Research Article

Avik Chakraborti, Nilanjan Datta* and Mridul Nandi

On the optimality of non-linear computations for symmetric key primitives

https://doi.org/10.1515/jmc-2017-0011
Received March 13, 2017; revised March 25, 2018; accepted May 20, 2018

Abstract: A *block* is an n-bit string, and a (possibly keyed) *block-function* is a non-linear mapping that maps one block to another, e.g., a block-cipher. In this paper, we consider various symmetric key primitives with ℓ block inputs and raise the following question: *what is the minimum number of block-function invocations required for a mode to be secure?* We begin with encryption modes that generate ℓ' block outputs and show that at least $(\ell + \ell' - 1)$ block-function invocations are necessary to achieve the PRF security. In presence of a nonce, the requirement of block-functions reduces to ℓ' blocks only. If $\ell = \ell'$, in order to achieve SPRP security, the mode requires at least 2ℓ many block-function invocations. We next consider length preserving r-block (called *chunk*) online encryption modes and show that, to achieve online PRP security, each chunk should have at least $2r - 1$ many and overall at least $2r\ell - 1$ many block-functions for ℓ many chunks. Moreover, we show that it can achieve online SPRP security if each chunk contains at least $2r$ non-linear block-functions. We next analyze affine MAC modes and show that an integrity-secure affine MAC mode requires at least ℓ many block-function invocations to process an ℓ block message. Finally, we consider affine mode authenticated encryption and show that in order to achieve INT-RUP security or integrity security under a nonce-misuse scenario, either (i) the number of non-linear block-functions required to generate the ciphertext is more than ℓ or (ii) the number of extra non-linear block-functions required to generate the tag depends on ℓ.

Keywords: Block-cipher, block-function, affine mode, chunk

MSC 2010: 94A40, 68P25, 94A62

Communicated by: Tor Helleseth

1 Introduction

The common application of cryptography is to implement a secure communication between two parties Alice and Bob, who want to exchange information over an insecure channel. In the symmetric key setting, it is assumed that Alice and Bob have a secret shared key, either a one-time shared key or through some key-exchange protocols. They use this key to encrypt or authenticate a message using efficient symmetric-key algorithms such as encryption and message authentication codes or MAC. The encryption provides *privacy* or *confidentiality* (hiding the sensitive message) resulting in a ciphertext, whereas a MAC provides *integrity* or *authenticity* (authenticating the transmitted message M) by generating a tag.

For encryption, the sender takes a message m and the secret key κ, calls the encryption algorithm $\mathcal{E}(\kappa, m)$ to produce a *ciphertext* c and sends it to the receiver. The receiver now executes the decryption algo-

Avik Chakraborti, NTT Secure Platform Laboratories, Tokyo, Japan, e-mail: chakraborti.avik@lab.ntt.co.jp
***Corresponding author: Nilanjan Datta,** Department of Computer Science and Engineering, Indian Institute of Technology, Kharagpur, India, e-mail: nilanjan_isi_jrf@yahoo.com
Mridul Nandi, Applied Statistics Unit, Indian Statistical Institute, Kolkata, India, e-mail: mridul.nandi@gmail.com

rithm $\mathcal{D}(\kappa, c)$ that takes as input the ciphertext c and the secret key κ to recover the plain-text m. Since κ is known only to the sender and the receiver, no one else can succeed in decrypting c with non-negligible probability.

In message authentication codes, the sender takes a plain-text m and the secret key κ and executes the message authentication code algorithm $\mathcal{T}(\kappa, m)$ to generate a fixed length output called a *tag* t. The sender sends the pair (m, t) to the receiver. On receiving a message tag pair (m, t), the receiver uses the MAC algorithm $\mathcal{T}(\kappa, m)$ to generate a tag t' and verify whether t' is same as the received tag t. If so, the receiver is convinced about the authenticity of the message m. Again, the secrecy of the key κ ensures that no one other than the sender or the receiver can succeed in generating a valid tag t, with non-negligible probability.

An authenticated encryption (AE) scheme combines the above two algorithms (encryption and MAC) and provides the privacy as well as the authenticity or data integrity of a plain-text m. Along with the plain-text it can also take associated data (e.g., size and header information) d and guarantees only the integrity of d. The authenticated encryption algorithm $\mathcal{AE}.\mathcal{E}_k(n^*, d, m)$ takes as input a plain-text m, a nonce n^* and an associated data d (optional) and returns a ciphertext c along with a tag t. Sometimes we use the notion tagged-ciphertext to refer to (c, t), the ciphertext-tag pair together. The decryption algorithm $\mathcal{AE}.\mathcal{D}_\kappa(n^*, d, (c, t))$ takes a nonce n^*, an associated data d and the tagged-ciphertext (c, t). It returns the corresponding plain-text m if the tagged-ciphertext is valid or an abort symbol $\perp$ (meaning the tagged-ciphertext is not valid).

1.1 Affine mode of operation

An affine mode of operation is defined by an oracle algorithm which interacts with block-functions (usually keyed) as oracles such that all inputs to the block-functions are computed through some public affine functions (determined by the design) of the message and previous obtained responses of the block-functions. Finally, the output is also computed through another public affine function of the message and all the block-function outputs. This class covers a wide range of encryption, MAC and authenticated encryption algorithms.

Popular symmetric key enciphering schemes like Luby–Rackoff (LR) [17], Feistel type encryption schemes [3, 11], CMC [9], EME [7, 10] HCTR [6, 24], TET [8], HEH [23] etc. belong to affine modes of operation. Online constructions like HCBC [1], MHCBC [18], TC3 [22] etc. are also examples of affine modes of operation. Almost all the MAC functions (e.g., CBC-MAC [2], PMAC [4], TMAC [16], OMAC [12], DAG-based constructions [14], a sub-class of affine domain extension or ADE [19] etc.) are also covered by this approach. Moreover, most of the popular AE schemes like CCM, GCM, OCB or ELmD also use the affine modes. Thus, the affine modes of operation includes a wide class of symmetric key algorithms.

1.2 Our contribution

In Section 3, we begin with the affine mode encryption schemes and show that a PRF secure affine mode encryption requires at least $\ell + \ell' - 1$ many non-linear computations that process an ℓ block message and produces ℓ' block ($\ell' \geq \ell$) ciphertext. Moreover, if the construction is nonce based, then the number of block-functions required to achieve PRP comes down to only ℓ'. If we consider length-preserving encryption (i.e. $\ell' = \ell$) and we want SPRP security, then at least 2ℓ many block-functions are necessary.

Then we consider r-block affine online modes in Section 4. A sequence of r consecutive blocks are called chunks. Informally, an r-block affine online mode has the chunk-wise online property, meaning if two messages are identical up to the i-th chunk, then their i-th ciphertext chunks will be identical. In this work, we show that an r-block online encryption mode can achieve online PRP security if for any message of ℓ chunks it uses at least $2r\ell - 1$ many block-function invocations. Moreover, each chunk requires at least $2r - 1$ many block-functions and any two chunks with $2r - 1$ many block-functions must have at least one chunk with more than $2r$ many block-functions in between. We also show that in order to achieve the online SPRP security, each chunk requires at least $2r$ many non-linear computations, and hence to process a message of ℓ chunks, at least $2r\ell$ many non-linear computations are necessary.

Next, in Section 5, we consider linear MAC modes and show that an integrity-secure linear MAC mode requires at least ℓ many non-linear computations to process a message of ℓ-blocks. For all these cases, we have shown tightness of the bound.

Finally, we consider authenticated encryption modes in Section 6 and show that in order to achieve INT-RUP security and integrity security under a nonce-misuse scenario, either (i) the number of non-linear block-functions required to generate the ciphertext is more than ℓ or (ii) the number of additional non-linear block-functions required to generate the tag depends on ℓ.

1.3 Published and new contents

In [20], Nandi proved that a length preserving PRF (PRP) secure linear mode encryption requires at least $2\ell - 1$ many (respectively 2ℓ many) non-linear computations to process an ℓ block message. We have extended the results from linear modes to affine modes and from length preserving encryptions to length preserving as well as extending encryptions.

Section 4 is dedicated to find the optimal number of non-linear computations to process an ℓ chunk (a chunk is a sequence of r-blocks) messages for r-online affine modes. Section 5 considers affine MAC modes and we find the optimal number of non-linear computations to process ℓ blocks of MAC modes. These results are fresh and not published.

In Section 6, we mainly focus on finding the optimal number of non-linear computations to process an ℓ block message for an affine mode authenticated encryption scheme. The main result of the chapter is the insecurity of rate-1 authenticated encryptions in INT-RUP as well as in a nonce-misuse scenario. This result was published in [5].

2 Preliminaries

2.1 Notations

Let n be a positive integer. A block is defined as an element of the set $\mathbb{B} := \{0,1\}^n$. Any element $x \in \mathbb{B}^\ell$ is an ℓ-tuple $(x_1, \ldots, x_\ell)$. The number of blocks in x is denoted by $\ell := \|x\|$. We represent a tuple of blocks $(x_a, \ldots, x_b)$ by $x_{a\ldots b}$ and $(x_1, \ldots, x_a)$ by $x_{\ldots a}$. For any two ℓ-tuples $x = (x_1, \ldots, x_\ell)$ and $y = (y_1, \ldots, y_\ell)$, we denote by $x \oplus y$ the ℓ-tuple $(x_1 \oplus y_1, \ldots, x_\ell \oplus y_\ell)$.

Let $\mathbb{M}_{a\times b}$ be an $a \times b$ matrix with elements from $\mathbb{B}$. Then we denote the element in the i-th block-row and the j-th block-column by $M[i,j]$. By $M[i,*]$ and $M[*,j]$, we denote the i-th row and j-th column, respectively. For $1 \le i \le j \le a$, we also write $M[i \ldots j\ ;\ *]$ to mean the sub-matrix consisting of all rows between i and j. We simply write $M[\ldots j\ ;\ *]$ and $M[i \ldots\ ;\ *]$ to denote $M[1 \ldots j\ ;\ *]$ and $M[i \ldots a\ ;\ *]$, respectively. We define similar notation for columns. Suppose $\mathbb{M}_n(a,b)$ denotes the set of all partitioned matrices $M_{a\times b}$ (of size $a \times b$ as a block partitioned matrix and of size $an \times bn$ as a binary matrix), where $M[i,j]$ is a block-matrix for all $i \in \{1 \ldots a\}$ and $j \in \{1 \ldots b\}$. Let M and N be two matrices with the same number of rows. Then the concatenation of M and N is denoted by $[M : N]$. We use the notation $x\|y$ to denote the concatenation of two bit strings x and y. For a set S, we let $S^+ = \bigcup_{i=1}^{\infty} S^i$, $S^{\le n} = \bigcup_{i=1}^{n} S^i$ and $S^* = S^+ \cup \{\lambda\}$, where λ is the empty string. The usual choice of S is $\{0,1\}$. Let v^{tr} denote the transpose of a vector v.

2.2 Security definitions

In this section, we discuss various security notions of keyed functions. We call an oracle algorithm $\mathcal{A}$ a (t', q')-algorithm if it makes at most q' queries and runs in time t'. For notational simplicity, we skip the time parameter t' which is irrelevant in this paper. We also simply write $\mathrm{Func}(\ell, \ell')$ and $\mathrm{Perm}(\ell)$ to denote

the set of all functions from $\mathbb{B}^\ell$ to $\mathbb{B}^{\ell'}$ and permutations over $\mathbb{B}^\ell$, respectively. Throughout this paper, we assume $\mathcal{K}$ to be the key-space.

Definition 2.1 (PRF). Let $F : \mathcal{K} \times \mathbb{B}^\ell \to \mathbb{B}^{\ell'}$ be a (keyed) function. We say that F is (q_e, ϵ)-PRF if for any q_e-algorithm $\mathcal{A}$ the *prf-distinguishing advantage*

$$\mathbf{Adv}_F^{\mathrm{prf}}(\mathcal{A}) := \left|\mathsf{Pr}[\mathcal{A}^{F_\kappa} = 1; \kappa \xleftarrow{\$} \mathcal{K}] - \mathsf{Pr}[\mathcal{A}^{G} = 1; G \xleftarrow{\$} \mathrm{Func}(\ell, c)]\right|$$

is at most ϵ. We call a randomly chosen G to be the (uniform) *random function*.

Definition 2.2 ((S)PRP). A keyed permutation F over $\mathbb{B}^\ell$ is a function $F : \mathcal{K} \times \mathbb{B}^\ell \to \mathbb{B}^\ell$ such that for all keys $\kappa \in \mathcal{K}$ one has $F_\kappa := F(\kappa, \cdot) \in \mathrm{Perm}(\ell)$. A keyed permutation F is called (q_e, ϵ)-PRP if for any q_e-algorithm $\mathcal{A}$ the *prp-distinguishing advantage*

$$\mathbf{Adv}_F^{\mathrm{prp}}(\mathcal{A}) := \left|\mathsf{Pr}[\mathcal{A}^{F_\kappa} = 1; \kappa \xleftarrow{\$} \mathcal{K}] - \mathsf{Pr}[\mathcal{A}^{\Pi} = 1; \Pi \xleftarrow{\$} \mathrm{Perm}(\ell)]\right|$$

is at most ϵ. We call a randomly chosen Π to be the (uniform) *random permutation*. Similarly, a keyed permutation F is called (q_e, ϵ)-SPRP if for any q_e-algorithm $\mathcal{A}$ the *sprp-distinguishing advantage*

$$\mathbf{Adv}_F^{\mathrm{sprp}}(\mathcal{A}) := \left|\mathsf{Pr}[\mathcal{A}^{F_\kappa, F_\kappa^{-1}} = 1; \kappa \xleftarrow{\$} \mathcal{K}] - \mathsf{Pr}[\mathcal{A}^{\Pi, \Pi^{-1}} = 1; \Pi \xleftarrow{\$} \mathrm{Perm}(\ell)]\right|$$

is at most ϵ.

Definition 2.3 (Integrity or authenticity). A keyed function F is called (q_e, q_f, ϵ)-forgeable if for any (q_e, q_f)-algorithm $\mathcal{A}$ (an algorithm that makes q many queries and q_f many forging attempts) the *integrity (INT) advantage*

$$\mathbf{Adv}_F^{\mathrm{int}}(\mathcal{A}) := \left|\mathsf{Pr}[\mathcal{A}^{F_\kappa} \text{ forges}; \kappa \xleftarrow{\$} \mathcal{K}]\right|$$

is at most ϵ. We say that $\mathcal{A}$ forges if $\mathcal{A}$ can produce a *valid* message and tag pair $(m^*, f_\kappa(m^*))$ without querying m^* during the encryption queries.

Definition 2.4 (Integrity of ciphertext (INT-CTXT)). A length extending keyed function F is called (q_e, q_f, ϵ)-forgeable if for any (q_e, q_f)-algorithm $\mathcal{A}$ the *integrity of ciphertext (INT-CTXT) advantage*

$$\mathbf{Adv}_F^{\mathrm{int\_ctxt}}(\mathcal{A}) := \left|\mathsf{Pr}[\mathcal{A}^{F_\kappa} \text{ forges}; \kappa \xleftarrow{\$} \mathcal{K}]\right|$$

is at most ϵ. We say that $\mathcal{A}$ forges if $\mathcal{A}$ can produce a *valid* ciphertext $F_K(m^*)$ without querying m^* during the encryption queries.

Definition 2.5 (Integrity of ciphertext under released unverified plaintext). Let D_κ be the decryption algorithm without verification that outputs the plaintext without verification. A length extending keyed function F is called $(q_e, q_d, q_f, \epsilon)$-forgeable if for any (q_e, q_d, q_f)-algorithm $\mathcal{A}$ (an algorithm that makes q_e many encryption, q_d many unverified decryption and q_f many forging queries) the *integrity of ciphertext under release of unverified plaintext (INT-RUP) advantage*

$$\mathbf{Adv}_F^{\mathrm{int\_rup}}(\mathcal{A}) := \left|\mathsf{Pr}[\mathcal{A}^{F_\kappa, D_\kappa} \text{ forges}; \kappa \xleftarrow{\$} \mathcal{K}]\right|$$

is at most ϵ. The definition of forging is the same as used in INT-CTXT notion.

2.3 Useful properties of matrices

It is well known that the maximum numbers of linearly independent (binary) rows and columns of a matrix $A \in \mathbb{M}_n(s, t)$ are the same, and this number is called the rank of the matrix, denoted by $\mathrm{rank}(A)$. It is easy to see that $\mathrm{rank}(A) \le \min\{ns, nt\}$. Consider a system of solvable linear equations $A \cdot x = b$ or $w \cdot A = b$. One can solve the equation for some x or w, respectively, (not necessarily unique) using the Gaussian elimination

method, denoted by solve and solve^*, respectively:

$$x = \mathsf{solve}(A, b), \quad w = \mathsf{solve}^*(A, b).$$

By convention, whenever a non-zero solution exists, it returns a non-zero solution. Note that if

$$w^{\mathrm{tr}} = \mathsf{solve}(A^{\mathrm{tr}}, b^{\mathrm{tr}}),$$

then $w \cdot A = b$ (by applying the transpose). Now we state two important results.

Lemma 2.6. *Let $A \in \mathbb{M}_n(s, t)$ and $r := \mathrm{rank}(A)$.*
(i) *If $r < ns$ (i.e. presence of row-dependency), then $\mathsf{solve}(A^{\mathrm{tr}}, 0)$ returns a non-zero solution.*
(ii) *Similarly, for $r < nt$ (i.e. presence of column-dependency), $\mathsf{solve}(A, 0)$ returns a non-zero solution.*
(iii) *Finally, if $r = nt$ (i.e. full column rank) and $b := A \cdot w$, then $\mathsf{solve}(A, b)$ returns a unique solution.*

Lemma 2.7. *Suppose $A \in \mathbb{M}_n(s, s)$ is a non-singular matrix, i.e. rank(A) = ns. Let $t < s$ and*

$$B = \begin{pmatrix} A[\ldots t, *] & \mathbf{0} \\ \mathbf{0} & A[\ldots t, *] \\ A[t+1\ldots, *] & A[t+1\ldots, *] \end{pmatrix},$$

where $\mathbf{0}$ denotes the zero matrix of appropriate size. Then $\mathrm{rank}(B) = n(s + t)$ (i.e. full row-rank).

As the results are straightforward, we skip the proofs.

3 Optimality for PRF and SPRP in affine modes

In this section, we find the tight lower bound on the number of invocations of building blocks for achieving PRP and SPRP security as in Table 1.

To do so, we first provide the formal description of an affine mode encryption scheme.

3.1 Formal definition of affine mode encryption

An affine mode encryption scheme

$$\mathcal{E}_{\ell,\ell',q,\widetilde{\Pi},\kappa'} := (\mathcal{E}_1, \mathcal{E}_2),$$

parameterized by a family of q many permutations $\widetilde{\Pi} = (\pi_1, \ldots, \pi_q)$, takes an ℓ-block message m and an optional nonce n^* and returns an ℓ'-block ciphertext c as follows:

$$\begin{aligned} u_i &= \mathcal{E}_1(v_1, \ldots, v_{i-1}, m, \kappa, n^*), \quad i = 1, \ldots, q, \\ v_i &= \pi_i(u_i), \qquad\qquad\qquad\qquad\quad i = 1, \ldots, q, \\ c &= \mathcal{E}_2(v_1, \ldots, v_q, m, \kappa, n^*). \end{aligned}$$

Here u_i and v_i's are the i-th non-linear input and output, respectively, and

$$\kappa = \begin{pmatrix} 1 \\ \kappa' \end{pmatrix},$$

where $\kappa' := (\kappa_1, \ldots, \kappa_{k-1})$ is the set of masking keys used. The entry 1 ensures that the construction is affine. It is easy to see that the function $\mathcal{E}_{\ell,\ell',q,\widetilde{\Pi},\kappa'}$ can alternatively represented by a $(q + \ell') \times (q + \ell + k + 1)$ encryption co-efficient matrix E (with E_{11} strictly lower triangular) such that

$$\begin{pmatrix} u \\ c \end{pmatrix} = \begin{pmatrix} (E_{11})_{q\times q} & (E_{12})_{q\times \ell} & (E_{13})_{q\times k} & (E_{14})_{q\times 1} \\ (E_{21})_{\ell'\times q} & (E_{22})_{\ell'\times \ell} & (E_{23})_{\ell'\times k} & (E_{24})_{\ell'\times 1} \end{pmatrix} \cdot \begin{pmatrix} v \\ m \\ \kappa \\ n^* \end{pmatrix},$$

Nonce	Security	Optimal number of block-functions	Examples with tight bound
No	PRF	$(\ell + \ell' - 1)$	3 round LR
No	SPRP	2ℓ	CMC
Yes	PRF	ℓ'	OCB

Table 1: Optimal block-functions required for affine mode encryption with ℓ blocks input and c blocks output (where $\ell' \geq \ell$).

where $\pi_i(u_i) = v_i$. The function needs to be decryptable, meaning there should exist a corresponding $(q + \ell') \times (q + \ell + k + 1)$ decryption matrix D, $\alpha := (\alpha_1, \ldots, \alpha_q) \in \{1, -1\}^q$ and a permutation β on $[1 \ldots q]$ such that

$$\begin{pmatrix} x \\ m \end{pmatrix} = \begin{pmatrix} (D_{11})_{q\times q} & (D_{12})_{q\times \ell} & (D_{13})_{q\times k} & (D_{14})_{q\times 1} \\ (D_{21})_{\ell'\times q} & (D_{22})_{\ell'\times \ell} & (D_{23})_{\ell'\times k} & (D_{24})_{\ell'\times 1} \end{pmatrix} \cdot \begin{pmatrix} y \\ c \\ \kappa \\ n^* \end{pmatrix},$$

where $\pi^{\alpha_i}_{\beta(i)}(x_i) = y_i$ with the property

$$(u_i, v_i) = \begin{cases} (x_{\beta(i)}, y_{\beta(i)}) & \text{if } \alpha_i = 1, \\ (y_{\beta(i)}, x_{\beta(i)}) & \text{if } \alpha_i = -1. \end{cases}$$

Now consider a length-preserving encryption scheme $\mathcal{E}_{\ell,\ell',q,\overline{\Pi},K}$. Until and unless specified, we will assume the encryption scheme does not use any nonce, i.e. $E_{14} = E_{24} = D_{14} = D_{24} = 0$. In the following two subsections, we will show PRF and SPRP distinguishing attacks on this scheme if $q \leq (\ell + \ell' - 2)$ and $q \leq (2\ell - 1)$, respectively. The tightness of the bound is achieved by 3-round Luby–Rackoff (which uses three block-functions to process two blocks and achieves PRP) and CMC (uses 2ℓ block-functions to process an ℓ block message and achieves SPRP). Moreover, we will show that under distinct nonce the optimal number of block-functions required to make $\mathcal{E}_{\ell,t,q,\overline{\Pi},K}$ secure drastically reduces to $q = \ell$.

3.2 PRF distinguishing attack on $\mathcal{E}_{\ell,\ell',q,\overline{\Pi},\kappa'}$ for $q \leq \ell + \ell' - 2$

Here we provide the attack assuming $q = (\ell + \ell' - 2)$. The attack can be trivially extended to all such constructions with $q < (\ell + \ell' - 2)$.

Description of the PRF Distinguisher $\mathcal{A}_{\mathsf{prf}}$.

(step 1) (Finding a suitable difference in a pair of plaintext queries): Find $\delta \in \mathbb{B}^\ell$:

$$\delta = \mathsf{solve}(E_{12}[\ldots(\ell - 1)\,;\,*], 0^{\ell-1}).$$

(step 2) (Make the queries with the difference obtained in (step 1)): Now the distinguisher makes two ℓ block queries 0^ℓ and δ and obtains corresponding responses $c = \mathcal{E}(0^\ell)$ and $c' = \mathcal{E}(\delta)$. For simplicity, we ignore the parameter set of $\mathcal{E}$ as it is clear from the context.

(step 3) (Find a nullifier of unknown intermediate values): Find $\delta' \in \mathbb{B}^{\ell'}$:

$$\delta' = \mathsf{solve}^*(E_{21}[*\,;\,\ell \ldots (\ell + \ell' - 2)], 0^{\ell'-1}).$$

(step 4) (The distinguisher event): If $\delta' \cdot \Delta c = \delta' \cdot E_{22} \cdot \delta$, then it returns 1 (decision for the keyed construction), else it returns 0 (decision for uniform random permutation).

Brief explanation. In (step 1) and (step 3), we can find such a non-zero δ and δ', respectively, using Lemma 2.6. Now let $u_1, v_1, \ldots, u_{\ell+\ell'-2}, v_{\ell+\ell'-2}$ and $u'_1, v'_1, \ldots, u'_{\ell+\ell'-2}, v'_{\ell+\ell'-2}$ be the intermediate inputs-outputs for the two queries 0 and δ, respectively. From Lemma 2.7 we have $1 \leq i \leq (\ell - 1)$, $u_i = u'_i$ and $v_i = v'_i$, i.e. $\Delta v_{1\ldots(\ell-1)} = 0$, and hence

$$\delta' \cdot \Delta c = \delta' \cdot E_{21}[*\,;\,\ell \ldots (\ell + \ell' - 2)] \cdot \Delta v_{\ell\ldots(\ell+\ell'-2)} + \delta' \cdot E_{22} \cdot d = \delta' \cdot E_{22} \cdot d.$$

The distinguishing advantage of the above distinguisher $\mathcal{A}_{\mathsf{prf}}$ is at least $\frac{1}{2}$ since for a random permutation the event $\delta' \cdot \Delta c = \delta' \cdot E_{22} \cdot d$ holds with probability $\frac{1}{2}$, whereas for the keyed construction it holds with probability 1.

A generalized distinguisher $\mathcal{A}^{\mathsf{gen}}_{\mathsf{prf}}$ on $\mathcal{E}_{\ell,\ell',q,\bar{\Pi},\kappa'}$. Now we define a generalized distinguisher against this mode assuming there exists an integer ℓ^* such that

$$\mathrm{rank}(E_{12}[\ldots \ell^*\,;\,*]) < n\ell \quad \text{and} \quad \mathrm{rank}(E_{21}[*\,;\,(\ell^*+1)\ldots q]) < n\ell'.$$

Note that the above assumptions always hold for $\ell^* = (\ell - 1)$ when $q \le (\ell + \ell' - 2)$. However, if $q \ge (\ell + \ell' - 1)$, both conditions do not necessarily hold. Whenever the assumptions hold, we have the following similar distinguisher.

Distinguisher $\mathcal{A}^{\mathsf{gen}}_{\mathsf{prf}}$ against $\mathcal{E}_{\ell,\ell,q,\bar{\Pi},\kappa'}$.

(step 1) Due to our assumptions, we can find $\delta \in \mathbb{B}^{\ell}$ and $\delta' \in \mathbb{B}^{\ell'}$:

$$\delta = \mathsf{solve}(E_{12}[\ldots t\,;\,*], 0^{\ell-1}), \quad \delta' = \mathsf{solve}^*(E_{21}[*\,;\,(t+1)\ldots q], 0^{\ell'-1}).$$

(step 2) Then we make two queries 0 and δ and obtain responses c and c'.
(step 3) The distinguisher returns 1 if $\delta' \cdot \Delta c = \delta' \cdot E_{22} \cdot \delta$, else 0.
This distinguisher will be used later while describing SPRP distinguishers.

3.3 SPRP distinguishing attack on $\mathcal{E}_{\ell,\ell,q,\bar{\Pi},\kappa'}$ with $q \le 2\ell - 1$

Now we will show that if $q = 2\ell - 1$, then we have an SPRP distinguisher. Again, the attack can be trivially extended to all such constructions with $q \le 2\ell - 1$.

A Rough Sketch of the Distinguisher. Here we provide a brief idea on how the distinguisher works. The distinguisher first makes two queries so that the first $(\ell - 1)$ intermediate inputs and outputs are identical for two encryption queries. Using the invertible property, we can actually obtain all the differences in the intermediate values. As the computation of the decryption algorithm must use the same internal inputs and outputs of the building blocks, we also know the differences of intermediate inputs and outputs if we decrypt the first two encryption queries. Now we find another decryption query for which the first ℓ intermediate input and output differences with one of the first two queries are fixed. So we can nullify the unknown $(\ell - 1)$ differences and obtain a distinguishing event. The formal attack details are described below: For this attack, we first assume the following:

$$E_{21}[\ell \ldots (2\ell-1)\,;\,*] \text{ and } D_{11}[\ldots \ell\,;\,*] \text{ are invertible.}$$

Observe that, if $\mathrm{rank}(E_{21}[\ell \ldots (2\ell-1)\ldots\,;\,*]) < n\ell$ or $\mathrm{rank}(D_{11}[\ldots \ell\,;\,*]) < n\ell$, then the two assumptions mentioned in the generalized distinguisher $\mathcal{A}^{\mathsf{gen}}_{\mathsf{prf}}$ hold for $t = \ell - 1$ and we can run the PRF-distinguisher $\mathcal{A}^{\mathsf{gen}}_{\mathsf{prf}}$. So, we assume that $E_{21}[\ell \ldots (2\ell-1)\,;\,*]$ and $D_{11}[\ldots \ell\,;\,*]$ are invertible. Now we provide the algorithmic description of the distinguisher.

Description of the SPRP-distinguisher $\mathcal{A}_{\mathsf{sprp}}$.

(step 1) (Find a suitable difference in a pair of plaintext queries): Find $\delta = \mathsf{solve}(E_{12}[\ldots(\ell-1)\,;\,*], 0^{\ell-1})$.
(step 2) (Make the queries with the difference obtained in (step 1)): Now the distinguisher makes two queries 0 and δ and obtains corresponding responses $c = \mathcal{E}(0^{\ell})$ and $c' = \mathcal{E}(\delta)$, respectively. Let $u_1, v_1, \ldots, u_{2\ell-1}, v_{2\ell-1}$ and $u'_1, v'_1, \ldots, u'_{2\ell-1}, v'_{2\ell-1}$ respectively denote the intermediate block-cipher inputs and outputs of the two queries. It is easy to see that $1 \le i \le (\ell - 1)$, $u_i = u'_i$, $v_i = v'_i$ and

$$\Delta c := c + c' = E_{21}[\ell \ldots (2\ell-1)\,;\,*] \cdot \Delta v_{\ell\ldots} + E_{22} \cdot \delta.$$

(step 3) (Solve Δu, Δv for the 2-queries): Using the invertible property of $E_{21}[\ell \ldots (2\ell-1)\,;\,*]$, we can actually solve $\Delta v_{\ell\ldots}$ and hence $\Delta u_{\ell\ldots}$. Thus, we know Δu and Δv.

(step 4) (Find Δx, Δy for the 2-queries): Suppose we make two (redundant) decryption queries c and c' (whose responses must be 0 and d) and let $x_1, y_1, \ldots, x_{2\ell-1}, y_{2\ell-1}$ and $x'_1, y'_1, \ldots, x'_{2\ell-1}, y'_{2\ell-1}$ respectively denote the intermediate inputs and outputs for the two queries. Then by the definition of the decryption algorithm we also know Δx and Δy, which are nothing but a (β, π)-reordering of $(\Delta u, \Delta v)$.

(step 5) (Find a difference for the final decryption query): Now we find a difference δ' such that

$$(D_{11}[\ldots \ell\,; *]\; D_{12}[\ldots \ell\,; *]) \cdot \begin{pmatrix} \delta' \\ \Delta y_1 \end{pmatrix} = \begin{pmatrix} \Delta x_1 \\ 0^{\ell-1} \end{pmatrix}.$$

We can solve the above equation for a non-zero δ', assuming that $\Delta x_1 \neq 0$ (since otherwise we do not get a non-zero δ'. Note that Δx_1 can be written as a function of c and c'. So for a random permutation, a function of c and c' becoming zero has low probability). Moreover, we already have the assumption that $D_{11}[\ldots \ell\,; *]$ is invertible.

(step 6) (Make the queries with the difference obtained in (step 5)): Now we make two decryption queries $\bar{c}$ and $\bar{c}'$, where $\bar{c}' = \bar{c} + \delta'$, with the first block-cipher input being x_1 and x'_1, respectively. While we set two queries, we should ensure that none of these have been obtained in the first two encryption queries (these are also called non-pointless or non-trivial queries). Let $x_1, y_1, \bar{x}_2, \bar{y}_2, \ldots, \bar{x}_{2\ell-1}, \bar{y}_{2\ell-1}$ and $x'_1, y'_1, \bar{x}'_2, \bar{y}'_2, \ldots, \bar{x}'_{2\ell-1}, \bar{y}'_{2\ell-1}$ respectively denote the intermediate inputs and outputs for these two queries, and let $\bar{M}$ and $\bar{M}'$ denote the corresponding responses. By the choice of δ', we know that $\Delta\bar{y}_{2\ldots\ell} = 0^{\ell-1}$.

(step 7) (Find a nullifier of unknown intermediate values, same as PRP distinguisher): Find

$$\delta^* = \mathsf{solve}^*(D_{21}[*, \ell+1\ldots], 0^{\ell-1}).$$

(step 8) (The distinguisher event): If $\delta^* \cdot \Delta\bar{M} = \delta^* \cdot D_{22} \cdot d'$, then it returns 1 (decision for the keyed construction), else it returns 0 (decision for uniform random permutation).

3.4 PRF distinguishing attack on nonce-based $\mathcal{E}_{\ell,\ell',q,\tilde{\Pi},\kappa'}$ with $q \leq \ell' - 1$

Here we provide the attack assuming $q = \ell' - 1$. The attack can be trivially extended to all those constructions with $q < \ell' - 1$.

Description of the PRF-distinguisher $\mathcal{A}_{\mathsf{n\_prf}}$.

(step 1) (Finding a suitable difference in a pair of plaintext queries): Let δ' be an ℓ block non-zero vector. Find $\delta = \mathsf{solve}(E_{12}[\ldots \ell'\,; *], E_{14} \cdot \delta')$.

(step 2) (Make the queries with the difference obtained in (step 1)): Now the distinguisher makes two queries $(n^*, 0^\ell)$ and $(n^* + \delta', \delta)$ and obtains corresponding responses $c = \mathcal{E}(n^*, 0^\ell)$ and $c' = \mathcal{E}(n^* + \delta', \delta)$.

(step 3) (The distinguisher event): If $\Delta c = (E_{22} \cdot \delta + E_{24} \cdot \delta')$, then it returns 1 (decision for the keyed construction), else it returns 0 (decision for uniform random function).

3.5 Optimal number of block-functions for affine mode enciphering schemes

Here we provide the optimal number of block-functions required for PRF and SPRP security for length preserving encryption schemes and PRF security for length expanding encryption schemes.

Theorem 3.1. *An affine mode encryption that takes a message of length ℓ blocks (and outputs ℓ' blocks), requires at least $2\ell - 1$ many block-functions to achieve PRP security. If we consider a length-preserving encryption (i.e. $\ell' = \ell$), then it requires at least 2ℓ block-function calls to achieve SPRP security.*

Proof. The result follows from the description of $\mathcal{A}_{\mathsf{prp}}$ and $\mathcal{A}_{\mathsf{sprp}}$. □

Theorem 3.2. *An affine mode encryption with nonce (non-repeating) that takes a message of length ℓ blocks (and outputs ℓ' blocks) requires at least ℓ' many block-functions to achieve PRF security.*

Proof. The result follows from the description of $\mathcal{A}_{\text{n\_prf}}$. □

The optimality is achieved with a simple variant of 3-round Luby Rackoff (PRF with $\ell' = \ell = 2$), CMC (SPRP with any ℓ) and OCB (nonce-based PRP with any ℓ and $\ell' = \ell + 1$).

4 Optimality for online PRF and online SPRP in affine modes

In this section, we concentrate on the affine mode online encryption. First, we provide the formal definition of it, then we find some important properties and use them show the results in Table 2.

Security	Optimal number of total block-functions	Optimal number of block-functions per chunk	Examples with tight bound
online PRF	$2r\ell - 1$	$2r - 1$	A variant of HCBC-1
online SPRP	$2r\ell$	$2r$	TC3

Table 2: Optimal block-functions required for length-preserving r-online encryption. Here we consider the length of a message to be ℓ many r-blocks.

4.1 r-block online encryption mode

Tweak-updatable chunk-wise encryption. We call r-blocks a chunk. In this section, we describe tweak-updatable chunk-wise encryption, the basic building block of r-block online affine mode. Given r, the basic building block $\mathcal{E}_{q,\widetilde{\Pi}} := (\mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3)$ (parameterized by a family of q many permutations $\widetilde{\Pi} = (\pi_1, \ldots, \pi_q)$) takes an r-block message m and a tweak value w of t blocks and returns an r-block ciphertext c and an updated tweak w' as follows:

$$\begin{aligned} u_i &= \mathcal{E}_1(v_1, \ldots, v_{i-1}, m, w, \kappa), \quad && i = 1, \ldots, q, \\ v_i &= \pi_i(u_i), && i = 1, \ldots, q, \\ c &= \mathcal{E}_2(v_1, \ldots, v_q, m, w, \kappa), \\ w' &= \mathcal{E}_3(v_1, \ldots, v_q, m, w, \kappa). \end{aligned}$$

It is easy to see that the function $\mathcal{E}_{q,\widetilde{\Pi}}$ can alternatively be represented by a $(q + r + t) \times (q + r + t + k)$ encryption co-efficient matrix E (with E_{11} strictly lower triangular) such that

$$\begin{pmatrix} u \\ c \\ w' \end{pmatrix} = \begin{pmatrix} (E_{11})_{q\times q} & (E_{12})_{q\times r} & (E_{13})_{q\times t} & (E_{14})_{q\times k} \\ (E_{21})_{r\times q} & (E_{22})_{r\times r} & (E_{23})_{r\times t} & (E_{24})_{r\times k} \\ (E_{31})_{t\times q} & (E_{32})_{t\times r} & (E_{33})_{t\times t} & (E_{34})_{t\times k} \end{pmatrix} \cdot \begin{pmatrix} v \\ m \\ w \\ \kappa \end{pmatrix},$$

where $\pi_i(u_i) = v_i$. The function needs to be decryptable, meaning there should exist a corresponding $(q + r + t) \times (q + r + t + k)$ decryption matrix D, $\alpha := (\alpha_1, \ldots, \alpha_q) \in \{1, -1\}^q$ and a permutation β on $[1 \ldots q]$ such that

$$\begin{pmatrix} x \\ m \\ w' \end{pmatrix} = \begin{pmatrix} (D_{11})_{q\times q} & (D_{12})_{q\times r} & (D_{13})_{q\times t} & (D_{14})_{q\times k} \\ (D_{21})_{r\times q} & (D_{22})_{r\times r} & (D_{23})_{r\times t} & (D_{24})_{r\times k} \\ (D_{31})_{t\times q} & (D_{32})_{t\times r} & (D_{33})_{t\times t} & (D_{34})_{t\times k} \end{pmatrix} \cdot \begin{pmatrix} y \\ c \\ w \\ \kappa \end{pmatrix},$$

where $\pi^{\alpha_i}_{\beta(i)}(x_i) = y_i$ with the property

$$(u_i, v_i) = \begin{cases} (x_{\beta(i)}, y_{\beta(i)}) & \text{if } \alpha_i = 1, \\ (y_{\beta(i)}, x_{\beta(i)}) & \text{if } \alpha_i = -1. \end{cases}$$

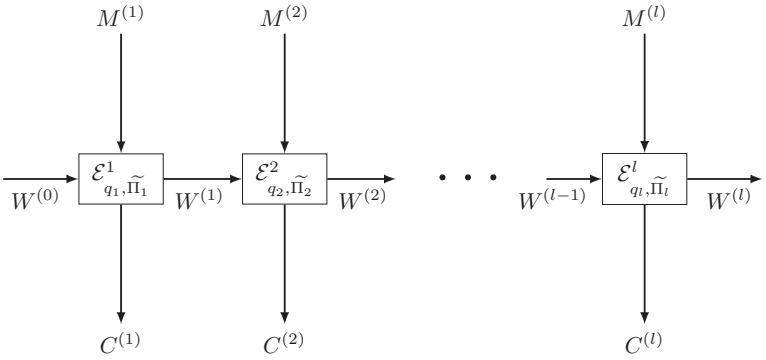

Figure 1: r-block linear online encryption mode.

***r*-block online encryption mode.** The r-block online encryption takes a tweak τ and a message M of ℓ chunks (each containing r blocks), and computes the ciphertext C of length ℓ as follows:

$$w^{(0)} = \tau,$$
$$(c^{(i)}, w^{(i)}) = \mathcal{E}^i_{q_i,\widetilde{\Pi}_i}(m^{(i)}, w^{(i-1)}) \quad \text{for } i = 1, \ldots, \ell.$$

It is easy to see that an r-block online encryption mode is defined by the sequence of $\mathcal{E}^i_{q_i,\widetilde{\Pi}_i}$ matrices seen in Figure 1.

Examples. HCBC [1], MHCBC [18] and TC3 [22] are some examples of linear online encryption modes with $r = 1$.

4.2 Finding tweak difference Δw_i for two messages

Let $\mathcal{E}^i_{q_i,\widetilde{\Pi}}$ be the i-th chunk-wise encryption. Suppose Δw_{i-1} (the state difference for two messages m_1 and m_2 of $(i-1)$ chunks) is known. We will show that if $\Delta w_{i-1} = 0$, then $q_i \geq 2r - 1$, and if Δw_{i-1} is non-zero, then $q_i \geq 2r$.

Lemma 4.1. *Consider a chunk-wise encryption $\mathcal{E}^i_{2r,\widetilde{\Pi}}$. Given a non-zero Δw_{i-1} for two messages, one can find two messages for which Δw_i is computable.*

Proof. Given a non-zero Δw, we find Δm and calculate the corresponding $\Delta w'$ as follows:

(i) Find Δm, a suitable difference in a pair of plaintext queries:

$$\Delta m = \mathsf{solve}(E^i_{12}[\ldots r\,;\ *], E^i_{13} \cdot \Delta w).$$

We can find such a Δm given $E^i_{13} \cdot \Delta w$ is non-zero. Now we set $m_1 = 0^r$ and $m'_1 = \Delta m$ and make queries to $\mathcal{E}^i_{2r,\widetilde{\Pi}}$. Let the i-th ciphertext chunks be c_1 and c'_1.

(ii) Find a nullifier $\delta^* \in \mathbb{B}^{r+1}$ of unknown intermediate values:

$$\delta^* = \mathsf{solve}^*(E^*, 0^r), \quad \text{where } E^* = \begin{pmatrix} E^i_{21}[\ldots r\,;\ (r+1)\ldots 2r] \\ E^i_{31}[1;\ (r+1)\ldots 2r] \end{pmatrix}.$$

(iii) Find the updated tweak difference $\Delta w'$ from the equation

$$\delta^* \cdot \begin{pmatrix} \Delta c \\ \Delta w' \end{pmatrix} = \delta^* \cdot \begin{pmatrix} E^i_{22} \\ E^i_{32} \end{pmatrix} \cdot \Delta M + \delta^* \cdot \begin{pmatrix} E^i_{23} \\ E^i_{33} \end{pmatrix} \cdot \Delta w. \tag{4.1}$$

Explanation. Let $u_1, v_1, \ldots, u_{2r}, v_{2r}$ and $u'_1, v'_1, \ldots, u'_{2r}, v'_{2r}$ respectively denote the intermediate inputs and outputs for the two queries 0^r and Δm. Now we have $1 \leq i \leq r$, $u_i = u'_i$, $v_i = v'_i$ and

$$\begin{pmatrix} \Delta c \\ \Delta w' \end{pmatrix} = \begin{pmatrix} E^i_{21}[\ldots r\,;\ (r+1)\ldots 2r] \\ E^i_{31}[1;\ (r+1)\ldots 2r] \end{pmatrix} \cdot \Delta v_{(r+1)\ldots} + \begin{pmatrix} E^i_{22} \\ E^i_{32} \end{pmatrix} \cdot \Delta m + \begin{pmatrix} E^i_{23} \\ E^i_{33} \end{pmatrix} \cdot \Delta w.$$

Multiplying both sides of the equation by δ^*, we obtain equation (4.1). □

Corollary 4.2. *Consider a chunk-wise encryption $\mathcal{E}^i_{2r-1,\widetilde{\Pi}}$. Given $\Delta w = 0$, one can find Δm for which $\Delta w'$ is computable.*

Proof. The proof is similar to the one of the previous lemma. For completeness, we provide the sketch of the proof. First, we find

$$\Delta m = \mathsf{solve}(E^i_{12}[\ldots(r-1)\,;\,*], 0^{r-1})$$

and make queries with $m_1 = 0^r$ and $m_2 = \Delta m$. Let the corresponding responses be c_1 and c_2. Then we find the nullifier $\delta^* \in \mathbb{B}^{r+1}$ similar to the previous proof:

$$\delta^* = \mathsf{solve}^*(E^*, 0^r), \quad \text{where } E^* = \begin{pmatrix} E^i_{21}[\ldots r\,;\,(r+1)\ldots 2r] \\ E^i_{31}[1;\,(r+1)\ldots 2r] \end{pmatrix}.$$

Now, one can find $\Delta w'$ from the equation

$$\delta^* \cdot \begin{pmatrix} \Delta c \\ \Delta w' \end{pmatrix} = \delta^* \cdot \begin{pmatrix} E^i_{22} \\ E^i_{32} \end{pmatrix} \cdot \Delta m. \qquad \square$$

4.3 Finding minimum number of block-functions for PRF security of the *i*-th tweak updatable chunk-wise encryption

Lemma 4.3. *Given a non-zero Δw, the chunk-wise encryption $\mathcal{E}^i_{2r-1,\widetilde{\Pi}}$ is not PRF.*

Proof. Here we describe an adversary $\mathcal{A}'_{\mathsf{prf}}$ that distinguishes $\mathcal{E}^i_{2r-1,\widetilde{\Pi}}$ from a random function, given a non-zero tweak value difference Δw.

Description of the PRF-distinguisher $\mathcal{A}'_{\mathsf{prf}}$.

(step 1) (Find the suitable difference in a pair of queries): Find $\Delta m \in \mathbb{B}^r$:

$$\Delta m = \mathsf{solve}(E^i_{12}[\ldots r\,;\,*], E^i_{13} \cdot \Delta w).$$

(step 2) (Make queries with the difference obtained in (step 1)): Now the distinguisher makes two queries with m_1 and $(m_1 + \Delta m)$ and obtains corresponding responses c_1 and c_2.

(step 3) (Find the nullifier of the unknown intermediate values): By using Lemma 2.6, find $\delta^* \in \mathbb{B}^r$:

$$\delta^* = \mathsf{solve}^*(E^i_{21}[\ldots r\,;\,r\ldots(2r-1)], E^i_{23} \cdot \Delta w).$$

(step 4) (The distinguisher event): If $\delta^* \cdot (c_1 + c_2) = \delta^* \cdot E^i_{22} \cdot \Delta m + \delta^* \cdot E^i_{23} \cdot \Delta w$, then it returns 1 (decision for the keyed construction), else it returns 0 (decision for uniform random permutation). $\square$

By using a similar idea, one can also show the following proposition.

Proposition 4.4. *Given $\Delta w = 0$, the chunk-wise encryption $\mathcal{E}^i_{2r-2,\widetilde{\Pi}}$ is not PRF.*

So, we observed that if a non-zero input tweak difference can be observed, then a tweak updatable chunk-wise encryption requires at least $2r$ many block-functions. Similarly, if only zero tweak difference is observed, then it requires at least $2r - 1$ many block-functions.

4.4 Finding the minimum number of block-functions for SPRP security of tweak updatable chunk-wise encryption

Lemma 4.5. *The chunk-wise encryption $\mathcal{E}^i_{2r-1,\widetilde{\Pi}}$ is not SPRP.*

Proof. Here we describe an SPRP distinguisher for an r-online mode having one chunk-wise encryption block $\mathcal{E}^i_{2r-1,\widetilde{\Pi}}$. The idea is similar to the one used by Nandi in [20].

Description of the SPRP-distinguisher $\mathcal{A}'_{\mathsf{sprp}}$.

(step 1) (Find a suitable difference in a pair of plaintext queries): Find $\delta \in \mathbb{B}^r$:

$$\delta = \mathsf{solve}(E^i_{12}[\ldots r-1\ ;\ *], 0^{r-1}).$$

(step 2) (Make the queries with the difference obtained in (step 1)): Now the distinguisher makes two queries $m^{i-1}\|0^r$ and $m^{i-1}\|\delta$ and obtains corresponding responses

$$c^{i-1}\|c = \mathcal{E}(m^{i-1}\|0) \quad \text{and} \quad c^{i-1}\|c' = \mathcal{E}(m^{i-1}\|\delta).$$

Let $u_1, v_1, \ldots, u_{2r-1}, v_{2r-1}$ and $u'_1, v'_1, \ldots, u'_{2r-1}, v'_{2r-1}$ respectively denote the intermediate inputs and outputs for the i-th block of the two queries. It is easy to see that $1 \le i \le r-1$, $u_i = u'_i$, $v_i = v'_i$ and

$$\Delta c := c + c' = E^i_{21}[r\ldots(2r-1)\ ;\ *] \cdot \Delta v_{r\ldots} + E^i_{22} \cdot \delta.$$

(step 3) (Solve Δu, Δv for the 2-queries): Using the invertible property of $E^i_{21}[r\ldots(2r-1)\ ;*]$, we can actually solve $\Delta v_{r\ldots}$ and hence $\Delta u_{r\ldots}$. Thus, we know Δu and Δv.

(step 4) (Find Δx, Δy for the 2-queries): Suppose we make two (redundant) decryption queries $c^{i-1}\|c$ and $c^{i-1}\|c'$ (whose responses must be $m^{i-1}\|0$ and $m^{i-1}\|\delta$) and let $x_1, y_1, \ldots, x_{2r-1}, y_{2r-1}$ and $x'_1, y'_1, \ldots, x'_{2r-1}, y'_{2r-1}$ respectively denote the intermediate inputs and outputs for the two queries. Then by the definition of the decryption algorithm we also know Δx and Δy, which are nothing but a (β, π)-reordering of $(\Delta u, \Delta v)$.

(step 5) (Find a difference for the final decryption query): Now we find a difference δ' such that

$$(D^i_{11}[\ldots r\ ;*]\ D^i_{12}[\ldots r\ ;\ *]) \cdot \begin{pmatrix} \delta' \\ \Delta y_1 \end{pmatrix} = \begin{pmatrix} \Delta x_1 \\ 0^{r-1} \end{pmatrix}.$$

We can solve the above equation for a non-zero δ', assuming that $\Delta x_1 \neq 0$ (see the remark below). Here we assume that the matrix $D^i_{11}[\ldots r\ ;*]$ is invertible.

(step 6) (Make the queries with the difference obtained in (step 5)): Now we make two decryption queries $c^{i-1}\|\bar{c}$ and $c^{i-1}\|\bar{c}'$, where $\bar{c}' = \bar{c} + \delta'$, with the first block-cipher input of the i-th chunk-wise encryption being x_1 and x'_1. While we set two queries, we should ensure that none of these have been obtained in the first two encryption queries (these are also called non-pointless or non-trivial queries). Let $x_1, y_1, \bar{x}_2, \bar{y}_2, \ldots, \bar{x}_{2r-1}, \bar{y}_{2r-1}$ and $x'_1, y'_1, \bar{x}'_2, \bar{y}'_2, \ldots, \bar{x}'_{2r-1}, \bar{y}'_{2r-1}$ respectively denote the intermediate inputs and outputs for these two queries, and let $\bar{m}$ and $\bar{m}'$ denote the corresponding responses. By the choice of δ', we know that $\Delta\bar{y}_{2\ldots r} = 0^{r-1}$.

(step 7) (Find a nullifier of unknown intermediate values, the same as the PRP distinguisher): Find a non-zero binary vector $\delta^* \in \mathbb{B}^r$:

$$\delta^* = \mathsf{solve}^*(D^i_{21}[*, r+1\ldots], 0^{r-1}).$$

(step 8) (The distinguisher event): If $\delta^* \cdot \Delta\bar{m} = \delta^* \cdot D^i_{22} \cdot \delta'$, then it returns 1 (decision for the keyed construction), else it returns 0 (decision for uniform random permutation). □

Remark. In the above attack, we assume that $\Delta x_1 \neq 0$ since otherwise we do not get a non-zero δ'. Note that Δx_1 can be written as a function of c and c', and for a random permutation the probability of the event $\Delta x_1 = 0$ is low.

4.5 Optimal number of block-functions for online PRP and SPRP

Theorem 4.6. *An r-block online encryption mode of ℓ chunks requires* (i) *at least $2r\ell - 1$ many block-functions with further restrictions that* (ii) *each chunk requires at least $2r - 1$ many block-functions and* (iii) *two chunks with $2r - 1$ many block-functions must have at least one chunk with more than $2r$ many block-functions in between.*

Proof. Condition (ii) follows directly from Proposition 4.4. Condition (iii) is obtained from the following arguments:

- Let i be the chunk with $(2r-1)$ block-functions. From Proposition 4.2 we can query two messages m_1 and m_2, of i-chunks each (keeping the first $(i-1)$ chunks identical) such that $\Delta w^{(i)}$ can be computed.
- Now using Lemma 4.3, we claim that $\mathcal{E}^{i+1}$ requires at least $2r$ many block-functions.
- As $\Delta w^{(i)}$ is known, if $\mathcal{E}^{i+1}, \ldots, \mathcal{E}^{i+j}$ use $2r$ many block-functions each, then, by applying Lemma 4.1 successively, one can compute $\Delta w^{(i+j)}$. Hence according to Lemma 4.3, $\mathcal{E}^{i+j}$ also requires at least $2r$ many block-functions. So, for any 2 chunks having $(2r-1)$ block-functions there must be one chunk having more than $2r$ many block-functions.

It is easy to see that conditions (ii) and (iii) imply condition (i), and hence the theorem follows. □

Theorem 4.7. *An r-block online encryption mode of ℓ chunks requires at least $2r\ell$ many block-functions with each chunk requiring at least $2r$ many block-functions.*

Proof. The proof follows from Lemma 4.5. □

The optimality is achieved with a simple variant of HCBC-1 (online PRP with $r = 1$) and TC3 (online SPRP with $r = 1$ and any ℓ).

5 Optimality for message authentication codes in affine modes

In this section, we first formally define affine mode message authentication codes and then show that ℓ many block-function calls are required to process a message of length ℓ. A simple variant of PMAC is an example with the tight bound.

5.1 Affine MAC modes

Given a parameter q and a family of q many permutations $\widetilde{\Pi} = (\pi_1, \ldots, \pi_q)$, an affine mode message authentication code $\mathcal{F}_{\ell,q,\widetilde{\Pi},\kappa'} := (\mathcal{F}_1, \mathcal{F}_2)$ takes a message m and returns a b-block tag t as follows:

$$\begin{aligned} u_i &= \mathcal{F}_1(v_1, \ldots, v_{i-1}, m, \kappa), \quad i = 1, \ldots, q, \\ v_i &= \pi_i(u_i), \qquad\qquad\qquad\quad i = 1, \ldots, q, \\ t &= \mathcal{F}_2(v_1, \ldots, v_q, m, \kappa). \end{aligned}$$

It is easy to see that the function $\mathcal{F}_{\ell,q,\widetilde{\Pi},\kappa'}$ can alternatively be represented by a $(q+b) \times (q+\ell+k)$ MAC co-efficient matrix F (with F_{11} being strictly lower triangular) such that

$$\begin{pmatrix} u \\ t \end{pmatrix} = \begin{pmatrix} (F_{11})_{q\times q} & (F_{12})_{q\times \ell} & (F_{13})_{q\times k} \\ (F_{21})_{b\times q} & (F_{22})_{b\times \ell} & (F_{23})_{b\times k} \end{pmatrix} \cdot \begin{pmatrix} v \\ m \\ \kappa \end{pmatrix},$$

where $\pi_i(u_i) = v_i$.

Examples. CBC-MAC [2], PMAC [4], TMAC [16], OMAC [12], DAG-based constructions [14], a sub-class of affine domain extension or ADE [19] etc. are some examples of linear MAC modes.

5.2 Optimal number of block-functions for affine MAC mode

Theorem 5.1. *A secure affine mode MAC requires at least ℓ many block-functions to respectively process a message M of ℓ-blocks.*

Proof. Here we show an integrity attack on the affine MAC mode $\mathcal{F}_{\ell,\ell-1,\widetilde{\Pi},K}$ (i.e. an affine MAC mode that uses $\ell - 1$ many block-functions). Assume $\ell > b$ and mount the attack as described below.

Description of the forger $\mathcal{A}_{\mathsf{forge}}$.
(step 1) (Find a suitable difference in a pair of plain-text queries): Find non-zero $\delta \in \mathbb{B}^{\ell}$:

$$\delta = \mathsf{solve}(F_{12}, 0^{\ell-1}).$$

(step 2) (Make an encryption query): Make a query m_1 of ℓ blocks. Let the corresponding tag be t_1.
(step 3) (Make the forging query with the difference obtained in (step 1)): Compute $m_2 = m_1 + \delta$ and $t_2 = t_1 + F_{22} \cdot \delta$. Forge with (m_2, t_2).
The theorem follows from the above attack. □

The optimality is achieved by the affine mode construction PMAC, assuming an independent masking key (not generated using block-function).

6 Optimality for authenticated encryption in affine modes

In this section, we consider affine mode authenticated encryption schemes based on four different types of settings, depending on whether we have to respect nonce (i.e. nonce-respecting) or we can repeat nonce (i.e. nonce-misuse) and whether we follow traditional AE settings or INT-RUP security settings. We recall that the rate of an AE scheme denotes the number of message blocks processed per block-function call. We call an AE scheme to be "rate-1" if the following holds: to process a message of ℓ blocks (i) exactly ℓ many block-function calls are used to generate the ciphertext and (ii) a constant number (denoted by z) of additional block-function calls are required to generate the tag. Instead of showing the exact number of block-functions required, we will find the result in terms of the rate of the mode; see Table 3.

6.1 Rate-1 affine mode authenticated encryption scheme

We represent a "rate-1" affine mode authenticated encryption as

$$E \cdot \begin{pmatrix} \kappa \\ m \\ y^* = \begin{pmatrix} y \\ y_{\mathrm{tag}} \end{pmatrix} \end{pmatrix} = \begin{pmatrix} x^* = \begin{pmatrix} x \\ x_{\mathrm{tag}} \end{pmatrix} \\ c^* = \begin{pmatrix} c \\ t \end{pmatrix} \end{pmatrix},$$

where $x = x^*[1 \dots \ell]$, $x_{\mathrm{tag}} = x^*[(\ell+1) \dots (\ell+z)]$, $y = y^*[1 \dots \ell]$ and $y_{\mathrm{tag}} = y^*[(\ell+1) \dots (\ell+z)]$. It is easy to see that a "rate-1" affine mode AE scheme has the following structure of E:

$$E = \begin{pmatrix} (E_{11})_{\ell\times k} & (E_{12})_{\ell\times\ell} & (E_{13})_{\ell\times\ell} & (E_{14})_{\ell\times z} \\ (E_{21})_{z\times k} & (E_{22})_{z\times\ell} & (E_{23})_{z\times\ell} & (E_{24})_{z\times z} \\ (E_{31})_{\ell\times k} & (E_{32})_{\ell\times\ell} & (E_{33})_{\ell\times\ell} & (E_{34})_{\ell\times z} \\ (E_{41})_{1\times k} & (E_{42})_{1\times\ell} & (E_{43})_{1\times\ell} & (E_{44})_{1\times z} \end{pmatrix}.$$

It is easy to check that E_{13} and E_{24} are strictly lower triangular matrices and $E_{14} = E_{34}$ are zero matrices.

For the decryption, we have identical representations as we replace E by D, x by u, y by v, m by c, and c by m:

$$D \cdot \begin{pmatrix} \kappa \\ c \\ v^* = \begin{pmatrix} v \\ v_{\mathrm{tag}} \end{pmatrix} \end{pmatrix} = \begin{pmatrix} u^* = \begin{pmatrix} u \\ u_{\mathrm{tag}} \end{pmatrix} \\ \begin{pmatrix} m \\ t \end{pmatrix} \end{pmatrix},$$

Nonce	Security	Optimal rate	Examples
Respecting	Traditional	1	OCB
Respecting	INT-RUP	< 1	OCB-IC
Misuse	Traditional	< 1	ELmD
Misuse	INT-RUP	< 1	OCB-IC

Table 3: Optimal rate of affine mode AE to achieve security in different settings.

where

$$u = u^*[1 \dots \ell], \quad u_{\text{tag}} = u^*[(\ell+1) \dots (\ell+z)],$$
$$v = v^*[1 \dots \ell], \quad v_{\text{tag}} = v^*[(\ell+1) \dots (\ell+z)].$$

It is easy to see that D has the following structure:

$$D = \begin{pmatrix} (D_{11})_{\ell\times k} & (D_{12})_{\ell\times\ell} & (D_{13})_{\ell\times\ell} & (D_{14})_{\ell\times z} \\ (D_{21})_{z\times k} & (D_{22})_{z\times\ell} & (D_{23})_{z\times\ell} & (D_{24})_{z\times z} \\ (D_{31})_{\ell\times k} & (D_{32})_{\ell\times\ell} & (D_{33})_{\ell\times\ell} & (D_{34})_{\ell\times z} \\ (D_{41})_{1\times k} & (D_{42})_{1\times\ell} & (D_{43})_{1\times\ell} & (D_{44})_{1\times z} \end{pmatrix}.$$

6.1.1 Important properties of the decryption matrix *D*

Lemma 6.1. *If* $\text{rank}(D_{33}) < (\ell - k)n$*, then the AE construction does not preserve privacy.*

Proof. We have the condition $D_{31} \cdot \kappa + D_{32} \cdot c + D_{33} \cdot v = m$. As the combined rank of $[D_{31} : D_{32} : D_{33}]$ is full (otherwise the scheme is not decryptable), we can find a row vector δ such that $\delta \cdot D_{32} \neq 0$ but $\delta \cdot D_{31} = 0$ and $\delta \cdot D_{33} = 0$. This gives a linear equation in c and m:

$$\delta \cdot D_{32} \cdot c = \delta \cdot m.$$

By using this equation, one can distinguish this scheme from a random function making a single query and checking whether the above equation holds or not. □

Lemma 6.2. *If* $\text{rank}(D_{12}) < (\ell - z)n$*, then the AE construction does not have integrity security.*

Proof. Let the decryption matrix for an AE Scheme be D, with $\text{rank}(D_{12}) < (\ell - z)n$. Now, we describe an integrity attacker (or forger) against the AE scheme as follows.

Description of the forger $\mathcal{A}_{\text{int}}$.

(step 1) (Finding a suitable difference in a pair of queries): Find a non-zero $\Delta c = (\Delta c_1, \dots, \Delta c_\ell)$ satisfying $D_{12} \cdot \Delta c = 0$ and $D_{22} \cdot \Delta c = 0$. The rank of D_{12} ensures that we will find such a Δc value for some ℓ.

(step 2) (Make an encryption query): Make an encryption query

$$(n^*, m = (m_1, m_2, \dots, m_\ell)).$$

Suppose $c = (c_1, c_2, \dots, c_\ell, t)$ to be the tagged ciphertext.

(step 3) (The forging event): Compute $\Delta t = D_{42} \cdot \Delta c$, and forge with $(n^*, c + \Delta c, t + \Delta t)$. □

6.1.2 Some examples

Here we provide some popular examples of authenticated encryption schemes which are "rate-1" block-cipher based, and for each of them we identify the underlying E matrix (considering complete block messages).

Example (iFeed [25]). iFeed is an example of a "rate-1" authenticated mode authenticated encryption construction with $z = 1$, key index $(\pi, (\kappa_0, \kappa_1, \kappa_2))$, $\rho_\mu = (\pi, \ldots, \pi)$ and encryption matrix E given as follows:

$$E_{11} = \begin{pmatrix} \mathbf{0} & \mathbf{2}^2 & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{2}^3 & \mathbf{1} & \mathbf{0} \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{0} & \mathbf{2}^{\ell+1} & \mathbf{1} & \mathbf{0} \end{pmatrix}, \quad E_{12} = \begin{pmatrix} (\mathbf{0}) & \mathbf{0} \\ \mathbf{I}_{\ell-1} & (\mathbf{0}) \end{pmatrix}, \quad E_{13} = E_{14} = (\mathbf{0}),$$

$$E_{21} = (\mathbf{0\ 2\ 1\ 0}), \quad E_{22} = (\mathbf{0\ 0}\ \ldots\ \mathbf{0\ 1}), \quad E_{23} = E_{24} = (\mathbf{0}),$$

$$E_{31} = \begin{pmatrix} \mathbf{0} & \mathbf{2}^3 & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{2}^4 & \mathbf{1} & \mathbf{0} \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{0} & \mathbf{2}^{\ell+2} & \mathbf{1} & \mathbf{0} \end{pmatrix}, \quad E_{32} = \mathbf{I}_\ell, \quad E_{33} = \mathbf{I}_\ell, \quad E_{34} = (\mathbf{0}),$$

$$E_{41} = (\mathbf{0\ 0\ 0\ 1}), \quad E_{42} = E_{43} = (\mathbf{0}), \quad E_{44} = \mathbf{1}.$$

Example (OCB [21]). OCB is another example of a "rate-1" authenticated encryption mode (which is not a feedback-based construction) with $z = 1$, key index (π, κ_0), $\rho_\mu = (\pi, \ldots, \pi)$ and encryption matrix E given as follows:

$$E_{11} = \begin{pmatrix} \mathbf{0} & \gamma_1 & \mathbf{1} \\ \mathbf{0} & \gamma_2 & \mathbf{1} \\ \vdots & \vdots & \vdots \\ \mathbf{0} & \gamma_{\ell-1} & \mathbf{1} \\ <\mathbf{n}> & (\gamma_\ell + \mathbf{2}^{-1}) & \mathbf{1} \end{pmatrix}, \quad E_{12} = \begin{pmatrix} \mathbf{I}_{\ell-1} & (\mathbf{0}) \\ (\mathbf{0}) & \mathbf{0} \end{pmatrix}, \quad E_{13} = E_{14} = (\mathbf{0}),$$

$$E_{21} = (\mathbf{0}\ \gamma_\ell\ \mathbf{1}), \quad E_{22} = (\mathbf{1}), \quad E_{23} = E_{24} = (\mathbf{0}),$$

$$E_{31} = \begin{pmatrix} \mathbf{0} & \gamma_1 & \mathbf{1} \\ \mathbf{0} & \gamma_2 & \mathbf{1} \\ \vdots & \vdots & \vdots \\ \mathbf{0} & \gamma_{\ell-1} & \mathbf{1} \\ \mathbf{0} & \gamma_\ell & \mathbf{1} \end{pmatrix}, \quad E_{32} = \begin{pmatrix} \mathbf{0}_{\ell-1} & (\mathbf{0}) \\ (\mathbf{0}) & \mathbf{1} \end{pmatrix}, \quad E_{33} = \mathbf{I}_\ell, \quad E_{34} = (\mathbf{0}),$$

$$E_{41} = E_{42} = E_{43} = (\mathbf{0}), \quad E_{44} = \mathbf{1}.$$

Here, $\mathbf{I}_\ell$ and $\mathbf{0}_\ell$ denote identity and all zero square matrices of dimension ℓ, respectively. We use $(\mathbf{0})$ to denote an all-zero sub-matrix of appropriate size. Furthermore, $\langle \mathbf{n} \rangle$ is used to denote the field element corresponding to the integer n (as we consider complete block messages).

6.2 INT-RUP attack on "rate-1" affine mode authenticated encryption

In this section, we prove the following theorem.

Theorem 6.3. *Any "rate-1" block-cipher-based affine mode authenticated encryption scheme is INT-RUP insecure.*

Proof. Here we describe an INT-RUP attacker on generic "rate-1" affine domain authenticated encryption schemes. The attack consists of one encryption and one unverified plaintext query. For any two vectors A^a and A^b, we use the notation ΔA^{ab} to denote the vector $(A^a + A^b)$.

Description of the INT-RUP attacker $\mathcal{A}_{\text{int\_rup}}$.

(step 1) (Make an encryption query): The adversary first makes an encryption query

$$(n^*, m^0 = (m_1^0, m_2^0, \ldots, m_\ell^0)).$$

Let $c^0 = (c_1^0, c_2^0, \ldots, c_\ell^0, t^0)$ be the tagged ciphertext.

(step 2) (Make an unverified plaintext query): Next, the adversary makes an unverified plaintext query $(n^*, c^1 = (c_1^1, c_2^1, \ldots, c_\ell^1))$. Let $m^1 = (m_1^1, m_2^1, \ldots, m_\ell^1)$ be the corresponding plaintext.

(step 3) (Find the v-difference of the two queries): Find Δv^{01} from

$$\Delta v^{01} = D_{33}^{-1} \cdot (\Delta m^{01} + D_{32} \cdot \Delta c^{01}).$$

(step 4) (Represent Δc^{0f} in terms of a non-zero vector δ): Given any non-zero binary vector $\delta = (\delta_1, \ldots, \delta_\ell)$, represent Δc^{0f} in terms of δ as follows:

$$\Delta c^{0f} = D_{12}^{-1} \cdot D^* \cdot \delta,$$

where

$$D^* = \begin{pmatrix} \Delta u_1^{01} + D_{13}^{11} \cdot \Delta v_1^{01} & D_{13}^{12} \cdot \Delta v_2^{01} & \cdots & D_{13}^{1\ell} \cdot \Delta v_\ell^{01} \\ D_{13}^{21} \cdot \Delta v_1^{01} & \Delta u_2^{01} + D_{13}^{22} \cdot \Delta v_2^{01} & \cdots & D_{13}^{2\ell} \cdot \Delta v_\ell^{01} \\ \vdots & \vdots & \ddots & \vdots \\ D_{13}^{\ell 1} \cdot \Delta v_1^{01} & D_{13}^{\ell 2} \cdot \Delta v_2^{01} & \cdots & \Delta u_\ell^{01} + D_{13}^{\ell\ell} \cdot \Delta v_\ell^{01} \end{pmatrix}.$$

(step 5) (Find a suitable δ that makes $\Delta u_{\text{tag}}^{0f} = 0$): Solve the following set of equations to find a δ that implies $\Delta u_{\text{tag}}^{0f} = 0$:

$$D_{22} \cdot \Delta c^{0f} + D_{23} \cdot \Delta v^{0f} = 0.$$

As Δc^{0f} and Δv^{0f} can be represented as linear combinations of δ, the above equality can be written as $(D_{22} \cdot D_{12}^{-1} \cdot D^* + D_{23} \cdot v^*) \cdot \delta = 0$. It is easy to see that this equation has at least one solution as long as $\ell > (z-1)n$. Let the solution be δ^*.

(step 6) (Compute the ciphertext and tag difference): Find Δc^{0f} and Δt^{0f} as we put $\delta = \delta^*$ in the following equations:

$$\Delta c^{0f} = D_{12}^{-1} \cdot D^* \cdot \delta,$$
$$\Delta t^{0f} = D_{42} \cdot \Delta c^{0f} + D_{43} \cdot \Delta v^* \cdot \delta.$$

(step 7) (Forging event): Forge with $(n^*, c^f := c^0 + c^{0f}, t^f := t^0 + t^{0f})$.

Brief explanation. The adversarial goal is to construct a forging query (with the same nonce and associated data) $(n^*, c^f = (c_1^f, c_2^f, \ldots, c_\ell^f), t^f)$, which realizes a $\delta = (\delta_1, \ldots, \delta_\ell)$ sequence. The ciphertext C^f realizes a δ-sequence if, given a non-zero binary vector $\delta = (\delta_1, \ldots, \delta_\ell)$, $u_i^f = u_i^{\delta_i}$ for all $i \le \ell$, and $u_i^f = u_i^0$ for all $i > \ell$. Note that, by definition, $\Delta u_i^{0f} = \delta_i \cdot \Delta u_i^{01}$ and $\Delta v_i^{0f} = \delta_i \cdot \Delta v_i^{01}$. So, one can write both Δu^{0f} and v^{0f} as a linear combination of δ, and hence represent Δc^{0f} in terms of δ:

$$\Delta c^{0f} = D_{12}^{-1} \cdot (\Delta u^{0f} + D_{32} \cdot \Delta v^{0f}) = D_{12}^{-1} \cdot D^* \cdot \delta.$$

Now, the adversary chooses a δ (we term it as δ^*) such that the second condition (i.e. $u_i^f = u_i^0$ for all $i > \ell$) gets satisfied. Finally, the adversary computes the ciphertext-difference Δc^{0f} and the tag-difference Δt^{0f} corresponding to δ^* and mounts the attack.

Case when at least one of D_{12} and D_{33} does not have full rank. From Lemma 6.1 and 6.2 we already know that $\text{rank}(D_{12})$ and $\text{rank}(D_{33})$ should be high. This ensures that if we set ℓ appropriately to a high value, we will have an $(n \times n)$ sub-matrix which has full rank for both D_{12} and D_{33}. More formally, from Lemmas 6.1 and 6.2 we know that $\text{rank}(D_{12}) > (\ell - (k+1))n$ and $\text{rank}(D_{33}) > (\ell - z)n$. It is easy to check that we can find a value of ℓ such that both the sub-matrices $D_{12}[\ell - n \ldots, \ell - n \ldots]$ and $D_{33}[\ell - n \ldots, \ell - n \ldots]$ have full rank. As k and z are small constants, one can ensure that we will find such an ℓ. Now one can easily modify the previous attack and apply it here. □

Remarks (Extension of the attack for any number of keys). In the definition of affine domain authenticated encryption, we have assumed k, the number of keys, to be constant. Some constructions like IACBC [15] and IAPM [13] use $\log \ell$ many keys while encrypting ℓ block messages. It is easy to see that our INT-RUP attack will be valid for these constructions as well. In general, this attack will be applicable for any "rate-1"

authenticated encryption scheme for which D_{11} and D_{22} are invertible, even if the number of masking keys it uses depends on the message length.

6.3 Integrity attack on "rate-1" affine mode authenticated encryption in nonce-misuse scenario

Theorem 6.4. *Any "rate-1" block-cipher-based AE scheme is not integrity secure against nonce-repeating adversaries.*

In this section, we describe a generic INT-CTXT attack under a nonce-misuse scenario on "rate-1" affine AE schemes. The technique is similar to that for the previous attack, except we make two encryption queries with the same nonce instead of one encryption and one INT-RUP query.

Description of the INT-RUP attacker $\mathcal{A}'_{\text{int\_rup}}$.

(step 1) (Make first encryption query): Make an encryption query $(n^*, m^0 = (m_1^0, m_2^0, \ldots, m_\ell^0))$. Let

$$c^0 = (c_1^0, c_2^0, \ldots, c_\ell^0, t^0)$$

be the tagged ciphertext.

(step 2) (Make second encryption query): Make another encryption query: $(n^*, m^1 = (m_1^1, m_2^1, \ldots, m_\ell^1))$. Let $c^1 = (c_1^1, c_2^1, \ldots, c_\ell^1, t^1)$ be the tagged ciphertext.

(step 3) (Make the forging): Forge with $(c^f = (c_1^f, c_2^f, \ldots, c_\ell^f), t^f)$, where $c_i^f = c_i^0 + \Delta c_i^{0f}$ and $t^f = t^0 + \Delta t^{0f}$ with c^{0f} and t^{0f} being defined as in the previous attack.

Clearly, the main idea of the previously described attack is to find one valid plaintext-ciphertext-tag pair $(n^*, m^0, (c^0, t^0))$ and one plaintext-ciphertext pair (n^*, m^1, c^1) for any choice of n^*, m^0 and c^1. By two nonce-misusing queries we can get $(n^*, m^0, (c^0, t^0))$ and $(n^*, m^1, (c^1, t^1))$ for any choice of n^*, m^0 and m^1. Thus we can follow the same procedure described in the last section and forge a valid ciphertext-tag pair.

6.4 On the optimality of "rate" for affine mode authenticated encryption

Theorem 6.5. *In a traditional nonce-respecting scenario, the optimum rate of a secure affine mode authenticated encryption is* 1 *(with* $z = 1$*). But if we consider a nonce-misuse setting or an INT-RUP model, then any block-cipher-based affine mode authenticated encryption must have "rate* < 1*".*

Proof. From Theorem 3.2 we know that a block-cipher-based affine mode authenticated encryption (that takes ℓ blocks and outputs $\ell + 1$ many blocks) needs at least $\ell + 1$ many block-functions to achieve PRF security, assuming distinct nonce criteria. Now, the existing construction OCB (that uses $\ell + 1$ many block-functions) achieves that, and hence provides the optimal bound.

The second part of the theorem follows directly from Theorems 6.3 and 6.4. □

7 Conclusion

In this paper, we have considered symmetric key affine modes of operation and determined the optimal number of non-linear computations required to achieve (i) PRF security of encryption modes (with and without nonce), (ii) SPRP security of length-preserving encryption modes, (iii) online PRF and online SPRP security of r-online encryption modes and (iv) message authentication codes. The tightness of these bounds has been shown through some known constructions achieving these bounds. Moreover, we have considered authenticated encryption modes and determined the maximum rate of the construction to achieve security in different settings. These are important theoretical results, which could be used as a guideline while designing new symmetric key algorithms, minimizing the number of block-function invocations.

References

[1] M. Bellare, A. Boldyreva, L. Knudsen and C. Namprempre, On-line ciphers and the hash-cbc constructions, in: *Advances in Cryptology—CRYPTO 2001*, Lecture Notes in Comput. Sci. 2139, Springer, Berlin (2001), 292–309.

[2] M. Bellare, J. Kilian and P. Rogaway, The security of cipher block chaining, in: *Advances in Cryptology—CRYPTO 1994*, Lecture Notes in Comput. Sci. 839, Springer, Berlin (1994), 341–358.

[3] T. P. Berger, M. Minier and G. Thomas, Extended generalized feistel networks using matrix representation, in: *Selected Areas in Cryptography—SAC 2013*, Springer, Berlin (2014), 289–305.

[4] J. Black and P. Rogaway, A block-cipher mode of operations for parallelizable message authentication, in: *Advances in Cryptology—Eurocrypt 2002*, Lecture Notes in Comput. Sci. 2332, Springer, Berlin (2002), 384–397.

[5] A. Chakraborti, N. Datta and M. Nandi, INT-RUP analysis of block-cipher based authenticated encryption schemes, in: *Topics in Cryptology—CT-RSA 2016*, Lecture Notes in Comput. Sci. 9610, Springer, Berlin (2016), 39–54.

[6] D. Chakraborty and M. Nandi, An improved security bound for HCTR, in: *Fast Software Encryption—FSE 2008*, Lecture Notes in Comput. Sci. 5086, Springer, Berlin (2008), 289–302.

[7] S. Halevi, EME*: Extending EME to handle arbitrary-length messages with associated data, in: *Progress in Cryptology—INDOCRYPT 2004*, Lecture Notes in Comput. Sci. 3348, Springer, Berlin (2004), 315–327.

[8] S. Halevi, Invertible universal hashing and the TET encryption mode, in: *Advances in Cryptology—CRYPTO 2007*, Lecture Notes in Comput. Sci. 4622, Springer, Berlin (2007), 412–429.

[9] S. Halevi and P. Rogaway, A tweakable enciphering mode, in: *Advances in Cryptology—CRYPTO 2003*, Lecture Notes in Comput. Sci. 2729, Springer, Berlin (2003), 482–499.

[10] S. Halevi and P. Rogaway, A parallelizable enciphering mode, in: *Topics in Cryptology—CT-RSA 2004*, Lecture Notes in Comput. Sci. 2964, Springer, Berlin (2004), 292–304.

[11] V. T. Hoang and P. Rogaway, On generalized feistel networks, in: *Advances in Cryptology—CRYPTO 2010*, Springer, Berlin (2010), 613–630.

[12] T. Iwata and K. Kurosawa, Omac: One-key cbc mac, in: *Fast Software Encryption—FSE 2003*, Lecture Notes in Comput. Sci. 2887, Springer, Berlin (2003), 129–153.

[13] C. S. Jutla, Encryption modes with almost free message integrity, in: *Advances in Cryptology—EUROCRYPT 2001*, Lecture Notes in Comput. Sci. 2045, Springer, Berlin (2003), 529–544.

[14] C. S. Jutla, Prf domain extension using dag, in: *Theory of Cryptography—TCC 2006*, Lecture Notes in Comput. Sci. 3876, Springer, Berlin (2006), 561–580.

[15] C. S. Jutla, Encryption modes with almost free message integrity, *J. Cryptology* **21** (2008), no. 4, 547–578.

[16] K. Kurosawa and T. Iwata, Tmac: Two-key cbc mac, in: *Topics in Cryptology—CT-RSA 2003*, Lecture Notes in Comput. Sci. 2612, Springer, Berlin (2003), 33–49.

[17] M. Luby and C. Rackoff, How to construct pseudo-random permutations from pseudo-random functions, in: *Advances in Cryptology—CRYPTO 1985*, Lecture Notes in Comput. Sci. 218, Springer, New York (1984), 447.

[18] M. Nandi, Two new efficient cca-secure online ciphers: Mhcbc and mcbc, in: *Progress in Cryptology—INDOCRYPT 2008*, Lecture Notes in Comput. Sci. 5365, Springer, Berlin (2008), 350–362.

[19] M. Nandi, A unified method for improving PRF bounds for a class of blockcipher based macs, in: *Fast Software Encryption—FSE 2010*, Lecture Notes in Comput. Sci. 6147, Springer, Berlin (2010), 212–229.

[20] M. Nandi, On the optimality of non-linear computations of length-preserving encryption schemes, in: *Advances in Cryptology—ASIACRYPT 2015 Part II*, Springer, Berlin (2015), 113–133.

[21] P. Rogaway, Efficient instantiations of tweakable blockciphers and refinements to modes ocb and pmac, in: *Advances in Cryptology—ASIACRYPT 2004*, Lecture Notes in Comput. Sci. 3329, Springer, Berlin (2004), 16–31.

[22] P. Rogaway and H. Zhang, Online ciphers from tweakable blockciphers, in: *Topics in Cryptology—CT-RSA 2011*, Lecture Notes in Comput. Sci. 6558, Springer, Berlin (2011), 237–249.

[23] P. Sarkar, Improving upon the tet mode of operation, in: *Information Security and Cryptology—ICISC 2007*, Lecture Notes in Computer Science 4817, Springer, Berlin (2007), 180–192.

[24] P. Wang, D. Feng and W. Wu, HCTR: A variable-input-length enciphering mode, in: *Information Security and Cryptology—CISC 2005*, Lecture Notes in Comput. Sci. 3822, Springer, Berlin (2005), 175–188.

[25] L. Zhang, W. Wu, H. Sui and P. Wang, iFeed[AES] v1, 2014, https://competitions.cr.yp.to/round1/ifeedaesv1.pdf.