A General Construction of Tweakable Block Ciphers and Different Modes of Operations

Debrup Chakraborty and Palash Sarkar

***Abstract*—This work builds on earlier work by Rogaway at Asiacrypt 2004 on tweakable block cipher (TBC) and modes of operations. Our first contribution is to generalize Rogaway's TBC construction by working over a ring R and by the use of a masking sequence of functions. The ring R can be instantiated as either GF(2^n) or as $\mathbb{Z}_{2^n}$. Further, over GF(2^n), efficient instantiations of the masking sequence of functions can be done using either a binary linear feedback shift register (LFSR); a powering construction; a cellular automata map; or by using a word-oriented LFSR. Rogaway's TBC construction was built from the powering construction over GF(2^n). Our second contribution is to use the general TBC construction to instantiate constructions of various modes of operations including authenticated encryption (AE) and message authentication code (MAC). In particular, this gives rise to a family of efficient one-pass AE modes of operation. Out of these, the mode of operation obtained by the use of word-oriented LFSR promises to provide a masking method which is more efficient than the one used in the well known AE protocol called OCB1.**

***Index Terms*—Authenticated encryption with associated data, message authentication code, modes of operations, tweakable block cipher (TBC).**

I. Introduction

SYMMETRIC ciphers form the backbone of encryption technology since all bulk encryptions are done using symmetric ciphers. A block cipher has to be used in an appropriate mode of operation for performing such encryption. Thus, designing efficient and secure modes of operations is as important as developing a secure block cipher.

Of special practical importance are modes of operations for authenticated encryption (AE). This allows both confidentiality and authentication in transmission of messages over an insecure channel. Conventional approaches to this problem require two block cipher invocations per block of the message. In recent years, there have been several proposals for AE which require one invocation per block of the message. This yields efficiency improvement by a factor of two over conventional approaches. The known one-pass proposals are IACBC, IAPM by Jutla [10]; XCBC, XECB by Gligor-Donescu [7]; and OCB, OCB1 by Rogaway [19].

All the above proposals are patented. This has prevented their adoption in NIST standards. In fact, NIST [1] has standardized a two-pass algorithm for achieving AE. Another undesirable effect of the patent claims is that this has led to some researchers proposing new two-pass AE protocols [3], [14].

A. Tweakable Block Ciphers

Liskov, Rivest, and Wagner [13] introduced the concept of tweakable block cipher (TBC) which is a block cipher with an additional input called a tweak. The tweak is meant to provide variability and not security. More formally, an n-bit TBC is a map $\widetilde{E} : \mathcal{K} \times \mathcal{T} \times \{0,1\}^n \rightarrow \{0,1\}^n$, where $\mathcal{K}$ is the key space and $\mathcal{T}$ is the tweak space. For every fixed $K \in \mathcal{K}$ and $T \in \mathcal{T}$, $\widetilde{E}(K,T,\cdot)$ denoted by $\widetilde{E}_K^T(\cdot)$ is a permutation of $\{0,1\}^n$. The key K is secret. A TBC encrypts an n-bit message M under a secret key K and a nonsecret tweak T to obtain an n-bit ciphertext $C = \widetilde{E}_K^T(M)$. Decryption is similar.

One of the primary motivations in [13] for introducing TBC was to build secure modes of operations starting from a TBC. However, the modes of operations given in [13] were not efficient. The theme of designing a suitable TBC and modes of operations based on it was developed by Rogaway in [19].

B. Rogaway's Construction of TBC and Modes of Operations

The work [19] makes two contributions.

First, an efficient method is given for constructing a TBC from a usual block cipher. Each tweak T consists of an n-bit string and a tuple of nonnegative integers. Efficiency in this context means the following. If T' is a tweak which is obtained from T by incrementing one of the components, then the cost of computing $\widetilde{E}_K^{T'}(M')$ having already computed $\widetilde{E}_K^T(M)$ is one block cipher call plus a small and constant number of shifts, XOR's and conditional operations. This construction is called the powering construction. More details of the construction are given in the context of our contribution in Section I-C.

The second contribution is to obtain various modes of operations of a block cipher using a TBC with an appropriate tweak space as an intermediate step. For example, an AE protocol is obtained in two steps. In the first step, a secure AE protocol is constructed from a TBC with tweak space $\{0,1\}^n \times \{0,1,2,\ldots\} \times \{0,1\}$; and in the second step it is shown how to construct such a TBC from a block cipher using the earlier mentioned method.

An important consequence is to obtain efficient constructions of several modes of operations. Use of TBC as an intermediate step makes the constructions cleaner and also the proofs clearer and shorter.

Manuscript received July 24, 2007; revised December 13, 2007. An earlier abridged version of the paper was published in *Proc. Inscrypt (Lecture Notes in Computer Science)*. Berlin, Germany: Springer-Verlag, 2002, vol. 4318, pp. 88–102.

D. Chakraborty is with the Computer Science Department, CINVESTAV-IPN, Av. IPN 2508, Mexico D.F. 07360, Mexico (e-mail: debrup@cs.cinvestav.mx).

P. Sarkar is with the Applied Statistics Unit, Indian Statistical Institute, Kolkata, Pin 700 108, India (e-mail: palash@isical.ac.in).

Communicated by E. Okamoto, Associate Editor for Complexity and Cryptography.

Digital Object Identifier 10.1109/TIT.2008.920247

C. Our Contributions

In this paper, we develop the work on construction of efficient TBC and modes of operations based on it. Our work depends heavily on the work of Rogaway [19]. Below we mention our specific contributions and relate to the work of [19].

Tweakable Block Cipher: We define a sequence $f_1, f_2, \ldots, f_{2^n-2}$, with $f_i : \{0,1\}^n \to \{0,1\}^n$, of functions with a particular set of properties to be a masking sequence. Given block cipher $E : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ and a masking sequence, we define a TBC having tweak space $\mathcal{T} = \{0,1\}^n \times \{1, \ldots, 2^n - 2\}$ by either the XE or the XEX constructions.

In the XE construction:

$$\widetilde{E}_K^{N,i}(M) = E_K(M + f_i(\mathcal{N}))$$

whereas in the XEX construction:

$$\widetilde{E}_K^{N,i}(M) = E_K(M + f_i(\mathcal{N})) - f_i(\mathcal{N})$$

where (N, i) is the tweak and $\mathcal{N} = E_K(N)$. Addition (and subtraction) is over a commutative ring $\boldsymbol{R} = (\{0,1\}^n, +, \cdot)$ with identity. Typical instantiations of $\boldsymbol{R}$ are as GF(2^n) and $\mathbb{Z}_{2^n}$.

In the case where $\boldsymbol{R}$ is GF(2^n), we use a primitive polynomial $\tau(x)$ to represent GF(2^n) and consider $\mathcal{N}$ to be an n-bit vector. The map $f_i(\mathcal{N})$ is defined to be $f_i(\mathcal{N}) = \mathcal{N}G^i$, where G is an $n \times n$ matrix over GF(2) having $\tau(x)$ as its characteristic polynomial. Efficient realization of G can be done by a linear feedback shift register (LFSR), a powering construction used in [19] or as a cellular automata (CA) map.

Another representation of GF(2^n) is as a tower of fields. Under this representation, one can use a word-oriented LFSR to define the f_i's. Details are given in Section IV-B.

In the case where $\boldsymbol{R}$ is $\mathbb{Z}_{2^n}$, we define

$$f_i(\mathcal{N}) = ((i+1)\mathcal{N} \bmod p) \bmod 2^n$$

where $p = 2^n + \delta$ is the least prime greater than 2^n.

The XE and the XEX constructions were presented in [19] over GF(2^n) using the powering construction. The abstraction of the ring $\boldsymbol{R}$, the use of LFSR, CA, word-oriented LFSR over GF(2^n); and the instantiation of $\boldsymbol{R}$ as $\mathbb{Z}_{2^n}$ are new in this paper.

Authenticated Encryption: As mentioned in Section I-B, Rogaway constructs an AE protocol from a TBC and shows how to instantiate the TBC with a block cipher using the powering up construction. This instantiation requires the computation of a discrete logarithm over GF(2^n).

We show two methods to instantiate Rogaway's AE construction with our general TBC construction. The first method, which we call linear separation, is based on Rogaway's technique. Thus, as in the case of Rogaway, when we work over GF(2^n), the linear separation method requires the computation of a discrete logarithm (as a one-time design stage activity). The second method, which we call interleaved separation, is introduced in this paper. This method does not require the discrete log computation and hence is more generally applicable.

In [19], Rogaway also presents constructions of pseudorandom function (PRF), message authentication code (MAC), and authenticated encryption with associated data (AEAD) protocols from TBCs with appropriate tweak spaces and shows how to instantiate these with his TBC construction. We show how to instantiate the PRF, MAC, and AEAD protocols of Rogaway with the general TBC construction using the techniques of linear and interleaved separation.

In summary, our generalization of Rogaway's work comes in two parts.

1) Rogaway describes the XE and the XEX constructions over GF(2^n) using the powering construction. We generalize this by working over a ring $\boldsymbol{R}$ which can be instantiated as either GF(2^n) or as $\mathbb{Z}_{2^n}$. Over GF(2^n), we show that there are other efficient alternatives to the powering construction. Use of LFSR or CA provides similar efficiency as the powering construction, while the use of word-oriented LFSR promises to be faster.
2) Rogaway presents constructions of several modes of operations from TBCs with appropriate tweak spaces and shows how to instantiate these with his TBC constructions. We generalize his method of instantiation and also present a new way of instantiation of the different modes of constructions with the generalized TBC constructions.

A net effect of our generalization is to uncover a *family* of efficient, previously unknown protocols for AE, PRF, MAC, and AEAD. Rogaway's construction is a special case of this family.

D. Practical Significance of our Work

Rogaway's work [19] on AE, MAC, and AEAD provides very efficient constructions with tight security bounds. For example, the AE construction is fully parallelizable; makes $(m+2)$ block cipher calls for an m-block message; and uses an efficient method to generate the masks required. The security bound is already tight and it is quite unlikely that the efficiency can be significantly improved. So, what can one hope to achieve in the context of such excellent prior work?

The starting point of our work is that Rogaway presents a *single* example of each mode of operation. A natural question that we ask is whether there are other constructions with comparable security and efficiency. Our results show that there are indeed such constructions. We uncover a whole family of constructions which provides a developer with a wide variety of choices. This, by itself, may be considered to be of some practical importance.

In both Rogaway's work and our generalization of the AE protocol, the number of block cipher calls for an m block message is $m+2$. Also, the time for executing the block cipher calls dominates the total time for encryption. However, it is possible to improve upon the efficiency of the mask generation procedure used in Rogaway's algorithm.

As mentioned earlier, one of the methods to implement the masking sequence is to use a word-oriented LFSR. Experience from the stream cipher design community suggests that for software implementation, a word-oriented LFSR is faster than a usual binary LFSR and the powering method. As a result, the AE mode of operation obtained from linear separation and masking using word-oriented LFSRs promises to be faster than the AE protocol (called OCB1) given in [19], which is based on the powering method.

Further, while the security and efficiency of [19] cannot be significantly improved (because they are already quite tight),

one of our constructions offers a flexibility of usage which is not available in Rogaway's work [19]. This has to do with the design stage discrete log computation required in [19]. The discrete log computation is required for different block sizes. More importantly, even for a fixed block size, the discrete log computation is required if the field representing polynomial is changed.

Easily Reconfigurable Family of Modes of Operations: Let us consider the AE protocol, though the discussion below applies equally well to the other protocols. As mentioned earlier, the ring $\boldsymbol{R}$ that we work over can be instantiated as GF(2^n). The idea is to view the AE mode of operation over GF(2^n) as being parameterized by the primitive polynomial $\tau(x)$ which represents the field. As a result, for every choice of $\tau(x)$ one obtains a specific mode of operation. Security is not affected—the security bound does not depend on $\tau(x)$ and remains the same for every choice of $\tau(x)$. There are situations where such a parameterized family of AE modes of operations may be useful. We outline one such possibility.

Consider the following scenario: A crypto company which develops AE modes of operations has many customers. All customers want a provably secure single-pass AE solution. However, they also require that the specific design that they will be using should be kept secret. In the paranoid world of crypto customers, especially from different national defence establishments, this can be a practical requirement.

Is it possible to satisfy such a customer requirement? The answer is yes, at least to a certain extent. The customer can randomly choose the primitive polynomial $\tau(x)$ and keep it a secret. By doing this, the customer does not loose either provable security or efficiency. Basically, in this context, provable security tells him that even if $\tau(x)$ is known, the protocol is as secure as the underlying block cipher. Now, by keeping $\tau(x)$ unknown, he gains an extra level of confidence, since knowledge of $\tau(x)$ is required to attack the system. The only condition on $\tau(x)$ is that it should be primitive. Since the number of primitive polynomials of degree n is quite large (for $n = 128$, there are around 2^{119} primitive polynomials), the customer can be assured that an adversary has a rather high uncertainty (about 119 bits) about the specific polynomial he is using.

First, suppose our crypto company wants to use Rogaway's construction to satisfy the needs of the customers. In Rogaway's construction, for each change of $\tau(x)$, a discrete log computation needs to be performed. The purpose of this computation is to ensure that the discrete log of $(x + 1)$ modulo $\tau(x)$ should be "large," since, otherwise, the proof of security breaks down. This requirement of a discrete log computation per change of polynomial makes Rogaway's construction unsuitable for the above application.

Now consider the technique of interleaved separation (introduced in this paper) to construct an AE mode of operation with $\boldsymbol{R}$ instantiated as GF(2^n). Unlike Rogaway's AE mode of operation, this mode of operation does not require any discrete log computation in the design phase. It is due to this difference, that one can obtain a greater flexibility of usage. Our crypto company creates a single product with $\tau(x)$ as a parameter. In software, this can be provided as an n-bit string, while in hardware, this is kept in a register of length n. This single product is given to a customer. The customer "customizes" this product by choosing a random primitive polynomial of degree n and plugging it into the design. No discrete log computation is required at any stage. Further, in a manner somewhat like a regular key change, the polynomial can also be changed by the customer at regular intervals. This idea can satisfy the customer's apparently conflicting requirements of provable security and obscurity.

We feel that the above practical issue will be attractive to crypto companies who actually develop crypto protocols. They gain a lot of flexibility at no extra cost and at no loss in security. On the other hand, theoreticians might not appreciate this advantage (and may consider the above application as artificial). For them, the abstraction of the masking sequence and the generalized versions of the XE and the XEX constructions will be of more interest.

E. Other Previous and Related Works

The formal model of security for AE was independently proposed by [11] and [2]. Jutla [10] proposed constructions for single-pass AE protocol, including one fully parallelizable protocol. Independent work due to Gligor and Donescu [7] also proposed single-pass AE protocols. A refinement and extension of Jutla's parallelizable protocol was done by Rogaway [20] and was called the OCB. (The masking strategy for the AE protocol OCB1 given in [19] is faster than that of OCB.)

Construction of MAC and AEAD protocols are also of equal importance. There has been a lot of research on the security model and design of these protocols [4], [9], [18]. A separate line of research has consisted of developing two-pass AE protocols (some examples are [15], [3], [14]). The work [14] presents an AE protocol which is somewhere between one and two pass protocols.

In a recent work, Minematsu [16] revisits the work on TBC appearing in [13] and [19]. The work [16] provides some improvements to the construction given in [13]. The XEX construction in [19] is presented in a more general form than what has been mentioned earlier in this paper. However, in its application to the construction of modes of operations, this generality is not required and a much more simpler form is used. In this paper, we have generalized this simpler form. In contrast, Minematsu [16] presents a new analysis of the XEX description as given in [19]. We would like to emphasize that none of the techniques for XEX construction introduced in this paper is present in [16] Also, none of the techniques for constructing modes of operations is present in [16]. Thus, this work and that of [16] though on the similar topics, are really of independent interest.

II. Preliminaries

Our notation and definitions closely follow [19].

A block cipher is a map $E : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$, where $\mathcal{K}$ is a finite nonempty set called the key space and for all $K \in \mathcal{K}$, $E(K,\cdot) = E_K(\cdot)$ is a permutation of $\{0,1\}^n$. A TBC is a map $\widetilde{E} : \mathcal{K} \times \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$, where $\mathcal{T}$ is a finite nonempty set called the tweak space and $\widetilde{E}(K,T,\cdot) = \widetilde{E}_K^T(\cdot)$ is a permutation of $\{0,1\}^n$. The inverse D of a block cipher is a map $D = E^{-1}$ such that $D(K, E(K, X)) = X$. Similarly, the inverse of a TBC satisfies $\widetilde{D}(K,T,\widetilde{E}(K,T,X)) = X$.

$\mathrm{Perm}(n)$ denotes the set of all permutations of $\{0,1\}^n$ and $\mathrm{Perm}(\mathcal{T},n)$ denotes the set of all mappings from $\mathcal{T}$ to

$\mathrm{Perm}(n)$. Similarly, $\mathrm{Rand}(n)$ denotes the set of all n bit to n bit functions and $\mathrm{Rand}(\mathcal{T}, n)$ denotes the set of all mappings from $\mathcal{T}$ to $\mathrm{Rand}(n)$. The notation $\pi \xleftarrow{\$} \mathrm{Perm}(n)$ denotes the choice of a random permutation on n bits while $\pi \xleftarrow{\$} \mathrm{Perm}(\mathcal{T}, n)$ denotes the choice of a random permutation $\pi(T, \cdot) = \pi_T(\cdot)$ for each element $T \in \mathcal{T}$.

An adversary is a probabilistic algorithm with possible access to encryption and/or decryption oracles. The notation $A^{O_1, O_2} \Rightarrow 1$ denotes the event that an adversary A outputs 1 after interacting with the oracles O_1 and O_2. We will assume that an adversary does not ask a query for which it can easily obtain an answer. Thus, it never repeats a query; does not ask for the decryption of a ciphertext which it has previously received as an output of an encryption query; and neither does it ask for the encryption of a plaintext which it has previously received as output of a decryption query. The notation $\mathbf{Adv}(A)$ denotes the advantage of an adversary A. The definitions of various advantages are as follows.

Definition 1: Let $E_K(\cdot)$ and $\widetilde{E}_K^T(\cdot)$ be a block cipher and a TBC, respectively, and let A be an adversary. We define the following advantages.

$$\begin{aligned}
&\mathbf{Adv}_E^{\mathrm{prp}}(A) \\
&\quad = \mathsf{Prob}[K \xleftarrow{\$} \mathcal{K} : A^{E_K(\cdot)} \Rightarrow 1] \\
&\qquad - \mathsf{Prob}[\pi \xleftarrow{\$} \mathrm{Perm}(n) : A^{\pi(\cdot)} \Rightarrow 1]. \\
&\mathbf{Adv}_E^{\pm\mathrm{prp}}(A) \\
&\quad = \mathsf{Prob}[K \xleftarrow{\$} \mathcal{K} : A^{E_K(\cdot), D_K(\cdot)} \Rightarrow 1] \\
&\qquad - \mathsf{Prob}[\pi \xleftarrow{\$} \mathrm{Perm}(n) : A^{\pi(\cdot), \pi^{-1}(\cdot)} \Rightarrow 1]. \\
&\mathbf{Adv}_{\widetilde{E}}^{\widetilde{\mathrm{prp}}}(A) \\
&\quad = \mathsf{Prob}[K \xleftarrow{\$} \mathcal{K} : A^{\widetilde{E}_K(\cdot,\cdot)} \Rightarrow 1] \\
&\qquad - \mathsf{Prob}[\pi \xleftarrow{\$} \mathrm{Perm}(\mathcal{T}, n) : A^{\pi(\cdot,\cdot)} \Rightarrow 1]. \\
&\mathbf{Adv}_{\widetilde{E}}^{\pm\widetilde{\mathrm{prp}}}(A) \\
&\quad = \mathsf{Prob}[K \xleftarrow{\$} \mathcal{K} : A^{\widetilde{E}_K(\cdot,\cdot), \widetilde{D}_K(\cdot,\cdot)} \Rightarrow 1] \\
&\qquad - \mathsf{Prob}[\pi \xleftarrow{\$} \mathrm{Perm}(\mathcal{T}, n) : A^{\pi(\cdot,\cdot), \pi^{-1}(\cdot,\cdot)} \Rightarrow 1].
\end{aligned}$$

Here D and $\widetilde{D}$ denote the inverses of E and $\widetilde{E}$, respectively.

The extension of these advantages to resource bounded advantages are done in the usual manner:

$$\mathbf{Adv}_{\Pi}^{\mathrm{xxx}}(\mathcal{R}) = \sup_A \{\mathbf{Adv}_{\Pi}^{\mathrm{xxx}}(A)\}$$

over all adversaries A that use resources at most $\mathcal{R}$. The resources of interest are the number of queries q made by the adversary, the total number σ_n of n-bit blocks provided by the adversary in all its queries and the running time t.

III. Construction of Tweakable Block Ciphers

Let $\boldsymbol{R} = (\{0,1\}^n, +, \cdot)$ be a commutative ring with identity. We define a sequence of functions.

Definition 2 (Masking Sequence): Let $f_1, f_2, \ldots, f_m$ be a sequence of functions where each $f_s : \{0,1\}^n \to \{0,1\}^n$. We say that the sequence is an (n, m, μ) masking sequence if the following properties hold for a fixed element α of $\{0,1\}^n$.

(1) $\mathsf{Prob}[f_s(\mathcal{N}) = \alpha] \leq \frac{1}{\mu}$, for $1 \leq s \leq m$.
(2) $\mathsf{Prob}[f_s(\mathcal{N}) = \mathcal{N} + \alpha] \leq \frac{1}{\mu}$, for $1 \leq s \leq m$.
(3) $\mathsf{Prob}[f_s(\mathcal{N}) = f_t(\mathcal{N}) + \alpha] \leq \frac{1}{\mu}$, for $1 \leq s, t \leq m$ and $s \neq t$.
(4) $\mathsf{Prob}[f_s(\mathcal{N}) = f_t(\mathcal{N}') + \alpha] \leq \frac{1}{\mu}$, for $1 \leq s, t \leq m$.

Here the operation " $+$" is over $\boldsymbol{R}$. The probabilities are taken over independent and random choices of $\mathcal{N}$ and $\mathcal{N}'$ from $\{0,1\}^n$.

In our constructions of f_s's we will have μ to be either equal to or slightly less than 2^n. There is an efficiency consideration while defining the f's. Given the value of $f_s(\mathcal{N})$, it should be "easy" to compute $f_{s+1}(\mathcal{N})$.

Property (3) of a masking sequence is reminiscent of the definition of almost universal hash functions. This is a keyed family of hash functions, such that for a randomly chosen key from the key space, the probability that two distinct messages collide for the corresponding hash function is low. If Property (3) is viewed in this way, $\mathcal{N}$ will correspond to the key of the hash function family, whereas s and t will be the distinct messages. Thus, the correspondence is not very natural and hence we do not explore it any further.

The construction of a TBC that we present below is a natural generalization of the construction given in [19]. We construct a TBC

$$\widetilde{E} : \mathcal{K} \times (\{0,1\}^n \times \{1, 2, \ldots, 2^n - 2\}) \times \{0,1\}^n \to \{0,1\}^n.$$

The tweak space $\mathcal{T} = \{0,1\}^n \times \{1, 2, \ldots, 2^n - 2\}$. We write $\widetilde{E}_K^{N,l}(M)$ to denote $\widetilde{E}(K, (N, l), M)$.

XE Construction: In this construction, $\widetilde{E}_K^{N,l}(M)$ is defined as follows:

$$\widetilde{E}_K^{N,l}(M) = E_K(M + \Delta) \tag{1}$$

where $\Delta = f_l(\mathcal{N})$ and $\mathcal{N} = E_K(N)$.

XEX Construction: In this construction, $\widetilde{E}_K^{N,l}(M)$ is defined as follows:

$$\widetilde{E}_K^{N,l}(M) = E_K(M + \Delta) - \Delta \tag{2}$$

where $\Delta = f_l(\mathcal{N})$ and $\mathcal{N} = E_K(N)$.

The operations " $+$" and " $-$" in the XE and the XEX constructions are over the ring $\boldsymbol{R}$. Further, the function $f_l()$ is from an $(n, 2^n - 2, \mu)$ masking sequence.

The Δ's act as masks. In the XE construction, the message block is masked, while in the XEX construction both the message block and the output of the encryption are masked. The XE and the XEX constructions were introduced by Rogaway [19]. We generalize by working over $\boldsymbol{R}$ and the use of the masking sequence of functions. Later we show that there are several different ways of efficiently instantiating $\boldsymbol{R}$ and the masking sequence.

We next prove the security of the XE and the XEX constructions. The proof of the XE construction is very similar to that given in [19]. The proof of the XEX construction was not given

in [19] and it was remarked that the proof is similar to that of the XE construction. However, the proof of the XEX construction requires an additional consideration of the range set of a random function and collisions in the range set. Avoiding such collisions requires a little more subtlety than the proof of the XE construction given in [19]. The following result generalizes the XE and the XEX construction of Rogaway by the use of the masking sequence of functions.

Theorem 1 (Security of XE and XEX Constructions):

Security of XE:

$$\mathbf{Adv}^{\widetilde{\mathrm{prp}}}_{\widetilde{E}}(t,q) \leq \mathbf{Adv}^{\mathrm{prp}}_{E}(t',2q) + \frac{5q^2}{2^{n+1}} + \frac{2q^2}{\mu}. \quad (3)$$

Security of XEX:

$$\mathbf{Adv}^{\widetilde{\pm\mathrm{prp}}}_{\widetilde{E}}(t,q) \leq \mathbf{Adv}^{\pm\mathrm{prp}}_{E}(t',2q) + \frac{5q^2}{2^{n+1}} + \frac{4q^2}{\mu} \quad (4)$$

In both the above inequalities, $t'=t+cq+c'$ for constants c, c'.

Proof: The proofs of the two constructions are presented separately.

Proof of the XE Construction: As in [19], a hybrid argument is required. The following five hybrids were identified in [19].

1) $p_1 = \mathsf{Prob}[k \xleftarrow{\$} \mathcal{K} : A^{\widetilde{E}_K(\cdot,\cdot)} \Rightarrow 1]$.
2) $p_2 = \mathsf{Prob}[\pi \xleftarrow{\$} \mathrm{Perm}(n) : A^{\widetilde{\pi}(\cdot,\cdot)} \Rightarrow 1]$.
3) $p_3 = \mathsf{Prob}[\rho \xleftarrow{\$} \mathrm{Rand}(n) : A^{\widetilde{\rho}(\cdot,\cdot)} \Rightarrow 1]$.
4) $p_4 = \mathsf{Prob}[\rho \xleftarrow{\$} \mathrm{Rand}(\mathcal{T},n) : A^{\rho(\cdot,\cdot)} \Rightarrow 1]$.
5) $p_5 = \mathsf{Prob}[\pi \xleftarrow{\$} \mathrm{Perm}(\mathcal{T},n) : A^{\pi(\cdot,\cdot)} \Rightarrow 1]$.

We have to bound $p_1-p_5=(p_1-p_2)+(p_2-p_3)+(p_3-p_4)+(p_4-p_5)$. The bounds on (p_1-p_2), (p_2-p_3), and (p_4-p_5) obtained in [19] also hold in our case. These bounds are as follows.

1) $p_1 - p_2 \leq \mathbf{Adv}^{\mathrm{prp}}_{E}(t',2q)$.
2) $p_2 - p_3 \leq 2q^2/2^n$.
3) $p_4 - p_5 \leq 0.5q^2/2^n$.

The main part of the proof is to bound p_3-p_4. We consider two games G_3 and G_4.

Game G_3: Each adversarial query is a triple (N,l,M), where (N,l) is the tweak and M is the message block. At the outset, a flag bad is set to false and the function $\rho(\cdot)$ is declared to be undefined everywhere. As the adversary's queries are answered, the function $\rho(\cdot)$ begins to get defined at certain points of the domain. Let $\mathrm{Domain}(\rho)$ denote the set of points at which ρ has currently been defined. Thus, initially $\mathrm{Domain}(\rho)$ is empty. The adversary then starts its queries. The jth query is denoted by (N^j,l^j,M^j) and is answered as follows.

1. **If** $N^j = N^i$ for some $i<j$ **then** $\mathcal{N}^j = \mathcal{N}^i$;
2. **else**
3. $\quad \mathcal{N}^j \xleftarrow{\$} \{0,1\}^n$;
4. $\quad$ **If** $N^j \in \mathrm{Domain}(\rho)$
   $\quad$ **then** bad $=$ true; $\boxed{\mathcal{N}^j = \rho(N^j);}$
5. $\rho(N^j) = \mathcal{N}^j$;
6. $Y^j \xleftarrow{\$} \{0,1\}^n$; $X^j = M^j + f_{l^j}(\mathcal{N}^j)$;
7. **If** $X^j \in \mathrm{Domain}(\rho)$
   $\quad$ **then** bad $=$ true; $\boxed{Y^j = \rho(X^j);}$
8. $\rho(X^j) = Y^j$;
9. **Return** Y^j

The above is similar to the algorithm given in of [19, Fig. 1] with one exception. In Step 6, we use the function $f_{l^j}(\cdot)$ and the addition $+$ is over the ring $\boldsymbol{R}$.

Game G_4: This game is the same as G_3 except that the statement $\mathcal{N}^j = \rho(N^j)$ in Step 3 and the statement $Y^j = \rho(X^j)$ in Step 7 are dropped.

Game G_3 is an accurate simulation of the game defining the experiment associated with p_3 while G_4 does this for p_4. The games G_3 and G_4 are identical until the flag bad is set to true. Thus, we have $p_3 - p_4 \leq \mathsf{Prob}[A \text{ sets bad to true in } G_3]$. We now have to upper-bound this probability.

The Y^j values are returned to the adversary. These are random quantities and the adversary could as well have generated these by itself. Thus, these provide the adversary with no information and we may assume that the adversary is nonadaptive. It asks a fixed sequence $(N^1,l^1,M^1),\ldots,(N^q,l^q,M^q)$ of queries hoping that some N^i and X^j will collide, or some X^i and X^j will collide. We now bound the probability of such collisions.

Case N^i, X^j: Recall $X^j = M^j + f_{l^j}(\mathcal{N}^j)$. Thus, $X^j - N^i = (M^j - N^i) + f_{l^j}(\mathcal{N}^j) = -\alpha + f_{l^j}(\mathcal{N}^j)$ for some fixed $\alpha \in \{0,1\}^n$. By the first property of the masking sequence of functions (see Definition 2), we have

$$\mathsf{Prob}[N^i = X^j] = \mathsf{Prob}[f_{l^j}(\mathcal{N}^j) = \alpha] \leq \frac{1}{\mu}.$$

Case X^i, X^j: This leads to two subcases.

Subcase $N^i \neq N^j$: In this case, $\mathcal{N}^i$ and $\mathcal{N}^j$ are chosen in the Game G_3 to be independent and uniformly distributed random quantities from $\{0,1\}^n$. We have

$$\begin{aligned}&\mathsf{Prob}[X^i = X^j]\\ &\quad = \mathsf{Prob}[(M^i - M^j) + f_{l^i}(\mathcal{N}^i) = f_{l^j}(\mathcal{N}^j)]\\ &\quad \leq \frac{1}{\mu}.\end{aligned}$$

Here we use the fourth property of Definition 2.

Subcase $N^i = N^j$: In this case, we have $\mathcal{N}^i = \mathcal{N}^j = \mathcal{N}$. If further $l^i = l^j$, then since the adversary does not repeat a query, we have $M^i \neq M^j$ and consequently, $\mathsf{Prob}[X^i = X^j] = 0$. So consider the case $l^i \neq l^j$. We have

$$\begin{aligned}&\mathsf{Prob}[X^i = X^j]\\ &\quad = \mathsf{Prob}[(M^i - M^j) + f_{l^i}(\mathcal{N}) = f_{l^j}(\mathcal{N})]\\ &\quad \leq \frac{1}{\mu}\end{aligned}$$

by the third property of Definition 2.

In each of the above cases, we have the probability of a collision to be upper-bounded by $1/\mu$. The domain contains at most $2q$ elements and hence the probability of a collision among the domain elements (whence bad is set to true) is at most $\binom{2q}{2}/\mu \leq 2q^2/\mu$. This completes the proof of the XE construction.

Proof of the XEX Construction: The proof of the XEX construction is more complicated, since the adversary is allowed to make decryption queries. The idea of the proof, however, is the same. On both encryption and decryption queries, the simulator returns random strings to the adversary and then adjusts the internal variables in a consistent manner. For the XE construction, the probability the adversary's advantage is bounded above by the probability of a collision in the $\mathrm{Domain}(\rho)$. For the XEX construction, the simulator needs to maintain both $\mathrm{Domain}(\rho)$ and $\mathrm{Range}(\rho)$ and the adversary's advantage is bounded above by the probability of a collision in either $\mathrm{Domain}(\rho)$ or $\mathrm{Range}(\rho)$. The collision analysis for $\mathrm{Range}(\rho)$ is a little different from that of $\mathrm{Domain}(\rho)$ as we point out later in the proof.

We assume that the adversary does not make any; *pointless queries*. In other words, the adversary does not query the decryption oracle with (N, C), if it had earlier obtained C as the output of an encryption query with (N, M). The converse is also assumed to hold, i.e., it does not query the encryption oracle with (N, M), if it had earlier obtained M as the output of a decryption query (N, C). Further, it does not repeat a query to either the encryption or the decryption oracles.

The hybrids in the case of the XEX construction are the following.

1) $p_1 = \mathsf{Prob}[K \xleftarrow{\$} \mathcal{K} : A^{\widetilde{E}_K(\cdot,\cdot),\widetilde{E}_K^{-1}(\cdot,\cdot)} \Rightarrow 1]$.
2) $p_2 = \mathsf{Prob}[\pi \xleftarrow{\$} \mathrm{Perm}(n) : A^{\widetilde{\pi}(\cdot,\cdot),\widetilde{\pi}^{-1}(\cdot,\cdot)} \Rightarrow 1]$.
3) $p_3 = \mathsf{Prob}[\rho_1, \rho_2 \xleftarrow{\$} \mathrm{Rand}(n) : A^{\widetilde{\rho_1}(\cdot,\cdot),\widetilde{\rho_2}(\cdot,\cdot)} \Rightarrow 1]$.
4) $p_4 = \mathsf{Prob}[\rho_1, \rho_2 \xleftarrow{\$} \mathrm{Rand}(\mathcal{T}, n) : A^{\rho_1(\cdot,\cdot),\rho_2(\cdot,\cdot)} \Rightarrow 1]$.
5) $p_5 = \mathsf{Prob}[\pi \xleftarrow{\$} \mathrm{Perm}(\mathcal{T}, n) : A^{\pi(\cdot,\cdot),\pi^{-1}(\cdot,\cdot)} \Rightarrow 1]$.

As before, we have to bound $p_1 - p_5 = (p_1 - p_2) + (p_2 - p_3) + (p_3 - p_4) + (p_4 - p_5)$. The bounds on $(p_2 - p_3)$ and $(p_4 - p_5)$ are the same as in the case of the XE construction while the bound on $(p_1 - p_2)$ is slightly different to take care of the fact that decryption queries are allowed.

1) $p_1 - p_2 \leq \mathbf{Adv}_E^{\pm\mathrm{prp}}(t', 2q)$.
2) $p_2 - p_3 \leq 2q^2/2^n$.
3) $p_4 - p_5 \leq 0.5q^2/2^n$.

Again, the main part of the proof is to bound $p_3 - p_4$.

Let us call the experiment associated with p_i to be Game i. In moving from Game 2 to Game 3, we are replacing the permutation π by the random function ρ_1 and the permutation π^{-1} by the random function ρ_2. In Game 3, the random functions ρ_1 and ρ_2 are used as in the XEX construction. In particular, ρ_1 is used whenever an encryption query is made and ρ_2 is used whenever a decryption query is made.

In Game 4, ρ_1 and ρ_2 are from the set $\mathrm{Rand}(\mathcal{T}, n)$. In other words, ρ_1 (also ρ_2) is a collection of random functions, one for each tweak in $\mathcal{T}$. Thus, for each (tweak, message) pair (N, M), the adversary expects to obtain a random bit string. We now present a unified description of Games 3 and 4. The jth query is either of the form (l^j, N^j, M^j) or (l^j, N^j, C^j) depending on whether the query is an encryption or a decryption query. The set Domain is the domain of ρ_1 and the range of ρ_2, while the set Range is the range of ρ_1 and the domain of ρ_2.

1. **If** $N^j = N^i$ for some $i < j$ **then** $\mathcal{N}^j = \mathcal{N}^i$;
2. **else**
3. $\quad \mathcal{N}^j \xleftarrow{\$} \{0,1\}^n$;
4. $\quad$ **If** $N^j \in$ Domain
   $\quad$ **then** bad $=$ true; $\boxed{\mathcal{N}^j = \rho_1(N^j);}$
5. $\rho_1(N^j) = \mathcal{N}^j$;
6. **If** the jth query is an encryption query **then**
7. $\quad X^j = M^j + f_{l^j}(\mathcal{N}^j)$;
7. $\quad C^j \xleftarrow{\$} \{0,1\}^n$; $Y^j = C^j + f_{l^j}(\mathcal{N}^j)$;
8. $\quad$ **if** $X^j \in$ Domain
   $\quad$ **then** bad $=$ true; $\boxed{Y^j = \rho_1(X^j);}$
9. $\quad \rho_1(X^j) = Y^j$; $C^j = Y^j - f_{l^j}(\mathcal{N}^j)$;
10. $\quad$ **return** C^j;
11. **if** the jth query is a decryption query **then**
12. $\quad Y^j = C^j + f_{l^j}(\mathcal{N}^j)$;
12. $\quad M^j \xleftarrow{\$} \{0,1\}^n$; $X^j = M^j + f_{l^j}(\mathcal{N}^j)$;
13. $\quad$ **if** $Y^j \in$ Range
   $\quad$ **then** bad $=$ true; $\boxed{X^j = \rho_2(Y^j);}$
14. $\quad \rho_2(Y^j) = X^j$; $M^j = X^j - f_{l^j}(\mathcal{N}^j)$;
15. $\quad$ **return** M^j.

Game 3 is the entire game, while Game 4 is obtained by removing the boxed entries. Both the games are the same unless bad is set. Hence, $p_3 - p_4$ is bounded above by the probability that bad is set. Our next task is to analyze this probability. In Game 4, the adversary obtains random strings on any input which it can generate by itself. Hence, we may assume the adversary to be nonadaptive. It submits a sequence of encryption and decryption queries and tries to set bad to be true. In fact, we will do more; we will allow the adversary to specify both the message and the ciphertext in all its queries and show that the probability of bad being true is still small. Thus, the adversaries queries are now of the form (l^j, N^j, M^j, C^j) for $j = 1, \ldots, q$.

The elements of the set Domain are of the form $N^j, M^j + f_{l^j}(\mathcal{N}^j)$ whereas the elements of the set Range are of the form $\mathcal{N}^j, C^j + f_{l^j}(\mathcal{N}^j)$. Note that the N^j values are never repeated in the domain. Further, now we have each M^j and C^j to be adversarily chosen and hence cannot assume any probability distribution on these quantities.

The domain set is similar to the case of the XE construction. Hence, the collision analysis of domain is similar to that of the

XE construction and we obtain that the probability of bad being set due to collision in domain is at most $2q^2/\mu$.

We now consider collisions in Range. There are three pairs of variables to consider giving rise to three cases below.

Case $(\mathcal{N}^i, \mathcal{N}^j)$: Clearly, $\mathsf{Prob}[\mathcal{N}^i = \mathcal{N}^j] = 1/2^n$ as both $\mathcal{N}^i$ and $\mathcal{N}^j$ are independent and randomly chosen quantities.

Case (Y^i, Y^j): Now

$$\begin{aligned}&\mathsf{Prob}[Y^i = Y^j]\\ &\quad = \mathsf{Prob}[C^i + f_{l^i}(\mathcal{N}^i) = C^j + f_{l^j}(\mathcal{N}^j)]\\ &\quad = \mathsf{Prob}[(C^i - C^j) + f_{l^i}(\mathcal{N}^i) = f_{l^j}(\mathcal{N}^j)].\end{aligned}$$

If $(l^i, N^i) = (l^j, N^j)$, then $C^i \neq C^j$ (as otherwise the adversary has made a pointless query) and $f_{i^l}(\mathcal{N}^l) = f_{i^j}(\mathcal{N}^j)$. In this case, $\mathsf{Prob}[Y^l = Y^j] = 0$.

If $(l^i, \mathsf{N}^i) \neq (l^j, \mathsf{N}^j)$, then as in the case of the XE construction, using Properties 1,3, and 4 of Definition 2, we have $\mathsf{Prob}[Y^i = Y^j] \leq 1/\mu$.

Case $(Y^i, \mathcal{N}^j)$: In this case, we need to use Property 2 of Definition 2. (This property was not required in the XE construction.)

$$\mathsf{Prob}[Y^i = \mathcal{N}^j] = \mathsf{Prob}[C^i + f_{l^i}(\mathcal{N}^i) = \mathcal{N}^j].$$

If $i \neq j$, then since $\mathcal{N}^i$ and $\mathcal{N}^j$ are independent random quantities and $f_{l^i}()$ is a bijective map, we have $\mathsf{Prob}[Y^i = \mathcal{N}^j] = 1/2^n$.

If $i = j$, then we have to consider $\mathsf{Prob}[C^i + f_{l^i}(\mathcal{N}^i) = \mathcal{N}^i]$, which by Property 2 of the masking sequence is bounded above by $1/\mu$.

Thus, in all cases, we have shown that the probability of a collision in between two range elements is bounded above by $1/\mu$. The range set has at most $2q$ elements and hence the probability of a range collision is at most $2q^2/\mu$. □

Note: In the above proof, we have used Property 2 of Definition 2, namely, $\mathsf{Prob}[f_l(\mathcal{N}) = \mathcal{N} + \alpha] \leq 1/\mu$, for any fixed string α and any randomly chosen string $\mathcal{N}$. If for any l, we have $f_l(\mathcal{N}) = \mathcal{N}$, then clearly the above condition cannot hold. Thus, in our instantiations of the masking functions, we have been careful to avoid $f_l(\mathcal{N}) = \mathcal{N}$ for any l. A similar condition is also highlighted in [16].

IV. Instantiating $\boldsymbol{R}$

The XE and the XEX constructions and the security proofs are obtained in the abstract setting of the ring $\boldsymbol{R}$ using a masking sequence. For efficient implementation, we have to specify $\boldsymbol{R}$ and also define appropriate masking sequences $f_1, \ldots, f_m$. The ring $\boldsymbol{R}$ can be endowed with two natural structures: The finite field GF(2^n) and the ring $\mathbb{Z}_{2^n}$. Note that once $\boldsymbol{R}$ and the f_i are specified, both the XE and the XEX constructions become concrete.

A. $\boldsymbol{R}$ as GF(2^n)

The set $\{0,1\}^n$ can be considered to be the set of all binary polynomials of degree less than n and made into the field GF(2^n) under multiplication modulo a fixed irreducible polynomial $\tau(x)$ of degree n. For our purpose, we will choose $\tau(x)$ to be a primitive polynomial.

Let G be an $n \times n$ matrix over GF(2) having $\tau(x)$ as its characteristic polynomial. We consider $\mathcal{N}$ to be an n-bit row vector. For $1 \leq i \leq 2^n - 2$, define

$$f_i(\mathcal{N}) = \mathcal{N}G^i. \tag{5}$$

Proposition 1: The sequence $f_1, f_2, \ldots, f_{2^n-2}$ defined by (5) is an $(n, 2^n - 2, 2^n)$ masking sequence (see Definition 2).

Proof:

(1) Note that $f_s(\mathcal{N}) = \mathcal{N}G^s$. Since G is invertible, the matrix G^s is also invertible. If $\mathcal{N}$ is uniformly distributed, the random variable $\mathcal{N}G^s$ is also uniformly distributed over $\{0,1\}^n$ and hence we have the desired result.

It is sufficient to show that the map $\mathcal{N} \mapsto \mathcal{N}(G^s \oplus I)$ is a bijection for any $s \geq 1$. In (3) below, we prove a more general result from which this follows.

(3) For $s \neq t$, define $\psi_{s,t}(\mathcal{N}) = f_s(\mathcal{N}) - f_t(\mathcal{N})$. We have to show that if $\mathcal{N}$ is uniformly distributed over $\{0,1\}^n$, then so is $\psi_{s,t}(\mathcal{N})$. This is achieved by showing that $\psi_{s,t}$ is a bijection. To prove Property 3 of Definition 2, we may assume $s, t \geq 1$. However, the bijective property holds even if one of s or t is 0 (but not both). So we will assume this in the argument below, which will also provide a proof of (2) above.

Let if possible $\psi_{s,t}(\mathcal{N}) = \psi_{s,t}(\mathcal{N}')$ for $\mathcal{N} \neq \mathcal{N}'$. Then

$$\begin{aligned}0 &= \psi_{s,t}(\mathcal{N}) - \psi_{s,t}(\mathcal{N}')\\ &= (f_s(\mathcal{N})) - f_t(\mathcal{N}))) - (f_s(\mathcal{N}')) - f_t(\mathcal{N}')))\\ &= \mathcal{N}(G^s - G^t) - \mathcal{N}'(G^s - G^t)\\ &= (\mathcal{N} - \mathcal{N}')(G^s - G^t).\end{aligned}$$

For any nonzero element $\beta \in \{0,1\}^n$, let $m_\beta(x)$ be the minimum degree polynomial such that $\beta m_\beta(G) = 0$. Then $m_\beta(x)$ divides any polynomial $p(x)$ for which $\beta p(G) = 0$. By the Cayley–Hamilton theorem, $\tau(G) = 0$ and hence $m_\beta(x) | \tau(x)$. By the irreducibility of $\tau(x)$, this implies $m_\beta(x) = \tau(x)$. Let $\beta = \mathcal{N} - \mathcal{N}'$ (under the usual identification of $\{0,1\}^n$ and the elements of GF(2^n)). Then $\tau(x) | (x^s - x^t)$. Without loss of generality assume $s > t$. Then $\tau(x) | x^t(x^{s-t} - 1)$. Since $\tau(x)$ does not divide x^t, we have $\tau(x) | (x^{s-t} - 1)$. It is well known that if $\tau(x)$ is a primitive polynomial of degree n, then it does not divide $x^i - 1$ for any $i < 2^n - 1$ (see for example [12]). Since $0 \leq t < s \leq 2^n - 2$, the fact that $\tau(x) | (x^{s-t} - 1)$ contradicts the above property of $\tau(x)$. Hence, we must have $\beta = 0$ and $\mathcal{N} = \mathcal{N}'$. This shows that $\psi_{s,t}()$ is an injection. Since it is a map from a finite set to itself, this implies that it is also a bijection. This completes the proof of (2).

(4) Since $\mathcal{N}$ and $\mathcal{N}'$ are independent random quantities and the maps $f_s()$ and $f_t()$ are bijective maps, it follows that $f_s(\mathcal{N})$ and $f_t(\mathcal{N}')$ are also independent and uniformly distributed random quantities and hence their difference is uniformly distributed over $\{0,1\}^n$. □

To specify the function $f_i()$, it is sufficient to specify the matrix G in (5). For the proof of Proposition 1, we only need $\tau(x)$ to be a primitive polynomial. However, a multiplication by a general G can be costly compared to one block cipher invocation.

On the other hand, if G has a simple form then it can be very fast to implement. We point out three efficient choices of G.

Let

$$\tau(x) = x^n \oplus t_{n-1}x^{n-1} \oplus \cdots t_1 x \oplus t_0.$$

Note that since $\tau(x)$ is primitive (and hence irreducible), the constant term t_0 must be 1. Define the matrix A_τ (having characteristic polynomial $\tau(x)$) as follows:

$$A_\tau = \begin{pmatrix} t_{n-1} & 1 & 0 & \dots & 0 & 0 \\ t_{n-2} & 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ t_1 & 0 & 0 & \dots & 0 & 1 \\ t_0 & 0 & 0 & \dots & 0 & 0 \end{pmatrix}.$$

Linear Feedback Shift Register (LFSR): We set $G = A_\tau$. The matrix A_τ (and hence G) can be implemented using a binary LFSR (see [12]).

Powering Construction: Let $a(x)$ be a polynomial of degree less than n. The map used in [19] is $a(x) \mapsto xa(x) \bmod \tau(x)$. Let $b(x) = xa(x) \bmod \tau(x)$. If the coefficients of $a(x)$ (resp., $b(x)$) are given by a vector $\mathcal{N}$ (resp., $\mathcal{N}'$), then $\mathcal{N}' = \mathcal{N}B_\tau$, where B_τ is the transpose of A_τ. Thus, in this case $G = B_\tau$.

Cellular Automata (CA): Another (perhaps less well known) linear map is a $90/150$ CA map. In this map, the matrix G is a tridiagonal matrix of the following form: $G_{i,j} = 1$, if $|i-j| = 1$; $G_{i,j} = 0$ or 1, if $i = j$; and $G_{i,j} = 0$ otherwise. The diagonal entries of G can be obtained from the polynomial $\tau(x)$ using a tridiagonalization procedure due to Tezuka and Fushimi [21].

Efficiency: All the above three methods are equally efficient to implement in both hardware and software. Thus, the LFSR- and the CA-based methods should be seen as *comparable* rather than better alternatives.

B. Word-Oriented LFSR

Suppose $n = n_1 n_2$ and consider GF(2^n) to be GF$((2^{n_1})^{n_2})$, i.e., as an extension field of degree n_2 over GF(2^{n_1}). Let $\tau_1(x)$ be an irreducible polynomial over GF(2) of degree n_1 and let GF(2^{n_1}) be represented using $\tau_1(x)$. Let $\tau_2(x)$ be a primitive polynomial over GF(2^{n_1}) (as represented by $\tau_1(x)$) of degree n_2. Then, it is well known that $\tau_2(x)$ does not divide $x^i - 1$ for $1 \le i \le 2^{n_1 n_2} - 2$ [12]. The field GF$((2^{n_1})^{n_2})$ is represented using $\tau_1(x)$ and $\tau_2(x)$. As is standard, when working modulo $\tau_2(x)$, we will identify polynomials over GF(2^{n_1}) of degree at most $n_2 - 1$ with vectors over GF(2^{n_1}) of dimension n_2.

Let G be an $n_2 \times n_2$ matrix with entries from GF(2^{n_1}). For $\mathcal{N} \in$ GF$((2^{n_1})^{n_2})$ define

$$f_i(\mathcal{N}) = \mathcal{N}G^i. \tag{6}$$

It is possible to show in a manner similar to that of Proposition 1, that the f_i's defined in (6) also form a masking sequence of functions.

The idea of using a tower of fields is not new. This idea is well known to the stream cipher community. Many stream ciphers have been proposed which use word-oriented LFSRs. For example, SNOW 1.0 uses the following parameters (see [6]): $n = 512$, $n_1 = 32$ and $n_2 = 16$, $\tau_1(x) = x^{32} \oplus x^{29} \oplus x^{20} \oplus x^{15} \oplus x^{10} \oplus x \oplus 1$, and $\tau_2(x) = x^{16} \oplus x^{13} \oplus x^7 \oplus \alpha^{-1}$, where $\tau_1(\alpha) = 0$. The polynomial $\tau_1(x)$ is irreducible over GF(2) and $\tau_2(x)$ is primitive over GF(2^{32}). These two polynomials define an LFSR of length 16 over GF(2^{32}). In software, the time for obtaining the next state of this LFSR is significantly faster than obtaining the next state of an LFSR of length 512 over GF(2).

This advantage in speed can also be utilized in the current context. We choose G to be a matrix which corresponds to one-step evolution of an LFSR whose connection polynomial is $\tau_2(x)$. Then the value of $f_i(\mathcal{N})$ can be obtained from $f_{i-1}(\mathcal{N})$ by evolving an LFSR over GF(2^{32}) once. To ensure that this is fast, we need to carefully choose the pair of polynomials $\tau_1(x)$ and $\tau_2(x)$ in a manner similar to that of SNOW 1.0 described above. The advantage is that, for software implementation, the corresponding word-oriented LFSR will be faster than any of the methods (powering, binary LFSR or CA) which work directly over GF(2).

It is also possible to realize GF(2^n) as a three-part extension. For example, GF(2^{32}) can be realized as a degree-four extension over GF(2^8). Such an idea has been used in SNOW 2.0 [6]. Again, for software implementation, such a word-oriented LFSR is faster than the powering method or binary LFSR. This shows that there are several possible ways of designing masking methods which are faster than the powering method used by Rogaway [19].

C. $\boldsymbol{R}$ as $\mathbb{Z}_{2^n}$

The set $\{0,1\}^n$ can be considered to be the set of all nonnegative integers less than 2^n and made into the ring $\mathbb{Z}_{2^n}$ by performing addition and multiplication modulo 2^n. Defining the masking sequence over $\mathbb{Z}_{2^n}$ is a bit tricky. This is because $\mathbb{Z}_{2^n}$ does not form a field. We first expand $\mathbb{Z}_{2^n}$ into a field.

Let $p > 2^n$ be a prime. Typically, we will choose the first such prime. We write $p = 2^n + \delta$. Then p is an $(n+1)$-bit integer and δ is usually very small compared to 2^n. Such primes are easy to find using standard mathematical software packages. For example, using PARI, we obtain the table of primes shown in Table I. These cover the most typical values of n used in practical applications. The set $\mathbb{Z}_p$ is a field under addition and multiplication modulo p and this field contains the integers $0, \dots, 2^n - 1$. For $i \ge 1$, we define

$$f_i(\mathcal{N}) = ((i+1) \times \mathcal{N} \bmod p) \bmod 2^n. \tag{7}$$

This idea of embedding the ring $\mathbb{Z}_{2^n}$ into a field $\mathbb{Z}_p$ has been earlier used in the literature [8], [22]. However, it has not been used in the context that we have used and to the best of our knowledge, the following result has not appeared earlier.

Proposition 2: The sequence $f_1, f_2, \dots, f_{2^n-2}$ defined by (7) is an $(n, 2^n - 2, 2^{n-1}/(\delta+1))$ masking sequence (see Definition 2).

Proof:

(1) First note that the map $\mathcal{N} \mapsto (i+1) \times \mathcal{N} \bmod p$ is an injection from $\mathbb{Z}_{2^n}$ to $\mathbb{Z}_p$. We can divide the image set of this map into two sets B_1 and B_2, where $B_1 \subseteq \{0, 1, \dots, 2^n - 1\}$ and $B_2 \subseteq \{2^n, \dots, 2^n + \delta - 1\}$. Now, when we perform the modulo 2^n operation, two elements of B_1 cannot collide and

TABLE I
PRIMES p OF THE FORM $2^n + \delta$ WITH THE SMALLEST POSSIBLE δ

n	80	96	128	160	192	256
p	$2^{80}+13$	$2^{96}+61$	$2^{128}+51$	$2^{160}+7$	$2^{192}+133$	$2^{256}+297$

neither can two elements of B_2 collide. The only possibility of collision is between an element of B_1 and an element of B_2. Thus, any element of $\mathbb{Z}_{2^n}$ has either zero, one, or two pre-images under the map $f_s()$. Since $\mathcal{N}$ is chosen uniformly from $\mathbb{Z}_{2^n}$, we have

$$\mathsf{Prob}[f_s(\mathcal{N}) = \alpha] \leq \frac{2}{2^n} = \frac{1}{2^{n-1}}.$$

(2) Follows from the more general argument given for (3) below.

(3) We are required to prove the result for $i, j \geq 1$ and $i \neq j$. However, the argument given below also holds for $i, j \geq 0$, though still with $i \neq j$. Strictly speaking f_j is not defined for $j = 0$. However, we extend to the case $j = 0$ in the natural manner by having $f_0(\mathcal{N}) = \mathcal{N}$. Then substituting $j = 0$ in the argument below gives the proof of (2) above.

For $i \neq j$, define

$$\begin{aligned}\psi_{i,j}(\mathcal{N}) &= f_i(\mathcal{N}) - f_j(\mathcal{N}) \\ &= (((i+1)\mathcal{N} \bmod p) \bmod 2^n \\ &\quad - ((j+1)\mathcal{N} \bmod p) \bmod 2^n) \bmod 2^n.\end{aligned}$$

We would like to count the maximum number of pre-images that an element in $\mathbb{Z}_{2^n}$ can have under $\psi_{i,j}$. There are too many modulo operations in the definition of $\psi_{i,j}$. This makes it difficult to analyze the function. We make things simpler by identifying two sets, where we can ignore some of the modulo operations. Define

$$\begin{aligned}A_1 = \{&\mathcal{N} \in \mathbb{Z}_{2^n} : (i+1)\mathcal{N} \bmod p < 2^n, \\ &(j+1)\mathcal{N} \bmod p < 2^n, \\ &0 \leq (i+1)\mathcal{N} \bmod p \\ &- (j+1)\mathcal{N} \bmod p < 2^n\}; \\ A_2 = \{&\mathcal{N} \in \mathbb{Z}_{2^n} : (i+1)\mathcal{N} \bmod p < 2^n, \\ &(j+1)\mathcal{N} \bmod p < 2^n, \\ &-2^n + 1 \leq (i+1)\mathcal{N} \bmod p \\ &- (j+1)\mathcal{N} \bmod p < 0\}; \\ A = A_1 \cup A_2; \\ \overline{A} = \mathbb{Z}_{2^n} \setminus A.\end{aligned}$$

Claim: If we restrict the domain of $\psi_{i,j}$ to A_1 or A_2, then we obtain an injective map.

Proof of Claim: We prove the claim for A_1. The proof for A_2 is similar. Let $\mathcal{N}_1, \mathcal{N}_2 \in A_1$. Then we can write

$$\begin{aligned}(i+1)\mathcal{N}_1 &= q_{i,1}p + r_{i,1}; \quad (j+1)\mathcal{N}_1 = q_{j,1}p + r_{j,1} \\ (i+1)\mathcal{N}_2 &= q_{i,2}p + r_{i,2}; \quad (j+1)\mathcal{N}_2 = q_{j,2}p + r_{j,2}\end{aligned}$$

where $0 \leq r_{i,1}, r_{j,1}, r_{i,2}, r_{j,2} < 2^n$. Also, $\psi_{i,j}(\mathcal{N}_1) = r_{i,1} - r_{j,1} \geq 0$ and $\psi_{i,j}(\mathcal{N}_2) = r_{i,2} - r_{j,2} \geq 0$. Let if possible, $\psi_{i,j}(\mathcal{N}_1) = \psi_{i,j}(\mathcal{N}_2)$ for $\mathcal{N}_1 \neq \mathcal{N}_2$. Then we have $r_{i,1} - r_{j,1} = r_{i,2} - r_{j,2}$ and so

$$p(q_1 - q_2) = (i-j)(\mathcal{N}_1 - \mathcal{N}_2)$$

where $q_1 = q_{i,1} - q_{j,1}$ and $q_2 = q_{i,2} - q_{j,2}$. Thus, p divides $(i-j)(\mathcal{N}_1 - \mathcal{N}_2)$ and hence, $p|(i-j)$ or $p|(\mathcal{N}_1 - \mathcal{N}_2)$. Since $0 \leq i, j, \mathcal{N}_1, \mathcal{N}_2 < 2^n$ and $p > 2^n$, this is not possible. This completes the proof of the claim.

It is possible that an element from A_1 and an element from A_2 have the same image under $\psi_{i,j}$. Thus, the number of pre-images of any element in $\mathbb{Z}_{2^n}$ under $\psi_{i,j}$ is at most $2 + |\overline{A}|$. We now upper-bound $|\overline{A}|$.

Note that

$$\begin{aligned}A = \{&\mathcal{N} \in \mathbb{Z}_{2^n} : (i+1)\mathcal{N} \bmod p < 2^n \text{ and} \\ &(j+1)\mathcal{N} \bmod p < 2^n\}\end{aligned}$$

and hence

$$\begin{aligned}\overline{A} &= \{\mathcal{N} \in \mathbb{Z}_{2^n} : 2^n \leq (i+1)\mathcal{N} \bmod p < p \text{ or} \\ &\qquad 2^n \leq (j+1)\mathcal{N} \bmod p < p\} \\ &= \{\mathcal{N} \in \mathbb{Z}_{2^n} : 2^n \leq (i+1)\mathcal{N} \bmod p < p\} \\ &\quad \bigcup \{\mathcal{N} \in \mathbb{Z}_{2^n} : 2^n \leq (j+1)\mathcal{N} \bmod p < p\}.\end{aligned}$$

Thus

$$\begin{aligned}|\overline{A}| \leq &|\{\mathcal{N} \in \mathbb{Z}_{2^n} : 2^n \leq (i+1)\mathcal{N} \bmod p < p\}| \\ &+ |\{\mathcal{N} \in \mathbb{Z}_{2^n} : 2^n \leq (j+1)\mathcal{N} \bmod p < p\}|.\end{aligned}$$

The map $(i+1) \mapsto (i+1)\mathcal{N} \bmod p$ from $\mathbb{Z}_{2^n}$ to $\mathbb{Z}_p$ is an injective map. Hence

$$|\{\mathcal{N} \in \mathbb{Z}_{2^n} : 2^n \leq (i+1)\mathcal{N} \bmod p < p\}| \leq \delta$$

and

$$|\{\mathcal{N} \in \mathbb{Z}_{2^n} : 2^n \leq (j+1)\mathcal{N} \bmod p < p\}| \leq \delta$$

where $\delta = p - 2^n$. Thus, $|\overline{A}| \leq 2\delta$. This shows that the number of pre-images of any element in $\mathbb{Z}_{2^n}$ under $\psi_{i,j}$ is at most $2(\delta + 1)$. Since the input $\mathcal{N}$ of $\psi_{i,j}$ is chosen uniformly at random from $\mathbb{Z}_{2^n}$, the probability of occurrence of any element in the range of $\psi_{i,j}$ is at most $(\delta+1)/2^{n-1}$. This completes the proof of (2).

(4) Let $X = f_s(\mathcal{N})$ and $Y = f_t(\mathcal{N}')$ be the dependent random variables defined from $\mathcal{N}$ and $\mathcal{N}'$, respectively. Then X and Y are independent random variables having identical distribution. From the proof of (1) they take values from the set $\mathbb{Z}_{2^n}$

with probabilities $0, 1/2^n$, and $2/2^n$. The event $X-Y=\alpha$ can be decomposed into the disjoint events ($X = a+\alpha \bmod 2^n$ and $Y = a$) for all $a \in \mathbb{Z}_{2^n}$. Using the independence of X and Y, we have

$$\begin{aligned}&\mathsf{Prob}[X-Y=\alpha]\\&\quad=\sum\nolimits_{a\in\mathbb{Z}_{2^n}}\mathsf{Prob}[X=a+\alpha \text{ and } Y=a]\\&\quad=\sum\nolimits_{a\in\mathbb{Z}_{2^n}}\mathsf{Prob}[X=a+\alpha]\mathsf{Prob}[Y=a]\\&\quad\le\sum\nolimits_{a\in\mathbb{Z}_{2^n}}\frac{2}{2^n}\times\frac{2}{2^n}\\&\quad=\frac{1}{2^{n-2}}.\end{aligned}$$

This completes the proof of (3). □

The security bound (obtained from the value of μ) of Proposition 2 ($\mu = 2^{n-1}/(\delta+1)$) is a little weaker than that of Proposition 1 ($\mu = 2^n$). This results from the fact that we have to enlarge the ring $\mathbb{Z}_{2^n}$ into the field $\mathbb{Z}_p$. On the other hand, the slight decrease in the security bound is immaterial from a practical point of view.

Efficiency: We will be computing the f_i's one after the other. Note that both $\mathcal{N}$ and $f_i(\mathcal{N})$ are in $\mathbb{Z}_{2^n}$. We first initialize a variable X to $\mathcal{N}$. The value of X will be evaluated modulo p, i.e., X can take any value between 0 and $p-1$. If we denote the ith value of X by X_i, then $X_i = (i+1)\mathcal{N} \bmod p$. To compute $f_{i+1}(\mathcal{N})$, we add $\mathcal{N}$ and X modulo p and take the last n bits of the result to be the value of $f_{i+1}(\mathcal{N})$. This requires only one multiprecision integer addition and at most one subtraction. Thus, software implementation of $f_i(\mathcal{N})$ will be efficient.

The exact comparative efficiency between the GF(2^n)-based method and the $\mathbb{Z}_{2^n}$-based method will, to some extent, depend on the implementation details. We note though, that both the methods will be quite efficient and the difference in speed may not be significant, especially in comparison to one block cipher invocation. Again, we do not claim to provide a more efficient alternative to the powering method of Rogaway; our claim is to provide another similarly efficient alternative to the powering method.

V. Authenticated Encryption

An authenticated encryption protocol consists of an encryption and a decryption algorithm. The encryption algorithm takes as input (the key and) a nonce and message and produces as output a ciphertext which consists of an encryption of the message and a tag. The decryption algorithm takes as input (the key and) a nonce and a ciphertext and produces either the corresponding message or returns invalid. Rogaway [19] obtains an AE protocol in two steps.

1) Given a TBC $\widetilde{F} : \mathcal{K} \times \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$ where $\mathcal{T} = \{0,1\}^n \times \{1,\ldots,2^{n/2}\} \times \{0,1\}$ and an integer $\tau \in [0 \ldots n]$, Rogaway provides a construction of an AE protocol.
2) The TBC $\widetilde{F}$ is instantiated in [19] using a TBC $\widetilde{E}$ obtained by the powering construction over GF(2^n) from XEX.

Rogaway's AE construction from the TBC $\widetilde{F}$ also holds in the more general setting of $\boldsymbol{R}$. Our contribution is essentially to the second step above. Recall that we have provided the construction of a TBC

$$\widetilde{E} : \mathcal{K} \times (\{0,1\}^n \times \{1,2,\ldots,2^n-2\}) \times \{0,1\}^n \to \{0,1\}^n.$$

Using this, we have to instantiate the $\widetilde{F}$. This means that we have to map the set $\{1,2,\ldots,2^{n/2}\}\times\{0,1\}$ to the set $\{1,2,\ldots,2^n-2\}$. Let

$$\phi : \{1,2,\ldots,2^{n/2}\} \times \{0,1\} \to \{1,2,\ldots,2^n-2\}$$

be this map. The requirement on ϕ is that it should be an injective map. (In [19], this requirement is called unique representability in the context of the powering construction over GF(2^n).)

Our contribution to the AE protocol of Rogaway [19] is in the different definitions of ϕ. We show two ways of defining ϕ. The first method, which we call linear separation, is based on Rogaway's method. The second method, which we call interleaved separation, is new to this work.

Let $\Delta_{i,b}(\mathcal{N}) = f_{\phi(i,b)}(\mathcal{N})$. Fig. 1 shows the AE protocol of [19] written using the Δ's. The statement on the security of the protocol is given in Section V-D.

In Fig. 1, the tweaks $\Delta_{1,0}(\mathcal{N}), \Delta_{2,0}(\mathcal{N}), \ldots, \Delta_{m,0}(\mathcal{N})$ are used to encrypt the m message blocks and the tweak $\Delta_{m,1}(\mathcal{N})$ is used to encrypt the tag. Thus, for the purpose of efficiency, the following two tasks must be efficient.

Task 1: Compute $\Delta_{i+1,0}(\mathcal{N})$ from $\Delta_{i,0}(\mathcal{N})$.
Task 2: Compute $\Delta_{m,1}(\mathcal{N})$ from $\Delta_{m,0}(\mathcal{N})$.

We next show two different methods for defining ϕ and efficiency of the two tasks in both the methods.

A. Linear Separation

Let L be an integer such that $2^{n/2} \le L < L+2^{n/2} \le 2^n-2$. Define

$$\phi(i,b) = i + Lb. \tag{8}$$

The injectivity of ϕ is easily verified. In Fig. 1, the use of (8) implies the following.

- For the message blocks we use masks $f_1(\mathcal{N}), f_2(\mathcal{N}), \ldots, f_m(\mathcal{N})$.
- For the tag we use the mask $f_{m+L}(\mathcal{N})$.

We now consider the two tasks.

1) *Task 1.:* Recall that earlier it has been shown that it is easy to obtain $f_{i+1}(\mathcal{N})$ from $f_i(\mathcal{N})$ for both the cases when $\boldsymbol{R}$ is realized as GF(2^n) or as $\mathbb{Z}_{2^n}$.
2) *Task 2.:* We show the efficiency of this task separately for the realization of $\boldsymbol{R}$ as GF(2^n) and $\mathbb{Z}_{2^n}$.

$\boldsymbol{R}$ as GF(2^n): In this case, the technique of [19] is applicable. Let L be the discrete log of $(x+1)$ in GF(2^n) realized using the primitive polynomial $\tau(x)$. (For $n = 64, 128$, the corresponding values of L are computed in [19] and satisfy the condition on L.) Thus, $x^L \equiv x \oplus 1 \bmod \tau(x)$ and so $x^L \oplus x \oplus 1 = q(x)\tau(x)$ for some polynomial $q(x)$.

Recall that the matrix G used to define the masking sequence of functions has $\tau(x)$ as its characteristic polynomial. Using the Cayley–Hamilton theorem, it follows that $\tau(G) = 0$ and hence

Algorithm Encrypt(K, N, M)	Algorithm Decrypt$(K, N, (C, \mathsf{tag}))$
Partition M into $M[1] \cdots M[m]$;	Partition C into $C[1] \cdots C[m]$;
$\mathcal{N} = E_K(N)$;	$\mathcal{N} = E_K(N)$;
$\mathsf{sum} = 0^n$;	$\mathsf{sum} = 0^n$;
for $i = 1$ to $m-1$ **do**	**for** $i = 1$ to $m-1$ **do**
$\mathsf{mask} = \Delta_{i,0}(\mathcal{N})$;	$\mathsf{mask} = \Delta_{i,0}(\mathcal{N})$;
$C[i] = E_K(M[i] + \mathsf{mask}) - \mathsf{mask}$;	$M[i] = E_K^{-1}(C[i] + \mathsf{mask}) - \mathsf{mask}$;
$\mathsf{sum} = \mathsf{sum} + M[i]$;	$\mathsf{sum} = \mathsf{sum} + M[i]$;
end for;	**end for**;
$\mathsf{mask} = \Delta_{m,0}(\mathcal{N})$;	$\mathsf{mask} = \Delta_{m,0}(\mathcal{N})$;
$\mathsf{Pad} = E_K(\mathsf{len}(M[m]) + \mathsf{mask}) - \mathsf{mask}$;	$\mathsf{Pad} = E_K(\mathsf{len}(C[m]) + \mathsf{mask}) - \mathsf{mask}$;
$C[m] = M[m] + \mathsf{Pad}$;	$M[m] = C[m] + \mathsf{Pad}$;
$C = C[1] \cdots C[m]$;	$M = M[1] \cdots M[m]$;
$\mathsf{sum} = \mathsf{sum} + (C[m]0^*) + \mathsf{Pad}$;	$\mathsf{sum} = \mathsf{sum} + (C[m]0^*) + \mathsf{Pad}$;
$\mathsf{mask} = \Delta_{m,1}(\mathcal{N})$;	$\mathsf{mask} = \Delta_{m,1}(\mathcal{N})$;
$T = E_K(\mathsf{sum} + \mathsf{mask}) - \mathsf{mask}$;	$T = E_K(\mathsf{sum} + \mathsf{mask}) - \mathsf{mask}$;
set tag to the first τ bits of T;	**set** tag$'$ to the first τ bits of T;
return (C, tag).	**if** tag $=$ tag$'$ **then return** M **else return** INVALID.

Fig. 1. Encryption and decryption algorithms of an AE protocol over $\boldsymbol{R}$. The encryption algorithm takes as input (K, N, M) where K is the key, N is the nonce, and M is the message. It produces as output a pair (C, tag). The decryption algorithm takes as input $(K, N, (C, \mathsf{tag}))$, where K and N are key and nonce, respectively, and (C, tag) is the ciphertext and tag pair. It produces as output either the message M or says that the pair (C, tag) is invalid. Here $\Delta_{i,b}(\mathcal{N}) = f(\mathcal{N})$.

$G^L \oplus G \oplus I_n = q(G)\tau(G) = 0$. Thus, for any $\mathcal{N} \in \{0,1\}^n$, we have $\mathcal{N}G^L = \mathcal{N}(G \oplus I_n)$. Hence, we have

$$\begin{aligned} f_{m+L}(\mathcal{N}) &= \mathcal{N}G^{m+L} \\ &= (\mathcal{N}G^m)G^L \\ &= f_m(\mathcal{N})G^L \\ &= f_m(\mathcal{N})(G \oplus I_n). \end{aligned}$$

In other words, given $X = f_m(\mathcal{N})$ we compute $Y = f_{m+L}(\mathcal{N})$ in the following manner: Compute $X_1 = XG$ and set $Y = X \oplus X_1$. Computation of XG requires one application of G, which is efficient in all the three cases—LFSR, powering, and CA.

Word-Oriented LFSR: As discussed earlier, such LFSRs are very efficient to implement. In particular, they are faster than the powering method of Rogaway [19]. To use word-oriented LFSRs with the technique of linear separation, we need to obtain $\tau_1(x)$ and $\tau_2(x)$ (see Section IV-B) such that the discrete log of $(x+1)$ modulo $\tau_2(x)$ is "large." We can then choose L to be equal to this discrete log and the discussion given above will hold. We have not tried to obtain a "suitable" pair $(\tau_1(x), \tau_2(x))$ but we expect that there are many such pairs for $n_1 = 32$ and $n_2 = 4$. For any such pair, the masking part of the resulting AE mode of operation will be significantly faster than the algorithm OCB given by Rogaway [19].

$\boldsymbol{R}$ as $\mathbb{Z}_{2^n}$: We choose $L = 2^{n/2}$. Recall that in this case $X_i = (i+1)\mathcal{N} \bmod p$ and $f_i(\mathcal{N}) = X_i \bmod 2^n$. Then $f_{m+L}(\mathcal{N}) = (X_m + 2^{n/2}\mathcal{N} \bmod p) \bmod 2^n$ and can be computed from X_m using one modulo p multiplication.

B. Interleaved Separation

In this case, we define $\phi(i,b)$ in the following manner:

$$\phi(i,b) = 2i + b. \tag{9}$$

The injectivity of ϕ is easily verified. In Fig. 1, the use of this map implies the following.

- For the message blocks we use masks $f_2(\mathcal{N}), f_4(\mathcal{N}), f_6(\mathcal{N}), \ldots, f_{2m}(\mathcal{N})$.
- For the tag we use the mask $f_{2m+1}(\mathcal{N})$.

The advantage of this method over the linear separation technique is that it does not require the computation of a discrete log during the design stage when $\boldsymbol{R}$ is instantiated as GF(2^n). The computation of Tasks 1 and 2 are quite efficient though it is a little slower than the linear separation method. Simple implementation tricks can speed up the mask computation.

C. Comparison of the AE Protocols

At a top level, we have four single-pass AE protocols. There are two options for instantiating the ring $\boldsymbol{R}$ (either as GF(2^n) or as $\mathbb{Z}_{2^n}$) and two options for constructing the protocol (either using linear or interleaved separation). This gives rise to a total of four different possibilities. Further, when we realize $\boldsymbol{R}$ as GF(2^n) there are different possibilities for implementing G. We have indicated four—as an LFSR; using the powering construction; as a CA; or using a word-oriented LFSR. Out of all these AE protocols, the masking method using word-oriented LFSR and linear separation will be the fastest. We mention that we have not implemented any of the AE protocols mentioned in this paper. Such work, we believe, is outside the scope of the current paper. A careful implementation of the different candidate algorithms and fine tuning the parameters is a possible future work.

The AE protocol in [19] corresponds to the instantiation of $\boldsymbol{R}$ as GF(2^n); G as the powering construction and using the technique of linear separation. Clearly, this is a special case of the suite of AE protocols that we have developed. There are other single-pass protocols which do not fall within the general description that we have developed. In particular, the protocols of Gligor and Donescu [7], Jutla [10], and the earlier protocol of Rogaway [20] are not covered by our general description.

Efficiency of Linear Versus Interleaved Separation: In the linear separation technique, the masks $f_1(\mathcal{N}), f_2(\mathcal{N}), \ldots, f_m(\mathcal{N})$ are used for the message blocks, where as in the interleaved separation technique, the masks $f_2(\mathcal{N}), f_4(\mathcal{N}), \ldots, f_{2m}(\mathcal{N})$ are used for the message blocks. Thus, it may seem that the interleaved separation technique results in a much

slower AE protocol compared to the linear separation technique. We argue that this is not the case. In particular, when $\boldsymbol{R}$ is realized as $\mathbb{Z}_{2^n}$, both methods have same efficiency. When $\boldsymbol{R}$ is realized as GF(2^n), the interleaved method can be slightly slower but not significantly so, since the difference in the time for generating the m masks is negligible in comparison to the time required for the m block cipher invocations.

Suppose $\boldsymbol{R}$ is realized as $\mathbb{Z}_{2^n}$. Then $f_i(\mathcal{N}) = ((i+1) \times \mathcal{N} \bmod p) \bmod 2^n$. As mentioned earlier, we will be using a variable X whose ith value is $X_i = (i+1) \times \mathcal{N} \bmod p$. Then $X_{i+1} = X_i + \mathcal{N} \bmod p$ and $X_{i+2} = X_i + 2\mathcal{N} \bmod p$. So, if we compute $2\mathcal{N} \bmod p$ once at the beginning, then computing X_{i+2} from X_i is as efficient as computing X_{i+1} from X_i. This shows that for $\boldsymbol{R}$ as $\mathbb{Z}_{2^n}$, both linear and interleaved separation techniques have similar efficiency.

Now consider the case when $\boldsymbol{R}$ is realized as GF(2^n). There are four possibilities—binary LFSR; powering; CA; and word-oriented LFSR. For concreteness, let us consider the powering method. We have to compare the time for computing $x^2\mathcal{N} \bmod \tau(x)$ (in the interleaved separation method) with that of computing $x\mathcal{N} \bmod \tau(x)$ (in the linear separation method). The first operation takes more time than the second operation, though not necessarily twice as much time.

More importantly, however, both these operations should be seen in the context of an AE mode of operation. Let t_1 and t_2, respectively, be the times for these two operations and let t be the time for one block cipher call. Then, the interleaved separation technique requires $t + t_1$ time per block, where as the linear separation technique requires $t + t_2$ time per block. We argue that the difference $t_1 - t_2$ is negligible with respect to t. Our rationale is that a block cipher performs much more operations than a few shifts and XOR's needed to implement a modulo multiplication by x. For example, AES-128 performs 160 table lookups in addition to other operations. A careful implementation (which we have not done) of the two methods can settle this point.

There is another aspect that we would like to point out. The previous construction of Rogaway [19] works directly over GF(2) and uses the technique of linear separation. To this we would like to compare the use of word-oriented LFSR using the interleaved method. Let t_3 be the time to generate the next mask in the first method (i.e., the time to compute one multiplication by x) and t_4 be the time to compute the next mask in the second method (i.e., the time to compute two evolutions of a word-oriented LFSR). The experience from design of stream ciphers suggests that for software implementation, t_4 is less than t_3. In other words, the interleaved technique with word-oriented LFSR will be faster than the linear separation technique with the powering method (as used by Rogaway). Again, a careful implementation, which we have not done, will settle this point.

An Easily Reconfigurable Family: Consider the situation when $\boldsymbol{R}$ is implemented as GF(2^n). In this case, the field representing polynomial $\tau(x)$ can be viewed as parameterizing the mode of operation. In other words, the construction can be viewed as a family of modes of operations, indexed by the set of primitive polynomials over GF(2^n). All constructions in the family have the same efficiency and the same security guarantee. Choosing $\tau(x)$ selects a particular member from the family.

The number of primitive polynomials over GF(2) of degree n is equal to $\mathsf{Tot}(2^n - 1)/n$, where $\mathsf{Tot}(i)$ is the Euler totient which is the number of positive integers less than i and coprime to i. The quantity $\mathsf{Tot}(2^n - 1)/n$ is fairly large (for $n = 128$, this value is around 2^{119}) and so we have a rather large family of modes of operations.

Now, suppose we use Rogaway's construction, i.e., the powering method with linear separation. In this case, whenever $\tau(x)$ is changed, we need to verify that the discrete log of $(x+1)$ with respect to the new $\tau(x)$ is "large" as otherwise the security proof might not hold. Thus, each change of $\tau(x)$ requires a discrete log computation.

In contrast, consider the interleaved separation technique. This does not require any discrete log computation. Hence, we can choose any primitive polynomial $\tau(x)$ and immediately obtain a construction. In both software and hardware implementations, the primitive polynomial can be provided as a parameter—in software as part of a header file and in hardware as a register. Choosing a new primitive polynomial and changing this parameter is quite simple. This provides an easily reconfigurable design. As discussed in Section I-D, this feature may have a practical appeal to developers of cryptographic products.

D. Security of AE Protocols

The security of an authenticated encryption protocol consists of two parts—privacy and authenticity. The adversary is given access to the encryption oracle and is assumed to be nonce respecting, i.e., it does not repeat a nonce in its queries to the oracle. Following Rogaway [19], the privacy of a encryption scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ against a nonce respecting adversary A is defined in the sense of "indistinguishability from random strings" in the following manner:

$$\mathbf{Adv}^{\mathrm{priv}}_{\Pi}(A) = \mathsf{Prob}[K \xleftarrow{\$} \mathcal{K} : A^{\mathcal{E}_K(\cdot,\cdot)} \Rightarrow 1] - \mathsf{Prob}[A^{\$(\cdot,\cdot)} \Rightarrow 1]$$

where $\$(\cdot,\cdot)$ is an oracle that takes (N, M) as input and returns $|M|$ many random bits as output. For defining authenticity, the adversary is said to successfully *forge* if it outputs a pair $(N, (C, \mathsf{tag}))$ which is valid and (C, tag) was not the result of any prior (N, M) query. Formally

$$\mathbf{Adv}^{\mathrm{auth}}_{\Pi}(A) = \mathsf{Prob}[K \xleftarrow{\$} \mathcal{K} : A^{\mathcal{E}(\cdot,\cdot)} \text{ forges}].$$

The result on the security of the AE protocol of Fig. 1 is stated below and is a minor modification of [17, Corollary 14].

Theorem 2: Let $\mathrm{AE}[\widetilde{E}, \tau]$ be constructed as in Fig. 1. Let $\widetilde{E}$ be instantiated by a block cipher $E : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$. Then

1) $\mathbf{Adv}^{\mathrm{priv}}_{\mathrm{AE}[E,\tau]}(t, \sigma_n) \le \mathbf{Adv}^{\mathrm{prp}}_{E}(t', \sigma_n) + \frac{5q^2}{2^{n+1}} + \frac{4q^2}{\mu}$
2) $\mathbf{Adv}^{\mathrm{auth}}_{\mathrm{AE}[E,\tau]}(t, 2\sigma_n) \le \mathbf{Adv}^{\pm\mathrm{prp}}_{E}(t', 2\sigma_n) + \frac{2^n - \tau}{(2^n - 1)} + \frac{5q^2}{2^{n+1}} + \frac{4q^2}{\mu}$

where $t' = t + cn\sigma_n$ for some absolute constant c; $\mu = 2^n$ if $\boldsymbol{R}$ is realized as GF(2^n), and $\mu = 2^{n-1}/(\delta+1)$ with $\delta = p - 2^n$ if $\boldsymbol{R}$ is realized as $\mathbb{Z}_{2^n}$.

VI. MAC Construction

A MAC protocol consists of two algorithms. The tag-generation algorithm takes as input (a key and) a message and produces as output a tag. The verification algorithm takes as input (a key and) a message–tag pair and returns either true (if the pair is valid) or false (if it is invalid).

In [19], the TBC obtained from the XE construction is used to construct a MAC protocol. In fact, a more general construction of a tweakable PRF is presented in [19]. A tweakable PRF is a map $\widetilde{\boldsymbol{F}} : \mathcal{K} \times \mathcal{V} \times \mathcal{M} \to \{0,1\}^\tau$ where $\mathcal{K} \neq \emptyset$ is the key space, $\mathcal{V} \neq \emptyset$ is the tweak space, $\emptyset \neq \mathcal{M} \subset \{0,1\}^*$ is the message space and $\tau \geq 1$.

Under the assumption (implicit in [19]) that at most B blocks are permissible in a single message, the general construction is described using a TBC

$$\widetilde{F}: \mathcal{K} \times (\{1,\ldots,B\} \times \{0,1,2\} \times \{0,\ldots,\mathsf{V}\}) \times \{0,1\}^n \to \{0,1\}^n.$$

The set $\{0,\ldots,\mathsf{V}\}$, where V is a small positive integer (≤ 7), is considered to be a tweak to the PRF (and hence MAC) algorithm itself.

For each tweak (i,j,v), the MAC algorithm associates a mask $\Delta_{i,j,v}$. The algorithm of [19] written in terms of the $\Delta_{i,j,v}$'s is shown in Fig. 2. The security statement is given in Section VI-C. The first $(m-1)$ message blocks are masked using $\Delta_{1,0,v}, \Delta_{2,0,v}, \ldots, \Delta_{m-1,0,v}$ and the last encryption is masked using $\Delta_{m,1,v}$ or $\Delta_{m,2,v}$ according as whether the last block is full or partial.

The TBC $\widetilde{F}$ is instantiated by the TBC $\widetilde{E}$ which, in turn, is instantiated by the block cipher E. This chain of instantiations can be written as follows:

$$\begin{aligned}\widetilde{F}^{i,j,v}(M) &= \widetilde{E}^{0^n,\phi(i,j,v)}(M)\\ &= E_K(M + \Delta_{i,j,v})\\ &= E_K(M + f_{\phi(i,j,v)}(\mathcal{N}))\end{aligned}$$

where $\mathcal{N} = E_K(0^n)$ and

$$\phi : \{1,\ldots,B\} \times \{0,1,2\} \times \{0,\ldots,\mathsf{V}\} \to \{1,\ldots,2^n-2\}$$

is an injective map. As in the case of AE, we identify two techniques for defining the map ϕ.

A. Linear Separation

Let L_1 and L_2 be two positive integers satisfying the following two conditions.

- $B + 2L_1 + \mathsf{V}L_2 \leq 2^n - 2$.
- $|L_1 j + L_2 v| > B$ for $-2 \leq j \leq 2$ and $-\mathsf{V} \leq v \leq \mathsf{V}$.

Define

$$\phi(i,j,v) = i + L_1 j + L_2 v. \tag{10}$$

Lemma 1: The map ϕ defined in (10) is an injection.

Proof: Let, if possible, $(i_1,j_1,v_1) \neq (i_2,j_2,v_2)$ and $\phi(i_1,j_1,v_1) = \phi(i_2,j_2,v_2)$. Then we have $i_1 - i_2 = L_1(j_2 - j_1) + L_2(v_2 - v_1)$, where $-B \leq i_1 - i_2 \leq B$, $-2 \leq j_2 - j_1 \leq 2$, and $-\mathsf{V} \leq v_2 - v_1 \leq \mathsf{V}$. From the given condition on L_1 and L_2, the minimum value of $|L_1(j_2 - j_1) + L_2(v_2 - v_1)|$ is greater than B while $|i_1 - i_2| \leq B$. Hence, if any one of $(j_2 - j_1)$ or $(v_2 - v_1)$ is not equal to zero, then $i_1 - i_2 = L_1(j_2 - j_1) + L_2(v_2 - v_1)$ cannot hold. If both are zeros, then $i_1 = i_2$ and we have $(i_1,j_1,v_1) = (i_2,j_2,v_2)$. This shows that ϕ is an injection. □

We now consider the two possibilities for $\boldsymbol{R}$.

1) $\boldsymbol{R}$ **as** GF(2^n): The values of L_1 and L_2 are, respectively, the discrete logs of $(x+1)$ and (x^2+x+1) with respect to the lexicographically first primitive polynomial $\tau(x)$ of degree n over GF(2). These values have been computed in [19] for $n = 128$ and $n = 64$ and satisfy the required condition for $B = 2^{n/2}$.

$$\begin{aligned}f_{i+jL_1+vL_2}(\mathcal{N}) &= \mathcal{N}G^{i+jL_1+vL_2}\\ &= \mathcal{N}G^i(G^{L_1})^j(G^{L_2})^v\\ &= \mathcal{N}G^i(I_n \oplus G)^j(I_n \oplus G \oplus G^2)^v\\ &= (\mathcal{N}(I_n \oplus G \oplus G^2)^v)G^i(I_n \oplus G)^j\\ &= XG^i(I_n \oplus G)^j\end{aligned}$$

where $X = \mathcal{N}(I_n \oplus G \oplus G^2)^v$. Note that v is a tweak to the MAC algorithm itself and is independent of the actual message to be authenticated. At the start, we compute $X = \mathcal{N}(I_n \oplus G \oplus G^2)^v$. The value $\mathcal{N} = E_K(0^n)$ is computed and then the map $(I_n \oplus G \oplus G^2)$ is applied v times to it. This can be done by the following algorithm.

1. $\mathcal{N} = E_K(0^n)$;
2. **for** $i = 1$ to v **do**
3. $\quad A = \mathcal{N}G$; $B = AG$; $\mathcal{N} = \mathcal{N} \oplus A \oplus B$;
4. **end do**;

Executing the above algorithm requires a total of $2v$ applications of G. Recall that each application of G is very cheap when G is realized using either an LFSR, or a powering construction, or as a CA map.

Once X is computed, we can iteratively compute XG^i by applying G to the previously generated value. Suppose the last value that is obtained is Z. To Z we apply $(I_n \oplus G)^j$. The value of j is 1 or 2 and applying $(I_n \oplus G)^j$ is similar to applying $(I_n \oplus G \oplus G^2)^v$ shown above.

Word-Oriented LFSR: As mentioned earlier in relation to the AE mode of operation, it is possible to choose the pair of polynomials $(\tau_1(x), \tau_2(x))$ such that the discrete logs of $x \oplus 1$ and $x^2 \oplus x \oplus 1$ have suitable values. In fact, we expect that there are many such choices of $(\tau_1(x), \tau_2(x))$.

2) $\boldsymbol{R}$ **as** $\mathbb{Z}_{2^n}$: Let $B = 2^{n/2} - 1$, $L_1 = (\mathsf{V}+1)2^{n/2}$, and $L_2 = 2^{n/2}$. Then the conditions on L_1 and L_2 are satisfied. We have

$$\begin{aligned}&f_{i+jL_1+vL_2}(\mathcal{N})\\ &\quad= ((i + jL_1 + vL_2 + 1)\mathcal{N} \bmod p) \bmod 2^n\\ &\quad= ((vL_2\mathcal{N} \bmod p) + (jL_1\mathcal{N} \bmod p)\\ &\qquad\quad + ((i+1)\mathcal{N} \bmod p) \bmod p) \bmod 2^n\\ &\quad= ((X_2 + X_1 + ((i+1)\mathcal{N}) \bmod p) \bmod p) \bmod 2^n\end{aligned}$$

where $X_2 = vL_2\mathcal{N} \bmod p$ and $X_1 = jL_1\mathcal{N} \bmod p$. Since v does not depend on the message, we start by computing $Z = X_2$. Let $Z_i = X_2 + (i+1)\mathcal{N} \bmod p$. Then the value of $f_{i+vL_2}(\mathcal{N})$ equals the n least significant bits of Z_i. Finally, we obtain the value of $f_{i+jL_1+vL_2}(\mathcal{N})$ by adding X_1 to Z_m and taking the n least significant bits.

B. Interleaved Separation

In this case, we define

$$\phi(i,j,v) = 3(\mathsf{V}+1)i + (\mathsf{V}+1)j + v. \tag{11}$$

The injectivity of ϕ is readily verified. Starting from $f_v(\mathcal{N})$ it is easy to compute $f_{3(\mathsf{V}+1)i+v}(\mathcal{N})$ iteratively for both the cases when $\boldsymbol{R}$ is GF(2^n) or $\mathbb{Z}_{2^n}$. Finally, it is also easy to compute the value of $f_{3\mathsf{V}m+\mathsf{V}j+v}(\mathcal{N})$ from $f_{3\mathsf{V}m+v}(\mathcal{N})$ in both cases. This technique does not require the integers L_1 and L_2 and hence in the case of $\boldsymbol{R}$ being realized as GF(2^n) there is no need for any discrete log computation. The disadvantage is that compared to the technique of linear separation, this technique is costlier. Computing the masks is about $3(\mathsf{V}+1)$ times costlier. In the case, where $\mathsf{V} = 1$, as in the application to the construction of AEAD, this cost is within tolerable limits.

C. Security

As in [19], the MAC construction is secure as a tweakable PRF. The advantage of an adversary A with respect to a tweakable PRF $\widetilde{\boldsymbol{F}}$ is defined in the following manner:

$$\begin{aligned}\mathbf{Adv}^{\widetilde{\mathrm{prf}}}_{\widetilde{\boldsymbol{F}}}(A) &= \mathsf{Prob}[K \xleftarrow{\$} \mathcal{K} : A^{\widetilde{\boldsymbol{F}}_K(\cdot,\cdot)} \Rightarrow 1] \\ &\quad - \mathsf{Prob}[\rho \xleftarrow{\$} \mathrm{Rand}(\mathcal{V}\times\mathcal{M},\tau) : A^{\rho(\cdot,\cdot)} \Rightarrow 1]. \end{aligned}\tag{12}$$

The security result of the MAC construction is similar to that of [17, Corollary 17]. We state the corresponding result.

Theorem 3: Fix $n \geq 1$ and $\tau \in [1..n]$. Let $E : \mathcal{K}\times\{0,1\}^n \to \{0,1\}^n$ be used to instantiate the XE construction of $\widetilde{E}$ as in Fig. 2. Then

$$\mathbf{Adv}^{\widetilde{\mathrm{prf}}}_{\mathrm{MAC}[E,\tau]}(t,\sigma_n) \leq \mathbf{Adv}^{\mathrm{prp}}_{E}(t',\sigma_n) + \frac{5q^2}{2^{n+1}} + \frac{2q^2}{\mu}$$

where $\mu = 2^{n-1}/(\delta+1)$ if $\boldsymbol{R}$ is instantiated as $\mathbb{Z}_{2^n}$ and $\mu = 2^n$ if $\boldsymbol{R}$ is instantiated as GF(2^n).

VII. Authenticated Encryption With Associated Data

An AEAD is a protocol which allows the authentication of a header (also called associated data) without encrypting it. The encryption algorithm for an AEAD protocol takes as input (the key and) a header, a nonce, and a message. It produces as output a ciphertext which consists of the encryption of the message and a tag which authenticates both the message and the header. The decryption algorithm takes as input (the key and) a header, a nonce and a ciphertext. It produces as output either the corresponding message or returns invalid. Authenticating the header without encrypting it is of use in some practical situations. One example is Internet packets which consist of a header and a message. Both of these must be authenticated. However, if the header is encrypted, then it will be difficult for Internet routers to forward the packets. An AEAD protocol exactly fits this application. See [18] for more details on applications of AEAD.

Algorithm Tag-Generation(K, v, M)
  Partition M into $M[1]\cdots M[m]$;
  $\mathcal{N} = E_K(0^n)$;
  sum $= 0^n$;
  for $i = 1$ to $m-1$ **do**
    mask $= \Delta_{i,0,v}$;
    $Y = E_K(M[i] +$ mask$)$;
    sum $=$ sum $+ Y$;
  end for;
  if $|M[m]| = n$
  then mask $= \Delta_{m,1,v}$; sum $=$ sum $+ M[m]$;
  else mask $= \Delta_{m,2,v}$; sum $=$ sum $+ (M[m]10^*)$;
  $T = E_K($sum $+$ mask$)$;
  set tag to the first τ bits of T;
  return tag.

Fig. 2. The tag-generation algorithm of a tweakable MAC protocol over $\boldsymbol{R}$. The algorithm takes as input (K, v, M) where K is the key, v is the tweak and M is the message. It produces as output a τ-bit tag.

It has been shown in [19] that the tweakable MAC can be combined with the AE construction to obtain an AEAD construction. The basic idea is to use the technique of ciphertext translation from [18] and tweak the MAC construction using $v = 1$. The header is authenticated by the MAC algorithm and the message is encrypted using the AE algorithm. Finally, the tag for the header is XORed into the required number of last bits of the output of the AE algorithm (which is the ciphertext and the tag for the message). We discuss how this can be done in our setting.

The input to the AEAD algorithm is a triple (N, H, M), where N is an n-bit nonce, H is the header, and M is the message. Let ϕ be an injective map (obtained by either the linear or the interleaved separation) from $\{1,\ldots,B\}\times\{0,1,2\}\times\{0,1\}$ to $\{1,\ldots,2^n-2\}$. For $(i,j,v) \in \{1,\ldots,B\}\times\{0,1,2\}\times\{0,1\}$ and $\mathcal{N} \in \{0,1\}^n$, we define a set of masks $\Delta_{i,j,v}(\mathcal{N}) = f_{\phi(i,j,v)}(\mathcal{N})$. The MAC construction requires a TBC obtained by the XE construction, while the AE construction requires a TBC obtained by the XEX construction. Both these constructions require masks of the type $f_k(\mathcal{N})$. Defining these masks will make the algorithm precise.

The masks for the first $h-1$ header blocks in the MAC algorithm are

$$\Delta_{1,0,1}(\mathcal{N}'), \Delta_{2,0,1}(\mathcal{N}'), \ldots, \Delta_{h-1,0,1}(\mathcal{N}')$$

where $\mathcal{N}' = E_K(0^n)$. The mask for the last header block is $\Delta_{h,1,1}(\mathcal{N}')$ or $\Delta_{h,2,1}(\mathcal{N}')$ according as whether H_h is full or partial.

In the AE algorithm, the masks are used as follows. The masks for the m message blocks are

$$\Delta_{1,0,0}(\mathcal{N}), \Delta_{2,0,0}(\mathcal{N}), \ldots, \Delta_{m,0,0}(\mathcal{N})$$

where $\mathcal{N} = E_K(N)$. The mask for encrypting the checksum sum in the AE algorithm is $\Delta_{m,1,0}(\mathcal{N})$. With the above mask

```
Algorithm Tag-Generation(K, v, M)
  Partition M into M[1] ··· M[m];
  N = E_K(0^n);
  sum = 0^n;
  for i = 1 to m − 1 do
    mask = Δ_{i+2};
    Y = E_K(M[i] + mask);
    sum = sum + Y;
  end for;
  if |M[m]| = n
  then mask = Δ_1; sum = sum + M[m];
  else mask = Δ_2; sum = sum + (M[m]10*);
  T = E_K(sum + mask);
  set tag to the first τ bits of T;
  return tag.
```

Fig. 3. The tag-generation algorithm of a MAC protocol over $\boldsymbol{R}$. The algorithm takes as input (K, M) where K is the key and M is the message. It produces as output a τ-bit tag.

definitions and the protocols in Figs. 1 and 2, it is easy to fill out the details of the AEAD protocol.

VIII. Different MAC and AEAD Constructions

The MAC construction described in Section VI is essentially the construction in [19] instantiated by the more general tweakable block cipher construction with the option of applying either the linear or the interleaved separation techniques. In this section, we describe a MAC construction which is different from that in [19] and an AEAD protocol based on it. The MAC construction that we describe is closer to the construction in [4]. The algorithm is described in Fig. 2. It requires the masks $\Delta_3, \Delta_4, \ldots, \Delta_{m+1}$ and either Δ_1 or Δ_2. Defining these masks from the f-functions is easy. For $i \geq 1$, define

$$\Delta_i = f_i(\mathcal{N}), \quad \text{where } \mathcal{N} = E_K(0^n).$$

Thus, starting from $f_3(\mathcal{N})$, we compute the masks in an iterative manner. The (minor) disadvantage is that we have to carry forward the values of both $f_1(\mathcal{N})$ and $f_2(\mathcal{N})$. This is because it is only at the end of the message we get to know which one will be required.

AEAD Protocol: Based on this MAC protocol, we can define an AEAD protocol in the following manner. Actually, we slightly modify the MAC protocol by defining

$$\left.\begin{array}{l} \Delta_1 = f_1(\mathcal{N}) \\ \Delta_2 = f_2(\mathcal{N}) \\ \Delta_i = f_{3(i-2)}(\mathcal{N}) \text{ and for } i \geq 3 \end{array}\right\}. \tag{13}$$

The outline of the AEAD algorithm is as follows. Let there be h header blocks $H_1, \ldots, H_h$ and m message blocks $M_1, \ldots, M_m$. The last header block H_h can be partial and the last message block M_m can be partial.

1) Generate a MAC for the header using Fig. 3 but using the definition of Δ given by (13) and with $\mathcal{N} = E_K(0^n)$. Let T be the produced tag. If the header is empty, set T to be the empty string.
2) Encrypt the message blocks using the AE algorithm of Fig. 1 but using the mask $f_{3(h+i)+1}(\mathcal{N})$ (with $\mathcal{N} = E_K(N)$, where N is the nonce) for the ith message block and the mask $f_{3(h+m)+2}(\mathcal{N})$ for the checksum sum. This gives us the pair (C, tag), where C is the ciphertext and tag is the tag.
3) XOR T into the last $|T|$ bits of (C, tag) and return the result.

IX. Conclusion

The concept of TBCs and the theme of designing modes of operations based upon TBCs was introduced in [13]. The first efficient construction of TBCs was presented in [19] and the same paper presented AE, MAC, and AEAD protocols. We build on the work in [19]. Our first contribution is to present a general construction of an efficient TBC. We work over a ring $\boldsymbol{R}$ which can be instantiated as either GF(2^n) or as $\mathbb{Z}_{2^n}$. The construction of TBC in [19] can be seen as a special case (instantiating $\boldsymbol{R}$ as GF(2^n) and using the powering construction) of our construction. The general TBC construction is used to instantiate general constructions of AE, MAC, and AEAD protocols from [19] in several ways. This leads to a suite of efficient protocols for these applications out of which only one of each kind has been described earlier in [19].

Acknowledgment

The authors would like to thank the reviewers for their comments which helped in improving the paper.

References

[1] [Online]. Available: http://csrc.nist.gov/CryptoToolkit/modes/
[2] M. Bellare and C. Namprempre, "Authenticated encryption: Relations among notions and analysis of the generic composition paradigm," in *ASIACRYPT (Lecture Notes in Computer Science)*, T. Okamoto, Ed. Berlin, Germany: Springer-Verlag, 2000, vol. 1976, pp. 531–545.
[3] M. Bellare, P. Rogaway, and D. Wagner, "The EAX mode of operation," in *Fast Software Encryption (Lecture Notes in Computer Science)*, B. K. Roy and W. Meier, Eds. Berlin, Germany: Springer-Verlag, 2004, vol. 3017, pp. 389–407.
[4] J. Black and P. Rogaway, "A block-cipher mode of operation for parallelizable message authentication," in *Proc. EUROCRYPT (Lecture Notes in Computer Science)*, L. R. Knudsen, Ed. Berlin, Germany: Springer-Verlag, 2002, vol. 2332, pp. 384–397.
[5] D. Chakraborty and P. Sarkar, "A general construction of tweakable block ciphers and different modes of operations," in *Proc. Inscrypt (Lecture Notes in Computer Science)*, H. Lipmaa, M. Yung, and D. Lin, Eds. Berlin, Germany: Springer-Verlag, 2006, vol. 4318, pp. 88–102.
[6] P. Ekdahl and T. Johansson, "A new version of the stream cipher SNOW," in *Selected Areas in Cryptography (Lecture Notes in Computer Science)*, K. Nyberg and H. M. Heys, Eds. Berlin, Germany: Springer, 2002, vol. 2595, pp. 47–61.
[7] V. D. Gligor and P. Donescu, "Fast encryption and authentication: XCBC encryption and XECB authentication modes," in *Proc. FSE (Lecture Notes in Computer Science)*, M. Matsui, Ed. Berlin, Germany: Springer-Verlag, 2001, vol. 2355, pp. 92–108.
[8] S. Halevi and H. Krawczyk, "MMH: Software message authentication in the gbit/second rates," in *Fast Software Encryption (Lecture Notes in Computer Science)*, E. Biham, Ed. Berlin, Germany: Springer-Verlag, 1997, vol. 1267, pp. 172–189.
[9] T. Iwata and K. Kurosawa, "Omac: One-key cbc mac," in *Proc. FSE (Lecture Notes in Computer Science)*, T. Johansson, Ed. Berlin, Germany: Springer-Verlag, 2003, vol. 2887, pp. 129–153.
[10] C. S. Jutla, "Encryption modes with almost free message integrity," in *Proc. EUROCRYPT (Lecture Notes in Computer Science)*, B. Pfitzmann, Ed. Berlin, Germany: Springer-Verlag, 2001, vol. 2045, pp. 529–544.
[11] J. Katz and M. Yung, "Complete characterization of security notions for probabilistic private-key encryption," in *Proc. Symp. Theory of Computing*, Portland, OR, May 2000, pp. 245–254.
[12] R. Lidl and H. Niederreiter, *Introduction to Finite Fields and Their Applications, Revised Edition*. Cambridge, U.K.: Cambridge Univ. Press, 1994.

[13] M. Liskov, R. L. Rivest, and D. Wagner, "Tweakable block ciphers," in *Proc. CRYPTO (Lecture Notes in Computer Science)*, M. Yung, Ed. Berlin, Germany: Springer-Verlag, 2002, vol. 2442, pp. 31–46.

[14] S. Lucks, "Two-pass authenticated encryption faster than generic composition," in *Fast Software Encryption (Lecture Notes in Computer Science)*, H. Gilbert and H. Hschuh, Eds. Berlin, Germany: Springer-Verlag, 2005, vol. 3557, pp. 284–298.

[15] D. A. McGrew and J. Viega, "The security and performance of the galois/counter mode (GCM) of operation," in *Proc. INDOCRYPT (Lecture Notes in Computer Science)*, A. Canteaut and K. Viswanathan, Eds. Berlin, Germany: Springer-Verlag, 2004, vol. 3348, pp. 343–355.

[16] K. Minematsu, "Improved security analysis of XEX and LRW modes," in *Selected Areas in Cryptography (Lecture Notes in Computer Science)*, E. Biham and A. M. Youssef, Eds. Berlin, Germany: Springer-Verlag, 2006, vol. 4356, pp. 96–113.

[17] P. Rogaway, "Efficient Instantiations of Tweakable Blockciphers and Refinements to Modes OCB and PMAC," [Online]. Available: http://www.cs.ucdavis.edu/~rogaway/papers/index.html.

[18] P. Rogaway, V. Atluri, Ed., "Authenticated-encryption with associated-data," in *Proc. ACM Conf. Computer and Communications Security*, 2002, pp. 98–107.

[19] P. Rogaway, "Efficient instantiations of tweakable blockciphers and refinements to modes OCB and PMAC," in *Proc. ASIACRYPT (Lecture Notes in Computer Science)*, P. J. Lee, Ed. Berlin, Germany: Springer-Verlag, 2004, vol. 3329, pp. 16–31.

[20] P. Rogaway, M. Bellare, and J. Black, "OCB: A block-cipher mode of operation for efficient authenticated encryption," *ACM Trans. Inf. Syst. Secur.*, vol. 6, no. 3, pp. 365–403, 2003.

[21] S. Tezuka and M. Fushimi, "A method of designing cellular automata as pseudo random number generators for built-in self-test for VLSI ," in *Finite Fields: Theory, Applications and Algorithms, Contemporary Mathematics*. Providence, RI: AMS, 1994, pp. 363–367.

[22] S. Vaudenay, "Decorrelation: A theory for block cipher security," *J. Cryptol.*, vol. 16, no. 4, pp. 249–286, 2003.