N_1, M_1, C_1, \tau_1\big), \cdots, \big(N_q, M_q, C_q, \tau_q\big) \Big\}. \tag{18}$$

$\mathcal{A}$ then calls the verify oracle with $\big(N, C, \tau\big)$, where $\big(N, C, \tau\big) \neq \big(N_i, C_i, \tau_i\big)$ for any $i = 1, \cdots, q$ since otherwise $\mathcal{A}$ does not win by definition (note, however, that by the definition of the nonce-respecting adversary, N can be equal to one of the recorded N_i's). We aim here to bound the probability that $\big(N, C, \tau\big)$ will be accepted as valid.

Let M be the plaintext message corresponding to the decryption of C, i.e., the decryption of the ciphertext in the attempted forgery. We aim to bound the probabilities of the two possible strategies for forgery: 1. the adversary attempts to forge a valid nonce-ciphertext-tag tuple for a plaintext of her choice (ForgeM); or 2. the adversary attempts to authenticate a nonce-ciphertext-tag tuple regardless of their corresponding plaintext (ForgeC).

[8] In the context of Remark 3, ForgeM means the adversary is attempting to authenticate a message M that collides with one one the M_i's, while ForgeC means that the adversary is attempting to authenticate a modified version of the observed C_i's.

BOUNDING THE PROBABILITY OF ForgeM. Assume $\mathcal{A}$ attempts to falsely authenticate a message of her choice. There are two possibilities:

1. $\underline{N \neq N_i}$: let $N \neq N_i$ for any $i = 1, \cdots, q$ and recall that the authentication tag is computed as $\tau = F\big(N, \text{counter}, h(M)\big)$. Therefore, since F is a random permutation, if $N \neq N_i$ for any i, the probability of predicting a valid tag is bounded by $2^{-\mathsf{tl}}$.
2. $\underline{N = N_i}$: let $N = N_i$ for some $i \in \{1, \cdots, q\}$. If $M = M_i$ then only $(N, C, \tau) = (N_i, C_i, \tau_i)$ will be validated and the adversary does not win by definition. Now, let $M \neq M_i$ and consider the two following scenarios: $|M| = |M_i|$ and $|M| \neq |M_i|$. Assume that $|M| = |M_i|$ and let m_d be a block in which M differs than M_i (there must be at least one such block since $M \neq M_i$). Let c_d be the ciphertext block corresponding to m_d. Then, for successful forgery, the forger must predict the correct value of c_d. Recall, however, that by the nonce-respecting property, the adversary could not have called the oracle with the same nonce and m_d to record c_d. Therefore, the probability of forgery is bounded by $2^{-\ell}$, the probability of predicting a random ciphertext block.
   Now, assume that $|M| \neq |M_i|$ and consider the two following scenarios: M and M_i have the same number of blocks, or M and M_i have different number of blocks. Let M and M_i have the same number of blocks and recall that the message is appended with the '$1||0^*||1$' as an EOM sequence. In such scenario, the position of the most significant bit of the EOM sequence in M must be different than its position in M_i. Let c_d be the ciphertext block corresponding to the plaintext block of M in which the most significant '1' of the EOM resides. Then, similar to the previous scenario, the probability of successful forgery is bounded by $2^{-\ell}$, the probability of predicting the correct c_d.
   Finally, assume that M and M_i have different number of blocks. Let M_i has α number of blocks while the plaintext in the attempted forgery, M, has β number of blocks. Then, τ_i is encrypted with the key $K_e \oplus \big(N||\mathsf{tostr}(\alpha+1, \mathsf{CtrLen})\big)$ while for τ to be validated it should be the encryption of the key $K_e \oplus \big(N||\mathsf{tostr}(\beta+1, \mathsf{CtrLen})\big)$. Since $\alpha \neq \beta$, the two keys are different and, since $\tau_i = F\big(N_i, \alpha, h(M_i)\big)$ and $\tau = F\big(N, \beta, h(M)\big)$, where F is a random permutation, τ cannot be correlated to τ_i. Furthermore, recall that, by design, the least significant bit of the EOM must be the least significant bit of the last message block. Therefore, τ must be a function of k_β, which is unknown to the forger. Hence, τ cannot be correlated to any of the c_i's, the ciphertext blocks of the observed C_i. Therefore, the adversary's probability of successful forgery is bounded by $2^{-\mathsf{tl}}$.

Therefore, from the above two cases, the probability of ForgeM is bounded by

$$\max\left\{2^{-\mathsf{tl}}, 2^{-\ell}\right\} = 2^{-\mathsf{tl}},$$

since tl is, by assumption, smaller than ℓ.

BOUNDING THE PROBABILITY OF ForgeC. Let M and M_i be the plaintext messages corresponding to C and C_i, respectively. Denote by Coll the event that $h(M) = h(M_i)$ for some $i \in \{1, \cdots, q\}$; i.e., M collides with one of the recorded M_i's. Also, let $\overline{\mathsf{Coll}}$ denotes the complement of Coll.

Obviously, there are two possible scenarios here: either M will collide with one of the M_i's or it will not. If M collides with M_i for an $i \in \{1, \cdots, q\}$, then $\big(C, \tau_i\big) \neq \big(C_i, \tau_i\big)$ will pass the integrity check. However, by Lemma 3, the probability that a collision will occur by modifying the ciphertext is:

$$\Pr\Big[\mathsf{Coll}\Big] = \Pr\Big[h(M) = h(M_i)\Big] \leq 2^{-n}. \tag{19}$$

Assume now that M does not collide with any of the M_i's. If no collision has occurred, then the adversary's probability of successful forgery is bounded by the probability of predicting the plaintext message corresponding to c, or the probability of guessing the valid authentication tag corresponding to c. That is,

$$\Pr\Big[\mathsf{ForgeC}|\overline{\mathsf{Coll}}\Big] \leq \max\left\{2^{-\ell}, 2^{-\mathsf{tl}}\right\} = 2^{-\mathsf{tl}}. \tag{20}$$

By equations (19) and (20), it follows that the probability of ForgeC can be written as

$$\Pr\left[\mathsf{ForgeC}\right] = \Pr\left[\mathsf{ForgeC}|\mathsf{Coll}\right] \cdot \Pr\left[\mathsf{Coll}\right] + \Pr\left[\mathsf{ForgeC}|\overline{\mathsf{Coll}}\right] \cdot \Pr\left[\overline{\mathsf{Coll}}\right] \tag{21}$$
$$\leq \Pr\left[\mathsf{Coll}\right] + \Pr\left[\mathsf{ForgeC}|\overline{\mathsf{Coll}}\right] \tag{22}$$
$$\leq 2^{-n} + 2^{-\mathsf{tl}}. \tag{23}$$

Hence, $\mathcal{A}$'s probability of successful forgery for the verify query is at most

$$\max\Big\{\Pr[\mathsf{ForgeM}], \Pr[\mathsf{ForgeC}]\Big\} = \max\Big\{2^{-\mathsf{tl}}, 2^{-n} + 2^{-\mathsf{tl}}\Big\} = 2^{-n} + 2^{-\mathsf{tl}}, \tag{24}$$

and the theorem follows.

6.3 Security Against Key-Recovery Attack (KRA)

Recently, Handschuh and Preneel [39] showed that, compared to block cipher based, MACs based on universal hash functions have a key-recovery vulnerability. In principle, a small probability of successful forgery on authentication codes is always possible. However, the work in [39] demonstrates that, for universal hash functions based MACs, once a successful forgery is achieved, subsequent forgeries can succeed with high probabilities. The main idea in their attack is to look for a collision in the message compression phase. Once two colliding messages are known, they can be used to extract partial information about the hashing keys. Using this key information an attacker can forge valid tags for fake messages.

For example, although the OK family cannot be used to construct secure MACs in the standard setup, assume it has been used for the sake of illustrating the attack. Let an adversary call the signing oracle on $(N, M = m_1||m_2)$ to obtain its authentication tag τ. The adversary now can call the verification oracle with $(N, M_1 = m_2||m_1)$ and the same tag τ. Obviously, the verification will pass if $k_1 \equiv k_2 \mod 2^n$ (in which case $k_1m_1 + k_2m_2 \equiv k_2m_1 + k_1m_2 \mod 2^n$).

Although the verification will pass with a small probability, the adversary can continuously call the verification oracle with $(N, M_i = m_2||\alpha_i m_1)$, for different α_i's until the message is authenticated. Let $(N, M' = m_2||\alpha m_1)$ be the message that passes the verification test, for some $\alpha \in \mathbb{Z}_{2^n}$. Then, the relation $k_1 \equiv \beta k_2 \mod 2^n$, where $\beta = (\alpha m_1 - m_2)(m_1 - m_2)^{-1}$ is exposed. With this knowledge, a man in the middle can always replace the first two blocks, $m_1||m_2$, of any future message M with $\beta^{-1}m_2||\beta m_1$ without violating its tag. This is because $k_1(\beta^{-1}m_2) + k_2(\beta m_1) = k_2m_2 + k_1m_1$ regardless of values of m_1 and m_2.

Handschuh and Preneel [39] defined three classes of weak keys in universal hash functions. Each class can be exploited in a way similar to the one discussed in the above example to substantially increase the probability of successful forgery after a single collision. This attack is shared by all universal hash based MACs [39]. As per [39], the recommended mitigation to this attack is to use the less efficient block cipher based MACs, or not to reuse the same hashing key for multiple authentication.

However, MACs in MtE compositions inherit a security against key-recovery attacks (KRA) from the encryption algorithm. Let

$$\mathsf{Adv}^{\text{kra}}_{\text{OKH[BC,OK]}}(\mathcal{A}) = \Pr\left[K \xleftarrow{\$} \mathcal{K} : \mathcal{A}^{\mathcal{SE}_K(\cdot,\cdot)} \text{ recovers }\right]$$

be $\mathcal{A}$'s advantage of successfully recovering the hashing keys of OK family when BC is the block cipher used to construct OKH, one gets the following.

Theorem 3. *Fix an* OK *hash family. Let* BC *be an sprp-rka-secure block cipher used with* OK *to construct the* OKH *of Section 5. Let* $\mathcal{A}$ *be an adversary launching key-recovery attacks against the* OKH*. Then, there is an adversary* $\mathcal{B}$ *attacking the block cipher in which*

$$\mathsf{Adv}^{\text{kra}}_{\text{OKH[BC,OK]}}(\mathcal{A}) \leq \mathsf{Adv}^{\text{sprp-rka}}_{\oplus,\mathsf{BC}}(\mathcal{B}).$$

Table 1. Performance comparison of the MMH hash family of Halevi and Krawczyk [38], the polynomial-evaluation (POLY) hash family of Bernstein [15], the NH hash family of Black et al. [17], and the proposed OK family.

	MMH family	POLY family	NH family	OK family
Collision probability	2^{-30}	2^{-96}	2^{-64}	2^{-64}
Hashed image (bits)	32	128	128	64
Speed (cycles/byte)	1.2	2.4	0.87	0.27

Proof. Recall that, while the tag is a function of the plaintext, the verify oracle takes a ciphertext-tag pair as an input, as opposed to message-tag pair in standard MACs. Let (N, C, τ) be a valid nonce-ciphertext-tag tuple and assume the adversary calls the verify oracle with (N, C_i, τ), for different C_i's, until the verification is granted. By the sprp-rka-security of the underlying block cipher, the ciphertext cannot be correlated to the plaintext by computationally bounded adversaries. Therefore, even though the plaintexts M and M_i corresponding to C and C_i, respectively, caused a collision, no information about M_i is revealed to the computationally bounded adversary. Therefore, unless the adversary can break the sprp-rka-security of the used block cipher to infer information about M_i, no information about the OK hashing key can be inferred, and the theorem follows.

7 Design and Performance Discussions

In this section, we discuss the main design ideas behind the proposed OKH scheme and compare its performance to other authentication codes in the cryptographic literature. The comparison is not meant to show that the proposed OKH is the fastest in the literature; its main purpose is to show that the ideas presented here can be a promising direction for potential designs. In fact, as mentioned earlier, the basic ideas presented here are the basis for the design of AVALANCHE [3], one of the candidates still standing in the onging competition for authenticated ciphers.

7.1 Design Rationale

First, note that there is a one-to-one correspondence between the set of n-bit sequences and the integer ring $\mathbb{Z}_{2^n}$. Hence, the integer ring $\mathbb{Z}_{2^n}$ is the natural choice when performing arithmetic on binary sequences. From a computational-efficiency point of view, the integer ring $\mathbb{Z}_{2^n}$ has an advantage over other finite integer rings in that modular reduction can simply be realized by truncating what is beyond the n^{th} bit (no nontrivial modular reduction is required). From a mathematical point of view, the integer ring $\mathbb{Z}_{2^n}$ possesses the unique property that an element $a \in \mathbb{Z}_{2^n}$ is invertible if and only if it is an odd integer. That is, the multiplicative group $\mathbb{Z}_{2^n}^*$ consists of all odd integers less than 2^n, and nothing else. Consequently, for a random number ϵ drawn uniformly from $\mathbb{Z}_{2^n}$ and an odd key k, the value of $\epsilon \cdot k \mod 2^n$ is uniformly distributed over $\mathbb{Z}_{2^n}$ (by Lemma 2). This fact, along with the fact that blockciphers can be realized as strong pseudorandom permutations, are the main principles behind the design of the OKH authenticated encryption composition. That is, by advancing the hashing phase to be applied to the plaintext, before blockcipher encryption, and restricting the hashing keys to the set of odd integers, one can show that a successful forgery by causing a collision in the hashing phase can only occur with a negligible probability (by Theorem 2).

In the literature, without advancing the hashing phase to be performed before blockcipher encryption, the objective of guaranteeing that a forgery attempt by causing a collision in the hashing phase can succeed with a negligible probability has been achieved by restricting the hash function to be universal. Intuitively, removing such a restriction on the hash function should only increase its speed. In Table 1, we compare the speed of the proposed OK family with the speeds of prominent universal hash families.

7.2 Comparison with Universal Hash Functions

We give here a detailed performance comparison between the OK family and the NH family of Black et al. [17], the fastest universal hash family in the literature directed for software implementations. Comparison with other known hash families is summarized in Table 1. As before, let M be a message to be authenticated and write M as a sequence of n-bit strings; i.e., $M = m_1||\cdots||m_\ell$, where $|m_i| = n$. Similarly, let the key $K = k_1||\cdots||k_\ell$ be the hashing key. Let NH_K be a random member of the NH family determined by the key K. Then, the compressed image of M is computed as

$$\mathsf{NH}_K(M) = \sum_{i=1}^{\ell/2}\Big((k_{2i-1} + m_{2i-1} \mod 2^n) \cdot (k_{2i} + m_{2i} \mod 2^n)\Big) \mod 2^{2n}. \tag{25}$$

On the other hand, the hash image of M as computed by the OK family is

$$\mathsf{OK}_K(M) = \sum_{i=1}^{\ell} k_i \cdot m_i \mod 2^n. \tag{26}$$

That is, when the block size is n, the OK computations are performed over $\mathbb{Z}_{2^n}$ while the NH computations are performed over the larger integer ring $\mathbb{Z}_{2^{2n}}$.

To give a numerical example, let $n = 64$ bits. Then, the OK family requires 64-bit computations while the NH family requires 128-bit computations. When using a 64-bit machine, this implies that NH computations must be split over two registers, while OK computations are performed using a single register. Splitting operations over two registers can slow down the speed by about 63%. More importantly, in standard compilers, there is no integer data type of size 128-bit. Therefore, to multiply two 64-bit integers, one needs to split each integer into two 32-bit parts and multiply with appropriate shifts. Using a 64-bit machine with 3.00GHz Intel(R) Xeon(TM) CPU running on UNIX operating system, the NH family runs at 0.87 cycles/byte while the OK family runs at 0.27 cycles/byte.[9]

7.3 Comparison with Current NIST Standard

In this section, we briefly compare the proposed OKH with the Carter-Wegman Counter (CWC) of Kohno et al. [46] and its standardized version the Galois/Counter Mode (GCM) of authenticated encryption [30]. In the comparison, we will focus on the authentication part since all the schemes require similar blockcipher encryption.

As mentioned earlier, both the GCM and the CWC modes first encrypt the plaintext and then authenticate the ciphertext using a universal hash family. The main difference between GCM/CWC and the proposed OKH is in the hashing phase. In both GCM and CWC, the used hash family is based on the POLY family of Bernstein [15]. As can be seen in Table 1, the OK hash family, being non-universal, is faster than the fastest universal hash family in the literature, including the POLY family used in the authentication part of the GCM and the CWC.

7.4 Comparison with Dedicated Authenticated Encryption Schemes

In this section, we compare the proposed OKH with two of the prominent dedicated authenticated encryption schemes, the IAPM of Jutla [43] and the OCB of Rogaway et al. [60]. First, we note that, as stated in [46], the speed of the CWC is comparable to IAPM and OCB in which the CWC can outperform them in some settings and they outperform the CWC in different settings. The main reason for that is because both the IAPM

[9] All codes are written in C.

and OCB require pre-processing (whitening) plaintext blocks before blockcipher encryption. This whitening process is actually a universal hash function. In particular, IAPM requires the generation of pair-wise differentially-uniform sequences named s_i. Each s_i is generated by performing modular multiplication over the finite field $\mathbb{Z}_p$ (similar to the multiplication of the MMH family in Table 1 but with larger primes).[10] In the OCB mode of operation, each message block, $M[i]$, is whitened using a string the authors denoted as $Z[i]$. The computation of each $Z[i]$ requires the generation of a Gray code γ_i (in which each γ_i and γ_{i+1} have a Hamming distance of one), multiply two polynomials over the field GF(2^n), and then take the reminder after dividing the multiplication result by a fixed irreducible polynomial. These polynomial operations are similar to the POLY family of Bernstein listed in Table 1.

Another advantage of the proposed OKH is that all blockcipher calls can be performed in parallel, as can be seen in Figure 1. On the other hand, IAPM and OCB, although parallelizable, require at least three sequential blockcipher calls. This is because the second through second-to-last blockcipher calls depend on the output of the the first blockcipher call; and the last blockcipher call depends on the outputs of all previous blockcipher calls.

7.5 Key Scheduling Effect

It is worth mentioning here that unlike the aforementioned schemes, the proposed OKH requires that the key of each blockcipher is different. Depending on the used blockcipher, due to different key scheduling mechanisms, changing the key can affect the performance of the entire operation. However, key scheduling differs heavily from one blockcipher to another. Even within the AES family itself the key scheduling of AES-128 is different than AES-192 which is, in turn, different than AES-256, let alone other AES candidates. Furthermore, with the fast spreading deployment of mobile and pervasive devices, there is an increasing effort to design new blockciphers suitable for computationally constrained devices (see, e.g., [19, 20, 26, 37, 40, 50, 53, 61, 71]) and, in some of them, light key scheduling is a major design goal. For instance, in the Light Encryption Device (LED) blockcipher there is no key scheduling [37]. Therefore, one will need to provide performance comparisons with multiple blockciphers to see the exact effect of key scheduling. Furthermore, as mentioned earlier, if key rescheduling is an expensive operation, one might resort to tweakable blockciphers instead. However, the main goal of this manuscript is to provide the theoretical basis of the possible advantages of MAC-then-Encrypt compared to the standardized Encrypt-then-MAC constructions. That is, the work is meant to be as independent as possible of the used blockcipher. Therefore, we omit detailed comparison of the effect of key scheduling.

8 Conclusion

In this paper, we proposed the OKH authenticated encryption scheme. By advancing the hashing phase before block cipher encryption, we showed how a hashing function that is not universal can be used without affecting security of authentication. Since the hash function does not have to be universal, it can be computed faster than universal hash function in the literature of cryptography. It was also shown that hashing the plaintext instead of the ciphertext can secure the hashing keys against key recovery attacks.

References

1. V. Afanassiev, C. Gehrmann, and B. Smeets. Fast message authentication using efficient polynomial evaluation. In *Proceedings of the 4th International Workshop on Fast Software Encryption – FSE'97*, volume 1267 of *Lecture Notes in Computer Science*, pages 190–204. Springer, 1997.

[10] The author proposed setting $p = 2^{128} - 159$ for 128-bit blockciphers and $p = 2^{64} - 257$ for 64-bit blockciphers.

2. B. Alomair. Authenticated Encryption: How Reordering can Impact Performance. In *the 10th International Conference on Applied Cryptography and Network Security – ACNS'12*, volume 7341 of *Lecture Notes in Computer Science*, pages 84–99. Springer, 2012.
3. B. Alomair. AVALANCHEv1. In *http://competitions.cr.yp.to/round1/avalanche-corr.pdf*. 2014.
4. B. Alomair. Universal hash-function families: From hashing to authentication. In *Progress in Cryptology–AFRICACRYPT'14*, pages 459–474. Springer, 2014.
5. B. Alomair, A. Clark, and R. Poovendran. The power of primes: security of authentication based on a universal hash-function family. *Journal of Mathematical Cryptology*, 4(2):121–147, 2010.
6. B. Alomair and R. Poovendran. Efficient Authentication for Mobile and Pervasive Computing. *IEEE Transactions on Mobile Computing*, 13(3):469–481, 2014.
7. B. Alomair and R. Poovendran. $\mathcal{E}$-MACs: Towards More Secure and More Efficient Constructions of Secure Channels. *IEEE Transactions on Computers*, 63(1):204–217, 2014.
8. M. Bellare, R. Canetti, and H. Krawczyk. Keying hash functions for message authentication. In *Proceedings of the 16th Annual International Cryptology Conference – CRYPTO'96*, volume 1109 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 1996.
9. M. Bellare, R. Guerin, and P. Rogaway. XOR MACs: New methods for message authentication using finite pseudorandom functions. In *Proceedings of the 15th Annual International Cryptology Conference – CRYPTO'95*, volume 963 of *Lecture Notes in Computer Science*, pages 15–28. Springer, 1995.
10. M. Bellare and T. Kohno. A theoretical treatment of related-key attacks: Rka-prps, rka-prfs, and applications. In *Proceedings of the 22nd International Conference on the Theory and Application of Cryptographic Techniques – EUROCRYPT'03*, Lecture Notes in Computer Science, pages 491–506. Springer, 2003.
11. M. Bellare, T. Kohno, and C. Namprempre. Breaking and provably repairing the SSH authenticated encryption scheme: A case study of the Encode-then-Encrypt-and-MAC paradigm. *ACM Transactions on Information and System Security*, 7(2):241, 2004.
12. M. Bellare and C. Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. *Journal of Cryptology*, 21(4):469–491, 2008.
13. M. Bellare and P. Rogaway. Encode-then-encipher encryption: How to exploit nonces or redundancy in plaintexts for efficient cryptography. In *Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security – ASIACRYPT'00*, volume 1976, pages 317–330. Springer, 2000.
14. M. Bellare, P. Rogaway, and D. Wagner. The EAX mode of operation. In *Proceedings of the 11th International Workshop on Fast Software Encryption – FSE'04*, volume 3017 of *Lecture Notes in Computer Science*, pages 389–407. Springer, 2004.
15. D. Bernstein. Floating-point arithmetic and message authentication. Unpublished manuscript, 2004. Available at `http://cr.yp.to/hash127.html`.
16. D. Bernstein. Cryptographic competitions. `http://competitions.cr.yp.to/index.html`, 2013.
17. J. Black, S. Halevi, H. Krawczyk, T. Krovetz, and P. Rogaway. UMAC: Fast and Secure Message Authentication. In *Proceedings of the 19th Annual International Cryptology Conference – CRYPTO'99*, volume 1666 of *Lecture Notes in Computer Science*, pages 482–499. Springer, 1999.
18. J. Black and P. Rogaway. A block-cipher mode of operation for parallelizable message authentication. In *Proceedings of the 21st International Conference on the Theory and Applications of Cryptographic Techniques – EUROCRYPT'02*, volume 2332 of *Lecture Notes in Computer Science*, pages 384–397. Springer, 2002.
19. A. Bogdanov, L. Knudsen, G. Leander, C. Paar, A. Poschmann, M. Robshaw, Y. Seurin, and C. Vikkelsoe. PRESENT: An Ultra-Lightweight Block Cipher. In *Proceedings of the 9th International Workshop on Cryptographic Hardware and Embedded Systems – CHES'07*, volume 4727 of *Lecture Notes in Computer Science*, pages 450–466. Springer, 2007.
20. J. Borghoff, A. Canteaut, T. Güneysu, E. Kavun, M. Knezevic, L. Knudsen, G. Leander, V. Nikov, C. Paar, C. Rechberger, et al. PRINCE–A Low-Latency Block Cipher for Pervasive Computing Applications. In *Advances in Cryptology–ASIACRYPT 2012*, pages 208–225. Springer, 2012.
21. A. Bosselaers, R. Govaerts, and J. Vandewalle. Fast hashing on the Pentium. In *Proceedings of the 16th Annual International Cryptology Conference – CRYPTO'96*, volume 1109 of *Lecture Notes in Computer Science*, pages 298–312. Springer, 1996.
22. H. Canetti and H. Krawczyk. Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels. In *Proceedings of the 20th International Conference on the Theory and Application of Cryptographic Techniques – EUROCRYPT'01*, volume 2045 of *Lecture Notes in Computer Science*, pages 451–472. Springer, 2001.
23. J. Carter and M. Wegman. Universal classes of hash functions. In *Proceedings of the 9th ACM Symposium on Theory of Computing – STOC'77*, pages 106–112. ACM SIGACT, 1977.
24. L. Carter and M. Wegman. Universal hash functions. *Journal of Computer and System Sciences*, 18(2):143–154, 1979.
25. J. Daemen and V. Rijmen. *The design of Rijndael: AES–the Advanced Encryption Standard*. Springer Verlag, 2002.
26. C. De Canniere, O. Dunkelman, and M. Knežević. KATAN and KTANTANa family of small and efficient hardware-oriented block ciphers. In *Cryptographic Hardware and Embedded Systems–CHES'09*, pages 272–288. Springer, 2009.
27. J. Degabriele and K. Paterson. On the (in)security of IPsec in MAC-then-encrypt configurations. In *Proceedings of the 17th ACM Conference on Computer and Communications Security – CCS'10*, pages 493–504. ACM SIGSAC, 2010.
28. N. Doraswamy and D. Harkins. *IPSec: the new security standard for the Internet, intranets, and virtual private networks*. Prentice Hall, 2003.

29. M. Dworkin. Recommendation for block cipher modes of operation: The CMAC mode for authentication. National Institute of Standards and Technology (NIST) Special Publication 800-38B, 2005.
30. M. Dworkin. Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC. National Institute for Standards and Technology (NIST) Special Publication 800-38D, 2007.
31. M. Etzel, S. Patel, and Z. Ramzan. Square hash: Fast message authentication via optimized universal hash functions. In *Proceedings of the 19th Annual International Cryptology Conference – CRYPTO'99*, volume 1666 of *Lecture Notes in Computer Science*, pages 234–251. Springer, 1999.
32. N. Ferguson, D. Whiting, B. Schneier, J. Kelsey, and T. Kohno. Helix: Fast encryption and authentication in a single cryptographic primitive. In *Proceedings of the 10th International Workshop on Fast Software Encryption – FSE'03*, volume 2887 of *Lecture Notes in Computer Science*, pages 330–346. Springer, 2003.
33. A. Freier, P. Karlton, and P. Kocher. The SSL Protocol Version 3.0. Internet Engineering Task Force (IETF), 2011.
34. V. Gligor and P. Donescu. Integrity-Aware PCBC Encryption Schemes. In *Proceedings of the 7th International Workshop on Security Protocols – SP'99*, volume 1796 of *Lecture Notes in Computer Science*, pages 153–168. Springer, 2000.
35. V. Gligor and P. Donescu. Fast Encryption and Authentication: XCBC Encryption and XECB Authentication Modes. In *Proceedings of the 9th International Workshop on Fast Software Encryption – FSE'01*, volume 2355 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2002.
36. S. Goldwasser and S. Micali. Probabilistic encryption* 1. *Journal of computer and system sciences*, 28(2):270–299, 1984.
37. J. Guo, T. Peyrin, A. Poschmann, and M. Robshaw. The LED block cipher. In *Cryptographic Hardware and Embedded Systems–CHES'11*, pages 326–341. Springer, 2011.
38. S. Halevi and H. Krawczyk. MMH: Software message authentication in the Gbit/second rates. In *Proceedings of the 4th International Workshop on Fast Software Encryption – FSE'97*, volume 1267 of *Lecture Notes in Computer Science*, pages 172–189. Springer, 1997.
39. H. Handschuh and B. Preneel. Key-Recovery Attacks on Universal Hash Function Based MAC Algorithms. In *Proceedings of the 28th Annual International Cryptology Conference – CRYPTO'08*, Lecture Notes in Computer Science, pages 144–161. Springer, 2008.
40. D. Hong, J. Sung, S. Hong, J. Lim, S. Lee, B. Koo, C. Lee, D. Chang, J. Lee, K. Jeong, H. Kim, J. Kim, and S. Chee. HIGHT: A New Block Cipher Suitable for Low-Resource Device. In *Proceedings of the 8th International Workshop on Cryptographic Hardware and Embedded Systems – CHES'06*, volume 4249 of *Lecture Notes in Computer Science*, pages 46–59. Springer, 2006.
41. T. Iwata and K. Kurosawa. omac: One-key cbc mac. In *Proceedings of the 10th International Workshop on Fast Software Encryption – FSE'03*, volume 2887 of *Lecture Notes in Computer Science*, pages 129–153. Springer, 2003.
42. T. Johansson. Bucket hashing with a small key size. In *Proceedings of the 16th International Conference on the Theory and Application of Cryptographic Techniques – EUROCRYPT'97*, volume 1233 of *Lecture Notes in Computer Science*, pages 149–162. Springer, 1997.
43. C. Jutla. Encryption modes with almost free message integrity. *Journal of Cryptology*, 21(4):547–578, 2008.
44. J. Katz and Y. Lindell. *Introduction to modern cryptography*. Chapman & Hall/CRC, 2008.
45. J. Katz and M. Yung. Unforgeable Encryption and Chosen Ciphertext Secure Modes of Operation. In *Proceedings of the 7th International Workshop on Fast Software Encryption – FSE'00*, volume 1978 of *Lecture Notes in Computer Science*, pages 25–36. Springer, 2001.
46. T. Kohno, J. Viega, and D. Whiting. CWC: A high-performance conventional authenticated encryption mode. In *Proceedings of the 11th International Workshop on Fast Software Encryption – FSE'04*, volume 3017 of *Lecture Notes in Computer Science*, pages 408–426. Springer, 2004.
47. H. Krawczyk. LFSR-based hashing and authentication. In *Proceedings of the 14th Annual International Cryptology Conference – CRYPTO'94*, volume 839 of *Lecture Notes in Computer Science*, pages 129–139. Springer, 1994.
48. H. Krawczyk. New hash functions for message authentication. In *Proceedings of the 15th Annual International Cryptology Conference – CRYPTO'95*, volume 921 of *Lecture Notes in Computer Science*, pages 301–310. Springer, 1995.
49. H. Krawczyk. The order of encryption and authentication for protecting communications(or: How secure is SSL?). In *Proceedings of the 21st Annual International Cryptology Conference – CRYPTO'01*, volume 2139 of *Lecture Notes in Computer Science*, pages 310–331. Springer, 2001.
50. G. Leander, C. Paar, A. Poschmann, and K. Schramm. New lightweight DES variants. In *Fast Software Encryption–FSE'07*, pages 196–210. Springer, 2007.
51. M. Liskov, R. Rivest, and D. Wagner. Tweakable block ciphers. In *Advances in Cryptology–CRYPTO'02*, pages 31–46. Springer, 2002.
52. U. Maurer and B. Tackmann. On the soundness of authenticate-then-encrypt: formalizing the malleability of symmetric encryption. In *Proceedings of the 17th ACM Conference on Computer and Communications Security – CCS'10*, pages 505–515. ACM SIGSAC, 2010.
53. A. Moradi, A. Poschmann, S. Ling, C. Paar, and H. Wang. Pushing the limits: a very compact and a threshold implementation of AES. In *Advances in Cryptology–EUROCRYPT'11*, pages 69–88. Springer, 2011.
54. F. Muller. Differential attacks against the Helix stream cipher. In *Proceedings of the 11th International Workshop on Fast Software Encryption – FSE'04*, volume 3017 of *Lecture Notes in Computer Science*, pages 94–108. Springer, 2004.

55. W. Nevelsteen and B. Preneel. Software performance of universal hash functions. In *Proceedings of the 18th International Conference on the Theory and Application of Cryptographic Techniques – EUROCRYPT'99*, volume 1592 of *Lecture Notes in Computer Science*, pages 24–41. Springer, 1999.
56. S. Paul and B. Preneel. Near Optimal Algorithms for Solving Differential Equations of Addition with Batch Queries. In *Proceedings of the 6th International Conference on Cryptology in India – INDOCRYPT'05*, volume 3797 of *Lecture Notes in Computer Science*, pages 90–103. Springer, 2005.
57. S. Paul and B. Preneel. Solving systems of differential equations of addition. In *Proceedings of the 10th Australasian Conference on Information Security and Privacy – ICISP'05*, volume 3574 of *Lecture Notes in Computer Science*, pages 75–88. Springer, 2005.
58. B. Preneel and P. Van Oorschot. MDx-MAC and building fast MACs from hash functions. In *Proceedings of the 15th Annual International Cryptology Conference – CRYPTO'95*, volume 963 of *Lecture Notes in Computer Science*, pages 1–14. Springer, 1995.
59. P. Rogaway. Bucket hashing and its application to fast message authentication. *Journal of Cryptology*, 12(2):91–115, 1999.
60. P. Rogaway, M. Bellare, and J. Black. OCB: A Block-Cipher Mode of Operation for Efficient Authenticated Encryption. *ACM Transactions on Information and System Security*, 6(3):365–403, 2003.
61. K. Shibutani, T. Isobe, H. Hiwatari, A. Mitsuda, T. Akishita, and T. Shirai. Piccolo: an ultra-lightweight blockcipher. In *Cryptographic Hardware and Embedded Systems–CHES'11*, pages 342–357. Springer, 2011.
62. V. Shoup. On fast and provably secure message authentication based on universal hashing. In *Proceedings of the 16th Annual International Cryptology Conference – CRYPTO'96*, volume 1109 of *Lecture Notes in Computer Science*, pages 313–328. Springer, 1996.
63. D. Stinson. Universal hashing and authentication codes. *Designs, Codes and Cryptography*, 4(3):369–380, 1994.
64. J. Tignol. *Galois' Theory of Algebraic Equations*. World Scientific, 2001.
65. G. Tsudik. Message authentication with one-way hash functions. *ACM Computer Communication Review*, 22(5):38, 1992.
66. H. van Tilborg. *Encyclopedia of cryptography and security*. Springer, 2005.
67. P. Wang, W. Wu, and L. Zhang. Cryptanalysis of the okh authenticated encryption scheme. In *International Conference on Information Security Practice and Experience–ISPEC'13*, volume 7863 of *Lecture Notes in Computer Science*, pages 353–360. Springer, 2013.
68. M. Wegman and J. Carter. New classes and applications of hash functions. In *Proceedings of the 20th Annual IEEE Symposium on Foundations of Computer Science – FOCS'79*, pages 175–182. IEEE Computer Society, 1979.
69. D. Whiting, B. Schneier, S. Lucks, and F. Muller. Phelix – fast encryption and authentication in a single cryptographic primitive. ECRYPT Stream Cipher Project, Report 2005/020, `www.ecrypt.eu.org/stream`, 2005.
70. H. Wu and B. Preneel. Differential-linear attacks against the stream cipher Phelix. In *Proceedings of the 14th International Workshop on Fast Software Encryption – FSE'07*, volume 4593 of *Lecture Notes in Computer Science*, pages 87–100. Springer, 2007.
71. W. Wu and L. Zhang. LBlock: a lightweight block cipher. In *Applied Cryptography and Network Security–ACNS'11*, pages 327–344. Springer, 2011.
72. T. Ylonen and C. Lonvick. The Secure Shell (SSH) Transport Layer Protocol. Technical report, RFC 4253, 2006.