

Practical Forgery Attacks on Limdolen and HERN

Raghvendra Rohit and Guang Gong

Department of Electrical and Computer Engineering, University of Waterloo,
Waterloo, Ontario, N2L 3G1, CANADA.
{rsrohit, ggong}@uwaterloo.ca

Abstract. In this paper, we investigate the security of Limdolen and HERN which are Round 1 submissions of the ongoing NIST Lightweight Cryptography Standardization Project. We show that some non-conservative design choices made by the designers solely to achieve a lightweight design lead to practical forgery attacks. In particular, we create **associated data-only**, **ciphertext-only** and **associated data and ciphertext** forgeries which require a feasible number of forging attempts.

Limdolen employs a tweaked PMAC based construction to offer authenticated encryption functionality. It has two variants, Limdolen-128 and Limdolen-256 with key sizes 128 and 256 bits, respectively. The designers claim 128(256)-bit integrity security for Limdolen-128(256). Our main observation is that it uses a sequence of period 2 consisting of only two distinct secret masks. This structural flaw attributes to a successful forgery (all three types) with probability 1 after observing the output of a single encryption query. While, HERN is a 128-bit authenticated encryption scheme whose high level design is inspired from the CAESAR finalist Acorn. We show a message modification strategy by appending/removing a sequence of consecutive ‘0’ bits. Accordingly, we can construct **associated data-only**, **ciphertext-only** and **associated data and ciphertext** forgery with the success rate of 2^{-1} , 2^{-1} and 1 after 2, 4 and 2 encryption queries, respectively.

Overall, our attacks defeat the claim of 128(256) and 128-bit integrity security of Limdolen-128(256) and HERN, respectively. We have experimentally verified the correctness of our attacks with the reference implementations. Notably, these are the first cryptanalytic results on both algorithms. Consequently, our results are expected to help in further understanding of similar designs.

Keywords. NIST lightweight cryptography standardization project · AEAD · Limdolen · HERN · Forgery

1 Introduction

The Internet of Things (IoT), sensor networks, distributed control systems and cyber physical systems are the most pre-eminent buzzwords these days. They have applications ranging from smart locks to wearable technology to home automation and healthcare. Typically, they operate in constrained environments and require reasonable efficiency with low implementation cost and sufficient security. The current standardized cryptographic primitives are designed for desktop and server environments, and many of them do not fit into the resource requirements of constrained devices. As a result, National Institute of Standards and Technology (NIST) initiated a lightweight cryptography project in 2013 and published the call for submissions of lightweight Authenticated Encryption with Associated Data (AEAD) algorithms and hash functions, in August 2018 [NIS19]. In total, NIST received 57 submissions and 56 out of them were announced as the Round 1 candidates in April 2019. Two of such submissions are Limdolen [Meh19] and HERN [YSMW19].

Limdolen is a family of lightweight AEAD algorithms with key sizes 128 and 256 bits. At a high level, it adopts a Parallelizable Message Authentication Code (PMAC) [BR02] mode to compute tag and then use counter mode of encryption to generate the ciphertext. The XOR value of tag and nonce serve as the initial counter. However, compared to PMAC where random and indistinguishable secret masks¹ are used, Limdolen-128/(256) utilizes two distinct 128(256)-bit secret masks only. The designers state that

“Due to Limdolen’s target of constrained environments, rather than a series of calculations, we will alternate between $i = 0$ and $i = 1$, the two most common values of i in $\gamma^i L$.”

¹ masks derived from PMAC key where PMAC key equals $\mathcal{E}_K(0^n)$

Moreover, during the tag computation phase, *the associated data and message are first combined together to form a single input and then the padding procedure is executed*. Based on the design choices and security proofs of PMAC and counter mode of encryption², the designers claim 128(256)-bit integrity security for Limdolen-128(256).

On the other hand, HERN is a 128-bit authenticated encryption scheme and adopts a stream cipher style construction similar to the CAESAR finalist Acorn [cae, Wu16]. The state size is 256 bits and at each clock cycle, 4 nonlinear bits are feedback to the state (except during ciphertext and tag generation phase). *After processing the associated data, the state is updated 512 times by adding ‘0’ bit stream to the feedback bits. A similar procedure is applied after plaintext processing*. Accordingly, they claim that HERN achieves 128-bit integrity security.

Analyzing the security of NIST LWC Round 1 submissions with respect to forgery attacks is crucial before they are standardized and put in practice. A strong motivation is the recent forgery and plaintext recovery attacks on OCB2 [IIMP19]. Its worth noting that OCB2 was included in ISO/IEC 19772:2009 [ISO] and forgeries are found a decade later. On a same note, practical forgeries are found for Round 1 submission SNEIKEN v1 [Saa19] by exploiting 1 round iterative differential [Per19, Kha19].

In this work, we investigate the security of Limdolen and HERN with reference to associated data-only, ciphertext-only and associated data and ciphertext forgeries in the nonce-respecting scenario. Table 1 presents a summary of our forgery attacks.

Table 1: Summary of forgery attacks on Limdolen and HERN. ‘–’ denotes that input could be either empty or non-empty.

Algorithm	Forgery type	# Enc. queries	# Dec. queries	Success prob.	# blocks
Limdolen-128	associated data-only	1	1	1	≥ 1
	ciphertext-only	1	1	1	≥ 4
	associated data and ciphertext	1	1	1	≥ 1
Limdolen-256	associated data-only	1	1	1	≥ 1
	ciphertext-only	1	1	1	≥ 4
	associated data and ciphertext	1	1	1	≥ 1
HERN	associated data-only	$2^n (1 \leq n \leq 63)$	2^n	1	–
	ciphertext-only	$2^{2n} (1 \leq n \leq 31)$	2^n	1	–
	associated data and ciphertext	$2^n (1 \leq n \leq 63)$	1	1	–

Our contributions. We present the practical forgery attacks on Limdolen and HERN in the nonce-respecting setting. Our attacks exploit the structural flaws in the underlying design of these algorithms. Thus, our contributions are summarized as follows.

- We exploit the period 2 secret masks of Limdolen-128/256 and show that the XOR sum value before the last block cipher call is always a constant even if we add/remove/permutate blocks arbitrary number of times.
- For both variants of Limdolen, we show the general construction of associated data-only, ciphertext-only and associated data and ciphertext forgeries which have a successful probability of 1 after observing the output of a single encryption query. While, after one query, the designers claim the success probability of 2^{-128} and 2^{-256} for Limdolen-128 and Limdolen-256, respectively.
- By modifying input data with a sequence of consecutive zero bits, we show that HERN can not distinguish between associated data and plaintext processing phases. This observation is independent of number of rounds.
- For HERN, we create associated data-only, ciphertext-only and associated data and ciphertext forgery with the success rate of 2^{-1} , 2^{-1} and 1 after 2, 4 and 2 encryption queries, respectively. For the same number of queries, designers claim the success rate of 2^{-127} , 2^{-126} and 2^{-127} , respectively. We present a generalized version of our attack, i.e., for $1 \leq n \leq 63$ ($1 \leq n' \leq 31$) the success rate of forgeries are 2^{-n} , $2^{-n'}$ and 1 after 2^n , $2^{2n'}$ and 2^n encryption queries, respectively.

² they consider it as SIV mode [RS07] in the reference document

- To validate our findings, we have experimentally verified the correctness with the reference implementations. We have also provided examples for each type of forgery.

Organization of the paper. A brief description of Limdolen is provided in Section 2. In Section 3, we present the details of forgery attacks on Limdolen along with the experimental results. Section 4 and 5 present the specifications and forgery attacks on HERN, respectively. Finally, the paper is concluded in Section 6. The notations used throughout the paper are defined in Table 2.

Table 2: Notations

Notation	Description
$X \odot Y, X \oplus Y, X Y, X Y$	bitwise AND, XOR, OR and concatenation of X and Y
$ X $	length of X in bits
$\{0, 1\}^{\geq n}$	bitstring with length at least n
$X \xleftarrow{\$} \{0, 1\}^n$	random n bitstring drawn from $\{0, 1\}^n$
$1^n, 0^n$	length n bitstring with all 1's, 0's
X^n	n repetitions of bitstring X
$\ll i, \lll i$	Left shift (left cyclic shift) by i bits
$(X_0, \dots, X_{l-1}) \stackrel{n}{\leftarrow} X$	n -bit block parsing of X where $ X_i = n$ for $0 \leq i \leq l-2$ and $1 \leq X_{l-1} \leq n$
$x_0, \dots, x_{ X -1}$	bit representation of X
$X[i]$	i -th byte of X starting from left
$K, N, T (k_i, n_i, t_i)$	key, nonce and tag (in bits)
$AD, M, C (ad_i, m_i, c_i)$	associated data, plaintext and ciphertext (in bits)

2 Specifications of Limdolen

Limdolen is a family of lightweight AEAD algorithms with key sizes 128 and 256 bits. We denote an instance of Limdolen by Limdolen- n and its corresponding underlying block cipher by Limdolen-BC- n where $n \in \{128, 256\}$. In this section, we first give a brief overview of Limdolen- n and then list the security goals claimed by the designers.

2.1 Description of Limdolen AEAD

Limdolen adopts a tweaked PMAC [BR02] based construction to provide AEAD functionality. It has two variants Limdolen- n , $n \in \{128, 256\}$. For both the variants, the size of key, nonce and tag are equal to n bits. We now describe the individual phases of Limdolen.

2.1.1 Padding The associated data AD and the message M are first concatenated together to form a single input message. It is then divided into chunks of n -bit blocks, i.e., $(X_0, \dots, X_{l-1}) \stackrel{n}{\leftarrow} AD||M$. If $|X_{l-1}| = n$, then a single byte is XORed to the last byte of X_{l-1} . This `pad_byte` equals `0xC0` (`0x80`) depending on whether the length of associated data is zero (non-zero). In case the number of bytes of X_{l-1} is less than $n/8$, first a `pad_byte` is appended to X_{l-1} , followed by adding zero bytes until the block length becomes n . This procedure is denoted by `addPaddingMarker()`.

Remark 1. The padding rule described above follows the Limdolen’s specification document (cf. Page 9 [Meh19]). However, in the reference implementation the `pad_byte` is always XORed to the last byte of X_{l-1} . Here, we emphasize that our attacks are independent of location of this byte.

2.1.2 Tag generation The tag computation of Limdolen- n is almost similar to PMAC [BR02] and is shown in Figure 1. First the PMAC key is derived by encrypting nonce with the master key. We denote it by `aeadK` where `aeadK = Limdolen-BC- n (K, N)`. Next, three n -bit masks given by

$$\alpha = \text{Limdolen-BC-}n(\text{aeadK}, 0^n), \text{ alpha_x} = \text{LB}(\alpha), \text{ and alpha_inv_x} = \text{RB}(\alpha)$$

are computed where the function $\text{LB}(\alpha)$ (resp. $\text{RB}(\alpha)$) rotates each byte of α left (resp. right) by 1. Each n -bit block X_i (except the last block) is XORed alternately with α or alpha_x which is then encrypted with $\text{Limdolen-BC-}n$ using aeadK as the key. At each iteration, the output is XORed to δ_c which acts as a checksum. The tag is then given by $T = \text{Limdolen-BC-}n(\text{aeadK}, \delta_c \oplus \text{alpha_inv_x} \oplus \text{addPaddingMarker}(X_{l-1}))$.

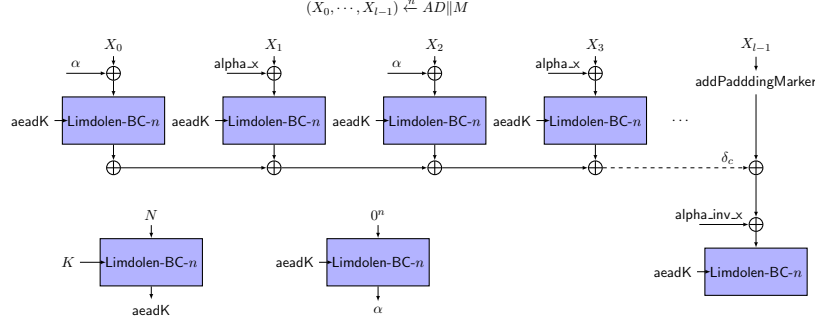


Fig. 1: Tag generation phase of $\text{Limdolen-}n$

2.1.3 Encryption The encryption is similar to the counter-mode of operation. The XOR value of nonce and tag is used as the initial counter. This phase is shown in Figure 2. The decryption is similar to encryption and hence the details are omitted.

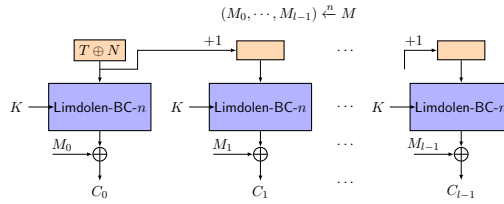


Fig. 2: Encryption phase of $\text{Limdolen-}n$

2.2 Security Claims

The security claims of Limdolen in the nonce-respecting setting are summarized in Table 3.

Table 3: Security claims of Limdolen in bits [Meh19]

Goal	Limdolen-128	Limdolen-256
Confidentiality of plaintext	128	256
Integrity of plaintext and associated data	128	256
Data limit (in blocks)	2^{64}	2^{128}

3 Forgery Attacks on Limdolen

In this section, we present the details of forgery attacks on both variants of Limdolen . First, we give a brief overview of the adversarial model and the main idea of our attack. Next, we show the construction of associated data-only, ciphertext-only and associated data and ciphertext forgeries that require a single encryption query and one forging attempt for successful verification. Finally, we provide the experimental results.

3.1 Adversarial Model

We assume that the adversary \mathcal{A} is nonce-respecting, which means it never makes two queries to the encryption oracle with the same nonce. Nevertheless, \mathcal{A} is allowed to repeat nonces in decryption queries. We say that “ \mathcal{A} forges” if decryption oracle ever returns a plaintext other than error symbol \perp on input of (N, AD, C, T) where (C, T) has never been output by encryption oracle on input of a query (N, AD, M) for some AD and M [Rog02]. In the sequel, we classify three types of forgeries based on the input modification.

- associated data-only: “ \mathcal{A} forges” by changing AD only
- ciphertext-only: “ \mathcal{A} forges” by changing C only
- associated data and ciphertext: “ \mathcal{A} forges” by changing both AD and C .

3.2 Core Idea of Forgery

For simplicity, we explain the idea for a single complete block of associated data which is given in Lemma 1.

Lemma 1. *Let $K \xleftarrow{\$} \{0, 1\}^n$ be fixed. Let $N \xleftarrow{\$} \{0, 1\}^n$, $AD_0 \xleftarrow{\$} \{0, 1\}^n$, $M = \epsilon$ and (ϵ, T) be the corresponding ciphertext and tag pair. Then for a positive integer $i \geq 1$ and $AD'_0 \xleftarrow{\$} \{0, 1\}^n$, $AD'_1 \xleftarrow{\$} \{0, 1\}^n$ and $AD' = (AD'_0 \| AD'_1 \| AD'_0 \| AD'_1)^i \| AD_0$, we have $C' = \epsilon$ and $T' = T$.*

Proof. Since $M' = M = \epsilon \implies C' = C = \epsilon$. We now look at the tag generation of AD and AD' . The respective tags are given by

$$T = \text{Limdolen-BC-}n(\text{aeadK}, \text{alpha_inv_x} \oplus \text{addPaddingMarker}(AD_0))$$

$$T' = \text{Limdolen-BC-}n(\text{aeadK}, \delta'_c \oplus \text{alpha_inv_x} \oplus \text{addPaddingMarker}(AD_0)),$$

where $\delta'_c = 0^n$ (see Figure 3 for $i = 1$ case). Thus $T' = T$. □

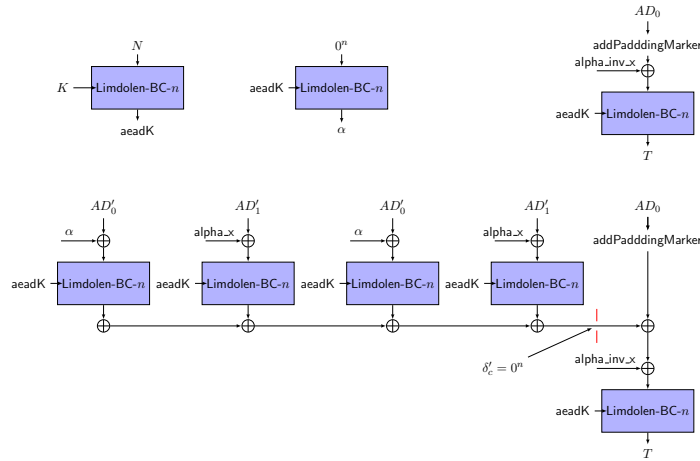


Fig. 3: Limdolen forgery for a single AD block

Corollary 1. *To construct forgery for arbitrary number of blocks, we only need to ensure that the XOR sum δ_c (see Figure 1) before the last call of block cipher is a constant.*

Remark 2. Lemma 1 trivially holds for partial last block.

3.3 Basic Forgery

We describe the basic minimal example of the forgery attack against Limdolen- n . We assume that blocks are complete and the number of blocks is at least 1. From now onwards, we refer Limdolen-BC- n with key K by $\mathcal{E}_K^n(\cdot)$.

3.3.1 Associated data-only forgery Let $u \geq 1$ and $i \geq 1$ be two positive integers. Fix $K \xleftarrow{\$} \{0, 1\}^n$. We construct forgery as follows.

Step 1 Let $N \xleftarrow{\$} \{0, 1\}^n$, $AD \leftarrow \{0, 1\}^{u \times n}$, $(AD_0, \dots, AD_{u-1}) \xleftarrow{n} AD$ and $M = \epsilon$. Encrypt (N, AD, M) and observe (C, T) .

Step 2 Let $X, Y \xleftarrow{\$} \{0, 1\}^n$ and $W = X \| Y \| X \| Y$.

Step 3 Forge with (N, AD', C, T) where $AD' = AD_0 \| \dots \| AD_{u-2} \| W^i \| AD_{u-1}$.

Note that $AD' \neq AD \implies$ the decryption query is valid. This will pass the verification with probability 1 and returns empty plaintext as the output. To see why this forgery works, consider the values of δ_c and δ'_c , which are given by

$$\delta_c = \bigoplus_{i \bmod 2=0}^{i < u-1} \mathcal{E}_{\text{aeadK}}^n(AD_i \oplus \alpha) \bigoplus_{i \bmod 2=1}^{i < u-1} \mathcal{E}_{\text{aeadK}}^n(AD_i \oplus \text{alpha_x})$$

If $u - 1$ is even then

$$\begin{aligned} \delta'_c &= \bigoplus_{i \bmod 2=0}^{i < u-1} \mathcal{E}_{\text{aeadK}}^n(AD_i \oplus \alpha) \bigoplus_{i \bmod 2=1}^{i < u-1} \mathcal{E}_{\text{aeadK}}^n(AD_i \oplus \text{alpha_x}) \\ &\quad 2i \bigoplus (\mathcal{E}_{\text{aeadK}}^n(X \oplus \alpha) \oplus \mathcal{E}_{\text{aeadK}}^n(Y \oplus \text{alpha_x})) \\ &= \delta_c \oplus 0^n \implies T' = T. \end{aligned}$$

Similarly, if u is odd then $\delta'_c = \delta_c \oplus 0^n$ and $T' = T$. The only difference is that masks α and alpha_x are interchanged.

Some observations on associated data-only forgery.

1. The converse also holds true, i.e., given $AD = AD_0 \| \dots \| AD_{u-2} \| W^i \| AD_{u-1}$, the modified associated data of the form $AD_0 \| \dots \| AD_{u-2} \| W^l \| AD_{u-1}$ will give the same tag for all l satisfying $1 \leq l < i$.
2. The forgery is independent of whether the last block is a partial AD/M block or consists of both AD and M bytes.
3. We can modify AD in a number of ways. For instance, the following modification also results in a successful forgery.

$$AD' = \begin{cases} X \| Y \| AD_0 \| \dots \| AD_{u-2} \| X \| Y \| AD_{u-1} & \text{if } u \text{ is odd,} \\ Y \| X \| AD_0 \| \dots \| AD_{u-2} \| X \| Y \| AD_{u-1} & \text{o.w.} \end{cases}$$

3.3.2 Ciphertext-only forgery Fix a integer $u \geq 4$ and $K \xleftarrow{\$} \{0, 1\}^n$. Let $S_e = \{0, 2, \dots\}$ and $S_o = \{1, 3, \dots\}$ be the set of even and odd integers less than $u - 1$. Consider two permutations π and ψ which permutes the set S_e and S_o , respectively. Assume that π and ψ are not identity permutations simultaneously. We now construct forgery as follows.

Step 1 Let $N \xleftarrow{\$} \{0, 1\}^n$, $AD = \epsilon$, $M \xleftarrow{n} \{0, 1\}^{u \times n}$ and $(M_0, \dots, M_{u-1}) \xleftarrow{n} M$. Encrypt (N, AD, M) and observe (C, T) .

Step 2 Let $(C_0, \dots, C_{u-2}, C_{u-1}) \xleftarrow{n} C$ and compute $Z_i = M_i \oplus C_i$ for $i = 0, \dots, u - 2$.

Step 3 Forge with (N, AD, C', T) where $C' = Z_0 \oplus M_{\pi(0)} \| Z_1 \oplus M_{\psi(0)} \| Z_2 \oplus M_{\pi(1)} \| Z_3 \oplus M_{\psi(1)} \| \dots \| C_{i-1}$.

We have $C' \neq C \implies$ the decryption query is valid. This will always pass the verification and returns

$$M_{\pi(0)} \| M_{\psi(0)} \| M_{\pi(1)} \| M_{\psi(1)} \| \dots \| M_{l-1}$$

as the output.

To see the correctness of this forgery, we look at the decryption of (N, AD, C', T) . First note that ciphertext computation is done via counter mode of operation (see Figure 2). Since the counter $T \oplus N$ is same for both encryption and decryption queries, then $M' = M_{\pi(0)} \| M_{\psi(0)} \| M_{\pi(1)} \| M_{\psi(1)} \| \dots \| M_{l-1}$ is obtained (not released yet). Next, to see if the tags of M' and M are same it is enough to show that $\delta'_c = \delta_c$. This follows trivially as the masking value is α and `alpha_x` for each element in S_e and S_o , respectively. So, permutating these sets individually will not change the XOR sum value. Formally, we have

$$\begin{aligned} \delta'_c &= \bigoplus_{\pi(i), i \in S_e} \mathcal{E}_{\text{aeadK}}^n(M_{\pi(i)} \oplus \alpha) \quad \bigoplus_{\psi(i), i \in S_o} \mathcal{E}_{\text{aeadK}}^n(M_{\psi(i)} \oplus \text{alpha_x}) \\ &= \bigoplus_{i \in S_e} \mathcal{E}_{\text{aeadK}}^n(M_i \oplus \alpha) \quad \bigoplus_{i \in S_o} \mathcal{E}_{\text{aeadK}}^n(M_i \oplus \text{alpha_x}) = \delta_c \implies T' = T. \end{aligned}$$

Remark 3. If π and ψ both are identity permutations then $C' = C \implies$ the decryption query is not valid. The number of valid forgeries then equals $\lceil \frac{u}{2} \rceil \lceil \frac{u-1}{2} \rceil - 1$.

Remark 4. Associated Data and Ciphertext Forgery is a direct application of associated data-only and ciphertext-only forgeries.

3.4 Forgeries Associated with Last Block

Until now, we have consider the cases where the last block is not modified. To forge the last block, all the previous blocks before it must contain AD bytes. Assume there is only 1 block and it consists of u bytes of AD and v bytes of M such that $u + v \leq n/8$. The forgery then proceed as follows.

Step 1 Let $N \xleftarrow{\$} \{0, 1\}^n$. Encrypt (N, AD, M) and observe (C, T) .

Step 2 Compute the keystream bytes $Z[i] = M[i] \oplus C[i]$ for $i = 0, \dots, v - 1$

Step 3 For $1 \leq l \leq v$, forge with (N, AD', C, T) where $AD' = AD \| M[0] \| M[l - 1]$ and

$$C' = \begin{cases} \epsilon & \text{if } l = v, \\ Z[0] \oplus M[l] \| \dots \| Z[v - l - 1] \oplus M[v - 1] & \text{o.w.} \end{cases}$$

We have $AD' \neq AD$ and $C' \neq C$. Thus, the decryption query is valid and will pass the verification with probability 1 as $AD' \| M' = AD \| M$. The output is $M' = M[l] \| \dots \| M[v - 1]$. Further note that this is a special case of associated data and ciphertext forgery.

Remark 5. The above forgery incorporates both cases of Remark 1 whether `pad_byte` is XORed to the last byte of block or it is appended after AD and M bytes in case of $u + v < n/8$.

3.5 Experimental Verification

We have verified the attacks using the reference implementation of Limdolen[Meh19]. In Tables 4 we list the examples of forgeries for Limdolen-128. The examples of Limdolen-256 are similar and hence omitted.

4 Specifications of HERN

HERN adopts a stream cipher based construction similar to the CAESAR finalist Acorn [Wu16]. The state consists of four 64-bit registers which are updated in an LFSR based style by feeding the two nonlinear bits a and b to the registers. An overview of core components of HERN is illustrated in Algorithm 1.

Table 4: Examples of forgeries for Limdolen-128

Input data	associated data-only	
K	000102030405060708090A0B0C0D0E0F	000102030405060708090A0B0C0D0E0F
N	6E22729F7CEA8F9E1EDFB968365BF23B	6E22729F7CEA8F9E1EDFB968365BF23B
AD	BEOA1CDB4142106B5F2BB5BC8911E75E	A5687AF34938ED433536D8AB281FED78 5D1808F6DD8D60B23EE9E0E061A5B93 A5687AF34938ED433536D8AB281FED78 5D1808F6DD8D60B23EE9E0E061A5B93 BEOA1CDB4142106B5F2BB5BC8911E75E
M	Empty string	Empty string
C	Empty string	Empty string
T	EF4F60E08694CABB285D3841C433645D	EF4F60E08694CABB285D3841C433645D
Input data	ciphertext-only	
K	000102030405060708090A0B0C0D0E0F	000102030405060708090A0B0C0D0E0F
N	92C2A61831DCDE2EF3DB6060DF03DD0A	92C2A61831DCDE2EF3DB6060DF03DD0A
AD	Empty string	Empty string
M	ACCC9952DBB1CC0C8FA8106D463F483A BF23441F82A4BC61D2BF42AF6E4C1F1A 19B86CF46A3800F9E01066264FAF600E D2A42D5449E9B51BA9F8CB1744EA315D	19B86CF46A3800F9E01066264FAF600E BF23441F82A4BC61D2BF42AF6E4C1F1A ACCC9952DBB1CC0C8FA8106D463F483A D2A42D5449E9B51BA9F8CB1744EA315D
C	07AC6C25FAP2BA41F3B808502BA15F66 13237F247E2777389835C8C5B88BC655 E5EB9286DF5EE3FB8140B3588BC18C11 FBF38906197E5B6E069E50E4D8FABF45	B2D899834B7B76B49C007E1B22317752 13237F247E2777389835C8C5B88BC655 509F67206ED72F0EEEF8C5138251A425 FBF38906197E5B6E069E50E4D8FABF45
T	EDFDE9B652A0FB16A7BFF22FD3B44D8	EDFDE9B652A0FB16A7BFF22FD3B44D8
Input data	associated data and ciphertext	
K	000102030405060708090A0B0C0D0E0F	000102030405060708090A0B0C0D0E0F
N	2E2CC56156A6ACF4D3B1CCE369F4C934	2E2CC56156A6ACF4D3B1CCE369F4C934
AD	0C558F14C1E88FED	0C558F14C1E88FED60D1B7E5BA6EDC
M	60D1B7E5BA6EDC62	62
CT	93C6C56CBBF3B39D	91
T	C248D7D75062DE6163AFC13CADEBC55B	C248D7D75062DE6163AFC13CADEBC55B

Algorithm 1 Core components of HERN

1: function H_core_step:	1: function Adda:	1: function H_enc_step(m):
2: $a \leftarrow \text{SB}(s_{30}^0, s_{29}^0, s_{32}^1, s_{24}^1, s_{31}^2, s_4^2, s_{15}^3, s_{14}^3)$	2: $s_{63}^0 \leftarrow s_{63}^0 \oplus a$	2: H_core_step
3: $b \leftarrow \text{SB}'(s_{30}^0, s_{29}^0, s_{32}^1, s_{24}^1, s_{31}^2, s_4^2, s_{15}^3, s_{14}^3) \oplus s_{32}^0$	3: $s_{63}^1 \leftarrow s_{63}^1 \oplus a$	3: $a \leftarrow a \oplus m$
4: $f^0 \leftarrow s_0^0 \oplus s_{31}^0 \oplus s_{32}^0 \oplus s_{13}^1$	4: function Addb:	4: Adda
5: $f^1 \leftarrow s_0^1 \oplus s_{28}^1 \oplus s_{30}^1 \oplus s_1^2$	5: $s_{63}^1 \leftarrow s_{63}^1 \oplus b$	5: $c \leftarrow b \oplus m$
6: $f^2 \leftarrow s_0^2 \oplus s_{22}^2 \oplus s_{27}^2 \oplus s_{26}^3$	6: $s_{63}^2 \leftarrow s_{63}^2 \oplus b$	6: return c
7: $f^3 \leftarrow s_0^3 \oplus s_8^3 \oplus s_{19}^3 \oplus s_{31}^0$	7: function SB'($x_0, y_0, x_1, y_1, x_2, y_2, x_3, y_3$):	7: H_if_step(x):
8: $s^i \leftarrow s^i \ll 1$, for $i = 0, 1, 2, 3$	8: return $x_0y_2 \oplus y_0y_3 \oplus x_1x_3 \oplus y_1x_2$	8: H_core_step
9: $s_{63}^i \leftarrow f^i$, for $i = 0, 1, 2, 3$		9: $a \leftarrow a \oplus x$
10: function SB($x_0, y_0, x_1, y_1, x_2, y_2, x_3, y_3$):		10: Adda
11: return $1 \oplus x_0y_0 \oplus x_1y_1 \oplus x_2y_2 \oplus x_3y_3$		11: Addb

4.1 Description of HERN AEAD

The HERN AEAD algorithm takes as input a 128-bit key K , 128-bit nonce N , $adlen$ bits associated data AD , m len bits plaintext M and outputs a m len bits ciphertext C and 128-bit authentication tag T . The encryption consists of 3 phases, namely 1) Initialization, 2) Processing plaintext and 3) Finalization, which are described as follows.

4.1.1 Initialization The initialization consists of loading the key K and constants into the state and processing the nonce N , associated data AD and running H_if_step (see Algorithm 1) for 512 steps with zero input.

- Load the state with K and constants.

- Process $N = n_0, n_1, \dots, n_{127}$. At each step, one bit of N is used to update the state, i.e, $H_if_step(n_i)$, for $i = 0, \dots, 127$.
- Process $AD = ad_0, ad_1, \dots, ad_{adlen-1}$. At each step, one bit of AD is used to update the state, i.e, $H_if_step(ad_i)$, for $i = 0, \dots, adlen - 1$.
- Run the H_if_step for 512 steps with zero-stream, i.e., $H_if_step(0)$, for $i = 0, \dots, 511$.

4.1.2 Processing plaintext The plaintext $M = m_0, m_1, \dots, m_{mlen-1}$ is used to update the state bit-by-bit and the corresponding ciphertext bit is generated using the function $H_enc_step(\cdot)$ (see Algorithm 1).

- $C \leftarrow \epsilon$
- $c_i \leftarrow H_enc_step(m_i)$, $C \leftarrow C || c_i$, for $i = 0, \dots, mlen - 1$

4.1.3 Finalization After processing all the plaintext bits, the H_if_step runs for 512 times with zero input, and then the tag is generated.

- $H_if_step(0)$, for $i = 0, \dots, 511$.
- $T \leftarrow \epsilon$
- $t_i \leftarrow H_enc_step(0)$, $T \leftarrow T || t_i$, for $i = 0, \dots, mlen - 1$
- return (C, T)

4.2 Security Claims

Considering the nonce-respecting setting and a data limit of 2^{64} bits (i.e., $adlen + mlen \leq 2^{64}$), the designers claim 128-bit security for confidentiality and integrity.

5 Forgery Attacks on HERN

In this section, we provide the details of forgery attacks on HERN. In particular, we show that a message can be modified by appending or removing a sequence of consecutive ‘0’ bits of length n . Moreover, we show that the best success rate of forgery is achieved for $n = 1$ case.

5.1 Basic Forgery

The adversarial model is similar to Subsection 3.1. In the following, we explain the minimal example of our forgery attack against HERN. For the description of forgeries, we let S_i, a_i, b_i denote the state of HERN and two nonlinear bits a and b at the beginning of i -th round.

5.1.1 Associated data-only forgery Let $1 \leq n \leq 63$ and $K \stackrel{\$}{\leftarrow} \{0, 1\}^{128}$ be fixed. To construct the forgery we proceed as follows.

Step 1 Let $N \stackrel{\$}{\leftarrow} \{0, 1\}^{128}$, $AD \stackrel{\$}{\leftarrow} \{0, 1\}^*$ and $M = \epsilon$. Encrypt (N, AD, M) and observe (C, T) .

Step 2 Repeat Step 1 until we obtain a tag whose first n bits are all zero. Define this query as $Q \stackrel{\text{def}}{=} (N, AD, M, C, T)$.

Step 3 For each $i = 0$ to $2^n - 1$, decrypt (N', AD', C', T') where

$$N' = N, \quad AD' = AD || 0^n, \quad C' = \epsilon$$

$$T' = T \ll n \mid (i_0 || \dots || i_{n-1}), \quad \text{and } (i_0, \dots, i_{n-1}) \stackrel{1}{\leftarrow} i.$$

If the verification succeeds with output as an empty plaintext, we stop.

The decryption queries are valid as $AD' \neq AD$ and $T' \neq T$. To see why such a query work, consider the encryption of Q and $Q' \stackrel{\text{def}}{=} (N, AD', \epsilon)$. This is illustrated in Lemma 2.

Lemma 2. *Let Q and Q' be defined as above and $|AD| = u$. Then $T' = T \ll n \mid \Delta$ where Δ is an n -bit string.*

Proof. After processing 128 bits of nonce and first u bits of AD , the states are same, i.e., $S_{128+u} = S'_{128+u}$. For query Q , as M is empty, $\text{H.lif_step}(\cdot)$ runs for 1024 times with zero input. For Q' , since $AD' = AD \parallel 0^n$ and $M' = \epsilon$, $\text{H.lif_step}(\cdot)$ is iterated for $n + 1024$ times with zero bit. The tag generation phase for Q and Q' starts from S_{1152+u} and $S'_{1152+u+n}$, respectively.

Note that the first n bits of T are zero and they are not added to the state. This is equivalent to the fact that $\text{H.lif_step}(0)$ runs for another n times starting from round $1152+u$. Hence, $S_{1152+u+n} = S'_{1152+u+n} \implies$ the last $128 - n$ bits of T are the same as the first $128 - n$ bits of T' . Since the states are unknown, the last n bits of T' has to be guessed. Thus, $T' = T \ll n \mid \Delta$. \square

Attack complexities. On average, step 2 requires 2^n encryption queries while step 3 needs 2^n decryption queries. Thus, for $1 \leq n \leq 63$, the success rate of forgery is 2^{-n} . For $n = 1$ the success rate is 2^{-1} after querying encryption oracle 2 times. This clearly violates the designers claim that success rate of forgery is 2^{-127} after two encryption queries.

5.1.2 Ciphertext-only forgery Let $1 \leq n \leq 31$ and $K \stackrel{\$}{\leftarrow} \{0, 1\}^{128}$ be fixed. We construct forgery as follows.

Step 1 Let $N \stackrel{\$}{\leftarrow} \{0, 1\}^{128}$, $AD \stackrel{\$}{\leftarrow} \{0, 1\}^*$, $M \leftarrow \{0, 1\}^{\geq 1}$. Encrypt $(N, AD, M \parallel 0^n)$ and observe (C, T) .

Step 2 Repeat Step 1 until a ciphertext whose last n bits are zero is obtained. Denote this query by (N, AD, M, C, T) .

Step 3 Decrypt (N', AD', C', T') where

$$\begin{aligned} N' &= N, \quad AD' = AD, \quad C' = c_0 \parallel \dots \parallel c_{|M|-n-1} \\ T' &= 0^n \mid T \gg n. \end{aligned}$$

Step 4 If verification fails, repeat Step 2 and Step 3.

We have $C' \neq C$ as the lengths are different and $T' \neq T$. Thus, each query in step 3 is a valid decryption query. Upon successful verification, only first $|M|-n$ bits of M are returned. A formal proof of correctness of decryption query is given in Lemma 3.

Lemma 3. *Let $Q \stackrel{\text{def}}{=} (N, AD, M)$ satisfy Step 2 with output as (C, T) . Let $AD' = AD$, $M' = m_0 \parallel \dots \parallel m_{|M|-n-1}$ and $Q' \stackrel{\text{def}}{=} (N, AD', M')$. Then $T' = 0^n \mid T \gg n$ iff the n nonlinear bits $b_{1152+|AD|+|M|-n}, \dots, b_{1152+|AD|+|M|-1}$ are all zero.*

Proof. The proof is similar to Lemma 2 and hence omitted. \square

Attack complexities. Step 2 requires 2^n encryption queries (on average), while to satisfy both Step 2 and Step 3 simultaneously, 2^{2n} encryption queries (on average) are needed. Thus, for $1 \leq n \leq 31$, the success rate of forgery is 2^{-n} after observing output of 2^{2n} encryption queries. The value of n is chosen to satisfy the data limit restriction of 2^{64} bits.

Remark 6. Similar to associated data-only forgery, the best success rate is achieved for $n = 1$ case which is 2^{-1} after 4 encryption queries.

5.1.3 Associated data and ciphertext forgery Let $1 \leq n \leq 63$ and $K \xleftarrow{\$} \{0, 1\}^{128}$ be fixed. The forgery then proceed as follows.

Step 1 Let $N \xleftarrow{\$} \{0, 1\}^{128}$, $AD \xleftarrow{\$} \{0, 1\}^*$, $M = 0^n$. Encrypt (N, AD, M) and observe (C, T) .

Step 2 Repeat step 1 until we obtain $C = 0^n$. Denote this query by (N, AD, M, C, T) .

Step 3 Forge with (N', AD', C', T') where

$$N' = N, AD' = AD||0^n, C' = \epsilon \text{ and } T' = T,$$

which will always be successful (with empty message as an output) as the states after $640 + |AD| + n$ rounds are same. The proof is similar to Lemma 2 and 3, and hence omitted.

Attack complexities. Step 2 requires 2^n encryption queries on average, while step 3 requires only a single decryption query. Thus, for $1 \leq n \leq 63$, the success rate of forgery is 1.

5.2 Experimental Verification

We have verified the attacks using the reference implementation of HERN [YSMW19]. In Table 5, we list the examples for $n = 8$.

Table 5: Examples of forgeries for HERN

Input data	associated data-only	
K	000102030405060708090A0B0C0D0E0F	000102030405060708090A0B0C0D0E0F
N	D8A4ADC965EECE56330E5CC01A53C928	D8A4ADC965EECE56330E5CC01A53C928
AD	CA5F	CA5F00
M	Empty string	Empty string
CT	Empty string	Empty string
T	00FC40BF26954B37993E9C56C6C49ACA	FC40BF26954B37993E9C56C6C49ACAB6
Input data	ciphertext-only	
K	000102030405060708090A0B0C0D0E0F	000102030405060708090A0B0C0D0E0F
N	3E1327BCC61246AC87901E0922C1A354	3E1327BCC61246AC87901E0922C1A354
AD	9524	9524
M	8500	85
CT	0D00	0D
T	8472B9D92F6AAC22CE3F188CC13D711C	008472B9D92F6AAC22CE3F188CC13D71
Input data	associated data and ciphertext	
K	000102030405060708090A0B0C0D0E0F	000102030405060708090A0B0C0D0E0F
N	7B8A185D3B33E4F906E02F291BEF6C06	7B8A185D3B33E4F906E02F291BEF6C06
AD	4328	432800
M	00	Empty string
CT	00	Empty string
T	A72C78D89FAD7A7D785EF13AB2EC085B	A72C78D89FAD7A7D785EF13AB2EC085B

6 Concluding Remarks

We have demonstrated a series of practical forgery attacks on Limdolen and HERN in the nonce-respecting scenario. Our attacks defeat the designer’s claim of 128(256) and 128-bit integrity security of Limdolen-128(256) and HERN, respectively. For both variants of Limdolen, we have shown the constructions of associated data-only, ciphertext-only and associated data and ciphertext forgeries which require a single encryption and a single decryption query, and have a successful probability of 1. The crux of our forgery attacks lie in Lemma 1 and the observation that only a sequence of period 2 consisting of $(\alpha, \alpha \oplus x)$ is used for masking. Moreover, we have found a discrepancy for the padding in the specification document and reference implementation (see Remark 1). However, the presented attacks are independent of this

inconsistency. To resist our attacks, the period 2 masking sequence has to be replaced by a sequence with unpredictable properties.

For HERN, we have found that associated data and message processing phases are not distinguishable. As a result one can modify a message by appending or removing a sequence of zero bits. Accordingly, we have presented round independent associated data-only, ciphertext-only and associated data and ciphertext forgeries with the success rate of 1 after 2(2), 4(2) and 2(1) encryption(decryption) queries, respectively. A simple fix to resist our attack is to complement a state bit (except the last bit of each register) after $640 + |AD|$ and $640 + |AD| + |M|$ clock cycles.

Acknowledgement

This work is supported by NSERC.

References

- [BR02] John Black and Phillip Rogaway. A block-cipher mode of operation for parallelizable message authentication. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 384–397. Springer, 2002.
- [cae] CAESAR: Competition for Authenticated Encryption: Security, Applicability, and Robustness. <https://competitions.cr.yp.to/caesar.html>.
- [IIMP19] Akiko Inoue, Tetsu Iwata, Kazuhiko Minematsu, and Bertram Poettering. Cryptanalysis of OCB2: Attacks on Authenticity and Confidentiality. *Cryptology ePrint Archive*, Report 2019/311, 2019. <https://eprint.iacr.org/2019/311>.
- [ISO] ISO: Information Technology - Security techniques - Authenticated encryption, ISO/IEC 19772:2009. International Standard ISO/IEC 19772 (2009).
- [Kha19] Mustafa Khairallah. Forgery Attack on SNEIKEN. 2019. <https://eprint.iacr.org/2019/408>.
- [Meh19] Carl E. Meher. Limdolen: A Lightweight Authenticated Encryption Algorithm. NIST LWC Round 1 Submission. 2019. <https://csrc.nist.gov/CSRC/media/Projects/Lightweight-Cryptography/documents/round-1/spec-doc/Limdolen-Spec.pdf>.
- [NIS19] NIST lightweight cryptography standardization process. <https://csrc.nist.gov/projects/lightweight-cryptography>, accessed 31 May 2019.
- [Per19] Leo Perrin. Probability 1 Iterated Differential in the SNEIK Permutation. *Cryptology ePrint Archive*, Report 2019/374, 2019. <https://eprint.iacr.org/2019/374>.
- [Rog02] Phillip Rogaway. Authenticated-encryption with associated-data. In *Proceedings of the 9th ACM conference on Computer and communications security*, pages 98–107. ACM, 2002.
- [RS07] Phillip Rogaway and Thomas Shrimpton. The SIV mode of operation for deterministic authenticated-encryption (key wrap) and misuse-resistant nonce-based authenticated-encryption. *Aug*, 20:3, 2007.
- [Saa19] Markku Juhani O. Saarinen. SNEIKEN and SNEIKHA authenticated encryption and cryptographic hashing. NIST LWC Round 1 Submission, 2019. Available online at <https://github.com/pqshield/sneik>.
- [Wu16] Hongjun Wu. ACORN: a lightweight authenticated cipher (v3). 2016. <https://competitions.cr.yp.to/round3/acornv3.pdf>.
- [YSMW19] Dingfeng Ye, Danping Shi, Yuan Ma, and Peng Wang. HERN and HERON: Lightweight AEAD and Hash Constructions based on Thin Sponge (v1). 2019. <https://csrc.nist.gov/CSRC/media/Projects/Lightweight-Cryptography/documents/round-1/spec-doc/>.