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

Debrup Chakraborty[1] and Palash Sarkar[2]

[1] Computer Science Department
CINVESTAV-IPN
Mexico, D.F., 07360, Mexico
email: debrup@cs.cinvestav.mx

[2] Applied Statistics Unit
Indian Statistical Institute
203, B.T. Road, Kolkata
India 700108.
email: palash@isical.ac.in

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 $\mathbf{R}$ and by the use of a masking sequence of functions. The ring $\mathbf{R}$ 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 Linear Feedback Shift Register (LFSR), a powering construction or a cellular automata map. 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 mode of operation. Such modes of operations have great practical utility. [3]

Keywords: tweakable block cipher, modes of operations, AE, MAC, AEAD.

1 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.

Liskov, Rivest and Wagner [10] introduced the concept of tweakable block cipher, which is a block cipher with an additional input called a tweak. The tweak is meant to provide variability and not security. They also showed that it is possible to build secure modes of operations starting from a TBC. This theme was developed by Rogaway in [15] where efficient constructions of TBC and different modes of operations were presented.

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 requires one invocation per block of the message. This yields an efficiency improvement by a factor of two over conventional approaches. The known one-pass proposals are IACBC, IAPM by Jutla [7]; XCBC, XECB by Gligor-Donescu [6]; and OCB, OCB1 by Rogaway [15]. Unfortunately, all these proposals have pending patent claims. This has prevented their wide spread use and adoption in standards. In fact, NIST [1] has standardised a two-pass algorithm for achieving AE. Another undesirable

[3] An abridged version of this paper appears as [5]

effect of the patent claims is that this has led to some researchers proposing new two-pass AE protocols [3, 11].

An important practical aspect of our work is to uncover a new family of efficient one-pass AE modes of operations. This provides a designer with a greater choice of algorithms. A discussion on patent issues regarding these new modes is given in Section A.

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 [15]. Below we mention our specific contributions and relate to the work of [15].

Tweakable block cipher: We define a sequence $f_1, f_2, \ldots, f_{2^n-2}$, with $f_i : \{0,1\}^n \rightarrow \{0,1\}^n$, of functions with a particular set of properties to be a masking sequence. Given block cipher $E : \mathcal{K} \times \{0,1\}^n \rightarrow \{0,1\}^n$ 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 $\mathbf{R} = (\{0,1\}^n, +, \cdot)$ with identity. Typical instantiations of $\mathbf{R}$ are as $GF(2^n)$ and $\mathbb{Z}_{2^n}$.

In the case where $\mathbf{R}$ 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 characteristics polynomial. Efficient realization of G can be done by a linear feedback shift register (LFSR), a powering construction used in [15] or as a cellular automata (CA) map. In the case where $\mathbf{R}$ 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 [15] over $GF(2^n)$ using the powering construction. The abstraction of the ring $\mathbf{R}$, the use of LFSR and CA and the instantiation of $\mathbf{R}$ as $\mathbb{Z}_{2^n}$ are new to this paper.

Authenticated Encryption (AE): Given a TBC with an appropriate tweak space, Rogaway [15] showed how to construct an AE protocol. Rogaway instantiates his AE construction with his TBC construction. This method 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 [15], 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.

Tweakable block cipher: Rogaway describes the XE and the XEX constructions over $GF(2^n)$ using the powering construction. We generalize this by working over a ring $\mathbf{R}$ which can be instantiated as either $GF(2^n)$ or as $\mathbb{Z}_{2^n}$. Further, over $GF(2^n)$, we show that there are other efficient alternatives to the powering construction.

Modes of Operations: 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.

PREVIOUS AND RELATED WORK: The formal model of security for AE was independently proposed by [8] and [2]. Jutla [7] proposed constructions for single-pass AE, including one fully parallelizable protocol. Independent work due to Gligor and Donescu [6] also proposed single-pass AE protocols. A refinement and extension of Jutla's parallelizable protocol was done by Rogaway [16] and was called the OCB.

In a separate development, the notion of TBCs and their application to modes of operations was proposed by Liskov, Rivest and Wagner [10]. The construction of TBC in [10] was not very efficient. The first efficient construction of TBC was given by Rogaway [15]. As discussed earlier, our work is a development on the work of [15].

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, 14]. A separate line of research has consisted of developing two-pass AE protocols (some examples are [12, 3, 11]). The work [11] presents an AE protocol which is somewhere between one and two pass protocols.

In a recent work, Minematsu [13] revisits the work on TBC appearing in [10] and [15]. The work [13] provides some improvements to the construction given in [10]. The XEX construction in [15] 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 generalised this simpler form. In contrast, Minematsu [13] presents a new analysis of the XEX description as given in [15]. We would like to emphasize that none of the techniques for XEX construction introduced in this paper is present in [13]. Also, none of the techniques for constructing modes of operations is present in [13]. Thus, this work and that of [13], though on the similar topics, are really of independent interest.

2 Preliminaries

Our notation and definitions closely follow [15].

A block cipher is a map $E : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$, where $\mathcal{K}$ is a finite non-empty 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 non-empty 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 $\widetilde{D}(K, T, \widetilde{E}(K, T, X)) = X$.

Perm(n) denotes the set of all permutations of $\{0,1\}^n$ and Perm$(\mathcal{T}, n)$ denotes the set of all mappings from $\mathcal{T}$ to Perm(n). Similarly Rand(n) denotes the set of all n bit to n bit functions and Rand$(\mathcal{T}, n)$ denotes the set of all mappings from $\mathcal{T}$ to 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 the 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.*

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

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.

3 Construction of Tweakable Block Ciphers

Let $\mathbf{R} = (\{0,1\}^n, +, \cdot)$ 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 $f_s : \{0,1\}^n \rightarrow \{0,1\}^n$. We say that the sequence is an (n,m,μ) masking sequence if the following properties hold.*

$$\begin{array}{llll}
(1) & \mathsf{Prob}[f_s(\mathcal{N}) = \alpha] & \le \frac{1}{\mu}, & \textit{for } 1 \le s \le m.\\
(2) & \mathsf{Prob}[f_s(\mathcal{N}) = \mathcal{N} + \alpha] & \le \frac{1}{\mu}, & \textit{for } 1 \le s \le m.\\
(3) & \mathsf{Prob}[f_s(\mathcal{N}) = f_t(\mathcal{N}) + \alpha] & \le \frac{1}{\mu}, & \textit{for } 1 \le s,t \le m \textit{ and } s \ne t.\\
(4) & \mathsf{Prob}[f_s(\mathcal{N}) = f_t(\mathcal{N}') + \alpha] & \le \frac{1}{\mu}, & \textit{for } 1 \le s,t \le m.
\end{array}$$

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

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 $f_{s+1}(\mathcal{N})$.

The construction of a TBC that we present below is a natural generalization of the construction given in [15]. We construct a TBC

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

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.

$$\widetilde{E}_K^{N,l}(M) = E_K(M + \Delta), \text{ where } \Delta = f_l(\mathcal{N}) \text{ and } \mathcal{N} = E_K(N). \qquad (1)$$

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

$$\widetilde{E}_K^{N,l}(M) = E_K(M + \Delta) - \Delta, \text{ where } \Delta = f_l(\mathcal{N}) \text{ and } \mathcal{N} = E_K(N). \tag{2}$$

The operations "+" and "−" in the XE and the XEX constructions are over the ring $\mathbf{R}$. 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 [15]. We generalize by working over $\mathbf{R}$ and the use of the masking sequence of functions. Later we show that there are several different ways of efficiently instantiating $\mathbf{R}$ 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 [15]. The proof of the XEX construction was not given in [15] 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 [15].

Theorem 1 (Security of XE and XEX Constructions).
Security of XE:

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

Security of XEX:

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

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

Proof :

Proof of the XE Construction: As in [15], a hybrid argument is required. The following five hybrids were identified in [15].

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 [15] also hold in our case. These bounds are as follows.

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

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 Domain(ρ) denote the set of points at which ρ has currently been defined. Thus, initially Domain(ρ) 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.       $\mathcal{N}^j \overset{\$}{\leftarrow} \{0,1\}^n$;
4.       if $N^j \in \text{Domain}(\rho)$ then bad = true; $\boxed{\mathcal{N}^j = \rho(N^j);}$
5. $\rho(N^j) = \mathcal{N}^j$;
6. $Y^j \overset{\$}{\leftarrow} \{0,1\}^n$; $X^j = M^j + f_{l^j}(\mathcal{N}^j)$;
7. if $X^j \in \text{Domain}(\rho)$ 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 Figure 1 of [15] with one exception. In Step 6, we use the function $f_{l^j}(\cdot)$ and the addition $+$ is over the ring $\mathbf{R}$.

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 non-adaptive. 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,

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

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

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

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 Domain(ρ). For the XEX construction, the simulator needs to maintain both Domain(ρ) and Range(ρ) and the adversary's advantage is bounded above by the probability of a collision in either Domain(ρ) or Range(ρ). The collision analysis for Range(ρ) is a little different from that of Domain(ρ) 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. $p_1 = \mathsf{Prob}[K \xleftarrow{\$} \mathcal{K} : A^{\widetilde{E}_K(.,.),\widetilde{E}_K^{-1}(.,.)} \Rightarrow 1]$.
2. $p_2 = \mathsf{Prob}[\pi \xleftarrow{\$} \mathrm{Perm}(n) : A^{\widetilde{\pi}(.,.),\widetilde{\pi}^{-1}(.,.)} \Rightarrow 1]$.
3. $p_3 = \mathsf{Prob}[\rho_1, \rho_2 \xleftarrow{\$} \mathrm{Rand}(n) : A^{\widetilde{\rho_1}(.,.),\widetilde{\rho_2}(.,.)} \Rightarrow 1]$.
4. $p_4 = \mathsf{Prob}[\rho_1, \rho_2 \xleftarrow{\$} \mathrm{Rand}(\mathcal{T}, n) : A^{\rho_1(.,.),\rho_2(.,.)} \Rightarrow 1]$.
5. $p_5 = \mathsf{Prob}[\pi \xleftarrow{\$} \mathrm{Perm}(\mathcal{T}, n) : A^{\pi(.,.),\pi^{-1}(.,.)} \Rightarrow 1]$.

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 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) according as 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.      $\mathcal{N}^j \xleftarrow{\$} \{0,1\}^n$;
4.      **if** $N^j \in$ Domain **then** bad = true; $\boxed{\mathcal{N}^j = \rho_1(N^j);}$
5.  $\rho_1(N^j) = \mathcal{N}^j$;
6.  **if** the $j$th query is an encryption query **then**
7.      $X^j = M^j + f_{l^j}(\mathcal{N}^j)$;
7.      $C^j \xleftarrow{\$} \{0,1\}^n$; $Y^j = C^j + f_{l^j}(\mathcal{N}^j)$;
8.      **if** $X^j \in$ Domain **then** bad = true; $\boxed{Y^j = \rho_1(X^j);}$
9.      $\rho_1(X^j) = Y^j$; $C^j = Y^j - f_{l^j}(\mathcal{N}^j)$;
10.     **return** $C^j$;
11. **if** the $j$th query is a decryption query **then**
12.     $Y^j = C^j + f_{l^j}(\mathcal{N}^j)$;
12.     $M^j \xleftarrow{\$} \{0,1\}^n$; $X^j = M^j + f_{l^j}(\mathcal{N}^j)$;
13.     **if** $Y^j \in$ Range **then** bad = true; $\boxed{X^j = \rho_2(Y^j);}$
14.     $\rho_2(Y^j) = X^j$; $M^j = X^j - f_{l^j}(\mathcal{N}^j)$;
15.     **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 analyse 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 non-adaptive. 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 $N^j, M^j + f_{i^j}(\mathcal{N}^j)$ whereas the elements of the set Range are of the form $\mathcal{N}^j, C^j + f_{i^j}(\mathcal{N}^j)$. 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.

$(\mathcal{N}^i, \mathcal{N}^j)$: 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.

(Y^i, Y^j): Now

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

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 $\mathsf{Prob}[Y^i = Y^j] \leq 1/\mu$.

$(Y^i, \mathcal{N}^j)$*:* 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 [13].

4 Instantiating R

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

4.1 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 $1 \leq l \leq 2^n - 2$, 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.

(2) 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 Defintion 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 non-zero 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 $\tau(x)|(x^{s-t} - 1)$. Since $0 \leq t < s \leq 2^m - 2$, the last condition contradicts the primitivity 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 $\tau(x) = x^n \oplus t_{n-1}x^{n-1} \oplus t_1 x \oplus t_0$. 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.

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

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 [9]).

Powering Construction: Let $a(x)$ be a polynomial of degree less than n. The map used in [15] 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 tri-diagonalization procedure due to Tezuka and Fushimi [17].

Hardware implementation of one LFSR or CA operation can be completed in one clock cycle. Hardware implementation of the powering construction requires one shift and one conditional XOR if $a_{n-1} = 1$ leading to one or two clock cycles per operation. Software implementation of all the above three methods requires a few shifts and XORs.

4.2 R as $\mathbb{Z}_{2^n}$

The set $\{0,1\}^n$ can be considered to be the set of all non-negative 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 following table of primes. These cover the most typical values of n used in practical applications.

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

The set $\mathbb{Z}_p$ is a field under addition and multiplication modulo p and this field contains the integers $0, \ldots, 2^n - 1$. For $i \geq 1$, we define

$$f_i(\mathcal{N}) = ((i+1) \times \mathcal{N} \bmod p) \bmod 2^n. \tag{6}$$

Proposition 2. *The sequence $f_1, f_2, \ldots, f_{2^n-2}$ defined by (6) 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 $B_1 \subseteq \{0, 1, \ldots, 2^n - 1\}$ and $B_2 \subseteq \{2^n, \ldots, 2^n + \delta - 1\}$. Now when we perform the modulo 2^n operation, two elements of B_1 cannot collide and 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 0, 1 or 2 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

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

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

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

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

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

where $0 \le r_{i,1}, r_{j,1}, r_{i,2}, r_{j,2} < 2^n$. Also, $\psi_{i,j}(\mathcal{N}_1) = r_{i,1} - r_{j,1} \ge 0$ and $\psi_{i,j}(\mathcal{N}_2) = r_{i,2} - r_{j,2} \ge 0$. Let if possible, $\psi_{i,j}(\mathcal{N}_1) = \psi_{i,j}(\mathcal{N}_2)$ for $\mathcal{N}_1 \ne \mathcal{N}_2$. 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 $0 \le i, j, \mathcal{N}_1, \mathcal{N}_2 < 2^n$ 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

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

and hence

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

Thus,

$$|\overline{A}| \le |\{\mathcal{N} \in \mathbb{Z}_{2^n} : 2^n \le (i+1)\mathcal{N} \bmod p < p\}| + |\{\mathcal{N} \in \mathbb{Z}_{2^n} : 2^n \le (j+1)\mathcal{N} \bmod p < p\}|$$

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,

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

where $\delta = p - 2^n$. Thus, $|\overline{A}| \le 2\delta$. 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

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

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.

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 multi-precision integer addition and at most one subtraction. Thus, software implementation of $f_i(\mathcal{N})$ will be efficient.

5 Authenticated Encryption

Rogaway [15] 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 $\tau \in [0..n]$, Rogaway provides a construction of an AE protocol.
2. The TBC $\widetilde{F}$ is instantiated in [15] 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 $\mathbf{R}$. 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 [15], 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 [15] 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 $\Delta_{i,b}(\mathcal{N}) = f_{\phi(i,b)}(\mathcal{N})$. Figure 1 shows the AE protocol of [15] written using the Δ's. The statement on the security of the protocol is given in Section 5.4.

Fig. 1. Encryption and decryption algorithms of an AE protocol over $\mathbf{R}$. 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 $\Delta_{i,b}(\mathcal{N}) = f(\mathcal{N})$.

```
Algorithm Encrypt($K, N, M$)
  Partition $M$ into $M[1] \cdots M[m]$;
  $\mathcal{N} = E_K(N)$;
  $\mathsf{sum} = 0^n$;
  for $i = 1$ to $m - 1$ do
    $\mathsf{mask} = \Delta_{i,0}(\mathcal{N})$;
    $C[i] = E_K(M[i] + \mathsf{mask}) - \mathsf{mask}$;
    $\mathsf{sum} = \mathsf{sum} + M[i]$;
  end for;
  $\mathsf{mask} = \Delta_{m,0}(\mathcal{N})$;
  $\mathsf{Pad} = E_K(\mathsf{len}(M[m]) + \mathsf{mask}) - \mathsf{mask}$;
  $C[m] = M[m] + \mathsf{Pad}$;
  $C = C[1] \cdots C[m]$;
  $\mathsf{sum} = \mathsf{sum} + (C[m]0^*) + \mathsf{Pad}$;
  $\mathsf{mask} = \Delta_{m,1}(\mathcal{N})$;
  $T = E_K(\mathsf{sum} + \mathsf{mask}) - \mathsf{mask}$;
  set tag to the first $\tau$ bits of $T$;
  return $(C, \mathsf{tag})$.
```

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

In Figure 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.

5.1 Linear Separation

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

$$\phi(i, b) = i + Lb. \tag{7}$$

The injectivity of ϕ is easily verified. In Figure 1, the use of (7) 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.

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 $\mathbf{R}$ is realized as $GF(2^n)$ or as $\mathbb{Z}_{2^n}$.

Task 2 We show the efficiency of this task separately for the realization of $\mathbf{R}$ as $GF(2^n)$ and $\mathbb{Z}_{2^n}$.

R *as* $GF(2^n)$*:* In this case, the technique of [15] 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 [15] 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 $G^L \oplus G \oplus I_n = q(G)\tau(G) = 0$. Thus, for any $\mathcal{N} \in \{0,1\}^n$, we have $\mathcal{N}G^L = \mathcal{N}(G \oplus I_n)$. Hence, we have

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

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.

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.

5.2 Interleaved Separation

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

$$\phi(i, b) = 2i + b. \tag{8}$$

The injectivity of ϕ is easily verified. In Figure 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 **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.

5.3 Comparison Among the AE Protocols

At a top level we have four single-pass AE protocols. There are two options for instantiating the ring **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 **R** as $GF(2^n)$ there are different possibilities for implementing G. We have indicated three – as an LFSR, using the powering construction or as a CA.

The AE protocol in [15] corresponds to the instantiation of **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 [6], Jutla [7] and the earlier protocol of Rogaway [16] are not covered by our general description.

From the viewpoint of efficiency consideration, the protocols that we have developed as well as the other protocols mentioned above have roughly the same efficiency, especially for long messages. Each of these AE protocols are quite efficient to implement and will satisfy the speed requirements of most applications. One may choose a particular protocol depending upon other factors.

5.4 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 [15], 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:

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

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,

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

The result on the security of the AE protocol of Figure 1 is stated below and is a minor modification of Corollary 6 of [15].

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

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

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

6 MAC Construction

In [15], the TBC obtained from the XE construction is used to construct a MAC protocol. In fact, a more general construction of PRF is presented in [15]. Under the assumption (implicit in [15]) 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 [15] written in terms of the $\Delta_{i,j,v}$'s is shown in Figure 2. The security statement is given in Section 6.3. 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.

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

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.

Fig. 2. The tag generation algorithm of a tweakable MAC protocol over **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.

```
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 $= \Delta_{i,0,v}$;
    $Y = E_K(M[i] + \mathsf{mask})$;
    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 $+ (M[m]10^*)$;
  $T = E_K(\mathsf{sum} + \mathsf{mask})$;
  set tag to the first $\tau$ bits of $T$;
  return tag.
```

6.1 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

$$\phi(i, j, v) = i + L_1 j + L_2 v. \tag{9}$$

Lemma 1. *The map ϕ defined in (9) 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. $\square$

We now consider the two possibilities for **R**.

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 [15] 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.     $A = \mathcal{N}G$; $B = AG$; $\mathcal{N} = \mathcal{N} \oplus A \oplus B$;
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.

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

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

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.

6.2 Interleaved Separation

In this case, we define

$$\phi(i, j, v) = 3(\mathsf{V} + 1)i + (\mathsf{V} + 1)j + v. \tag{10}$$

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 **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 the cases. This technique does not require the integers L_1 and L_2 and hence in the case of **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 more costlier. In the case, where $\mathsf{V} = 1$, as in the application to the construction of AEAD, this cost is within tolerable limits.

6.3 Security

The security result of the MAC construction is similar to that of Corollary 9 of [15]. 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 Figure 2. Then*

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

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

7 AEAD

It has been shown in [15] 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 [14] 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 definitions and the protocols in Figures 1 and 2, it is easy to fill out the details of the AEAD protocol.

8 Different MAC and AEAD Constructions

The MAC construction described in Section 6 is essentially the construction in [15] 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 [15] 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 Figure 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

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

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

$$\Delta_1 = f_1(\mathcal{N}); \Delta_2 = f_2(\mathcal{N}); \text{ and for } i \geq 3, \Delta_i = f_{3(i-2)}(\mathcal{N}). \qquad (*)$$

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.

Fig. 3. The tag generation algorithm of a MAC protocol over **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.

```
Algorithm Tag-Generation$(K, v, M)$
  Partition $M$ into $M[1] \cdots M[m]$;
  $\mathcal{N} = E_K(0^n)$;
  $\mathsf{sum} = 0^n$;
  for $i = 1$ to $m - 1$ do
    $\mathsf{mask} = \Delta_{i+2}$;
    $Y = E_K(M[i] + \mathsf{mask})$;
    $\mathsf{sum} = \mathsf{sum} + Y$;
  end for;
  if $|M[m]| = n$
  then $\mathsf{mask} = \Delta_1$; $\mathsf{sum} = \mathsf{sum} + M[m]$;
  else $\mathsf{mask} = \Delta_2$; $\mathsf{sum} = \mathsf{sum} + (M[m]10^*)$;
  $T = E_K(\mathsf{sum} + \mathsf{mask})$;
  set tag to the first $\tau$ bits of $T$;
  return tag.
```

1. MAC the header using Figure 3 but using the definition of Δ given by $(*)$. 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 Figure 1 but using the mask $f_{3(h+i)+1}(\mathcal{N})$ 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.

9 Conclusion

The concept of TBCs and the theme of designing modes of operations based upon TBCs was introduced in [10]. The first efficient construction of TBCs was presented in [15] and the same paper presented AE, MAC and AEAD protocols. We build on the work in [15]. Our first contribution is to present a general construction of an efficient TBC We work over a ring **R** which can be instantiated as either $GF(2^n)$ or as $\mathbb{Z}_{2^n}$. The construction of TBC in [15] can be seen as a special case (instantiating **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 [15] 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 [15].

References

1. http://csrc.nist.gov/CryptoToolkit/modes/.
2. Mihir Bellare and Chanathip Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In Tatsuaki Okamoto, editor, *ASIACRYPT*, volume 1976 of *Lecture Notes in Computer Science*, pages 531–545. Springer, 2000.
3. Mihir Bellare, Phillip Rogaway, and David Wagner. The EAX mode of operation. In Bimal K. Roy and Willi Meier, editors, *FSE*, volume 3017 of *Lecture Notes in Computer Science*, pages 389–407. Springer, 2004.
4. John Black and Phillip Rogaway. A block-cipher mode of operation for parallelizable message authentication. In Lars R. Knudsen, editor, *EUROCRYPT*, volume 2332 of *Lecture Notes in Computer Science*, pages 384–397. Springer, 2002.
5. Debrup Chakraborty and Palash Sarkar. A general construction of tweakable block ciphers and different modes of operations. In Helger Lipmaa, Moti Yung, and Dongdai Lin, editors, *Inscrypt*, volume 4318 of *Lecture Notes in Computer Science*, pages 88–102. Springer, 2006.

6. Virgil D. Gligor and Pompiliu Donescu. Fast encryption and authentication: XCBC encryption and XECB authentication modes. In Mitsuru Matsui, editor, *FSE*, volume 2355 of *Lecture Notes in Computer Science*, pages 92–108. Springer, 2001.
7. Charanjit S. Jutla. Encryption modes with almost free message integrity. In Birgit Pfitzmann, editor, *EUROCRYPT*, volume 2045 of *Lecture Notes in Computer Science*, pages 529–544. Springer, 2001.
8. Jonathan Katz and Moti Yung. Complete characterization of security notions for probabilistic private-key encryption. In *STOC*, pages 245–254, 2000.
9. R. Lidl and H. Niederreiter. *Introduction to finite fields and their applications, revised edition.* Cambridge University Press, 1994.
10. Moses Liskov, Ronald L. Rivest, and David Wagner. Tweakable block ciphers. In Moti Yung, editor, *CRYPTO*, volume 2442 of *Lecture Notes in Computer Science*, pages 31–46. Springer, 2002.
11. Stefan Lucks. Two-pass authenticated encryption faster than generic composition. In Henri Gilbert and Helena Handschuh, editors, *FSE*, volume 3557 of *Lecture Notes in Computer Science*, pages 284–298. Springer, 2005.
12. David A. McGrew and John Viega. The security and performance of the galois/counter mode (GCM) of operation. In Anne Canteaut and Kapalee Viswanathan, editors, *INDOCRYPT*, volume 3348 of *Lecture Notes in Computer Science*, pages 343–355. Springer, 2004.
13. Kazuhiko Minematsu. Improved security analysis of XEX and LRW modes. In *SAC*, Lecture Notes in Computer Science. Springer, 2006. to appear.
14. Phillip Rogaway. Authenticated-encryption with associated-data. In Vijayalakshmi Atluri, editor, *ACM Conference on Computer and Communications Security*, pages 98–107. ACM, 2002.
15. Phillip Rogaway. Efficient instantiations of tweakable blockciphers and refinements to modes OCB and PMAC. In Pil Joong Lee, editor, *ASIACRYPT*, volume 3329 of *Lecture Notes in Computer Science*, pages 16–31. Springer, 2004.
16. Phillip Rogaway, Mihir Bellare, and John Black. OCB: A block-cipher mode of operation for efficient authenticated encryption. *ACM Trans. Inf. Syst. Secur.*, 6(3):365–403, 2003.
17. 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, AMS*, pages 363–367, 1994.

A Patent Issues for AE Protocols

There are pending patent claims on all known one-pass AE protocols. To the best of our knowledge at the time of writing, none of the claims have actually matured. Nevertheless, one has to pay a licence fee to the designer for the use of the OCB algorithm.

Our work provides several new one-pass AE protocols. We have not applied for any patent on these algorithms and neither do we intend to do so in the future. This, at the least, ensures that we do not add to the already confusing patent issues regarding one-pass AE modes of operations. The natural question is whether our new algorithms can be used to bypass patent issues. This is a legal question and is beyond our capabilities to answer. We, however, note a few points below.

First, our work, like most other scientific work is based on earlier work appearing in the literature. In particular, the work by Rogaway [15] forms the theoretical basis of this paper. Depending on the actual nature of the patent claims, parts of our algorithms could be covered by it.

Second, we would like to emphasize, that our work has significant scientific novelty over and above the earlier work by Rogaway [15]. Several new ideas are introduced which lead to algorithms which are not present in Rogaway's work. We mention two examples.

1. The use of $\mathbb{Z}_p$ for defining the masks is new.
2. In the case of $GF(2^n)$, the use of CA and LFSRs are new.
3. The technique of interleaved separation is totally new. In particular, this avoids the use of discrete log computation during the design phase. The author of [15] considers this feature to be "amusing", but we view it at best to be an irritant.

Third, we welcome people to use our algorithms freely. We do not charge any kind of fee for such usage and we do not expect other people to levy charges for using our algorithms!