Authenticated Encryption for Very Short Inputs

Alexandre Adomnicăi[1], Kazuhiko Minematsu[2,3⋆], and Junji Shikata[3]

[1] Independent researcher, Paris, France, alexandre@adomnicai.me
[2] NEC, Kawasaki, Japan, k-minematsu@nec.com
[3] Yokohama National University, Yokohama, Japan, shikata-junji-rb@ynu.ac.jp

Abstract. We study authenticated encryption (AE) modes dedicated to very short messages, which are crucial for Internet-of-things applications. Since the existing general-purpose AE modes need at least three block cipher calls for non-empty messages, we explore the design space for AE modes that use at most two calls. We proposed a family of AE modes, dubbed Manx, that work when the total input length is less than $2n$ bits, using an n-bit block cipher. Notably, the second construction of Manx can encrypt almost n-bit plaintext and saves one or two block cipher calls from the standard modes, such as GCM or OCB, keeping the comparable provable security. We also present benchmarks on popular 8/32-bit microprocessors using AES. Our results show the clear advantage of Manx over the previous modes for such short messages.

Keywords: Authenticated encryption, Block cipher, Short inputs, Internet-of-Things

1 Introduction

Authenticated encryption (AE) is a symmetric-key cryptography function that provides both confidentiality and integrity of the input. AE can be realized by a mode of operation with a block cipher. Building such an AE mode has been one of the central topics since the concept of AE was established in the early 2000s [13, 22, 31]. A general guideline for designing AEs is that they must be able to accept messages of sufficient length. For example, GCM [1] is one of two NIST-recommended AE modes. It is widely deployed and can handle a single message of about 68 GBytes. The ongoing NIST lightweight cryptography (NIST LwC), which is a competition for lightweight AE schemes, requires 2^{50} bytes as the maximum input length in its call for algorithms [4].

On the other hand, the rise of applications using wireless communication with small devices – also known as Internet-of-Things (IoT) – has created a demand for AEs specializing in short inputs. We can, of course, pick a popular scheme from those used by (say) TLS, but their performances on short inputs are not always satisfactory for the limited computational resources. The performance problem of standard modes for short inputs was suggested by Iwata et al. [21],

⋆ This work was conducted as part of his duties at Yokohama National University.

and they proposed an AE mode aiming at reducing the computational overhead for short inputs. Since then, this problem has been acknowledged in the research community; for example, some NIST LwC proposals, including the finalists, feature good performance on short inputs, e.g., Ascon [18], ForkAE [7], and Romulus [19]. However, these schemes also support a sufficiently long input, as mentioned above.

Known AE modes, such as GCM, CCM [2] (another NIST-recommended mode), OCB [25][4], and COFB [9,17] require 3 to 5 block cipher calls for any non-empty message. This observation raises a natural question: **what AE modes are possible with at most two block cipher calls?**
Of course, the acceptable input should be very short, and we are interested in what input length could be covered by such two-call schemes. Our question may be insignificant for general-purpose protocols. Yet it is practically relevant in the field of IoT, where each message is very short, and one block cipher call often occupies a significant amount of the total computation. For example, Sigfox limits packet length up to 12 bytes [3], EnOcean limits 9 or 14 bytes [5] and Electronic Product Code used by the RFID protocol has a 96-bit message. NIST LwC call for algorithms states that efficiency for short messages, such as 8 bytes, is one of the evaluation measures. In principle, even a 1-bit message is sufficient for some applications such as device monitoring. Malik et al. [28] showed that 1 to 4 bytes are enough for healthcare applications for tiny medical sensors using Narrow-Band IoT standards. See the work by Andreeva et al. [8] for more examples. From a computational viewpoint, on 8-bit AVR microprocessors, one call to AES-128 takes more than 2,000 cycles [16, 24], so reducing a few block cipher calls would significantly improve the performance.

Our Contributions. We propose a family of two AE modes, dubbed Manx[5], that are dedicated to very short inputs. More concretely, Manx uses an n-bit block cipher E, and for the input consisting of ν-bit nonce, α-bit associated data, and ℓ-bit message, it works when $\sigma := \nu + \alpha + \ell$ is (roughly) at most $2n$ with certain restrictions on the parameters (ν, α, ℓ), using at most two calls of E. In particular, the first mode Manx1 allows $\nu + \alpha \approx 2n$ but limits $\ell < n - \tau$ to achieve τ-bit authenticity, while the second mode Manx2 allows $\ell \approx n$ if $\nu = \tau = n/2$. Moreover, Manx2 allows parallel implementation. By setting $\tau = n/2$, Manx2 is the first two-call mode without precomputation that supports about n-bit messages with $n/2$-bit security (thus the security is comparable to GCM or OCB).

Manx has some similarities to the classical Encode-then-Encipher (EtE) [15], however, the original EtE clearly does not work when σ exceeds n. By definition, the EtE uses just one call. Therefore, our work bridges the gap between the most primitive AE mode, i.e., EtE, and the general-purpose AE modes.

We do not claim the ultimate novelties of our proposals. However, we are unaware of any work on building concrete and optimized block cipher modes specialized on a range of very short inputs beyond the original EtE. We provide

[4] We mean the latest OCB3 [25] throughout the paper.
[5] Manx are felines with very short tails.

security proofs for the standard AE security notions, namely privacy and authenticity. The proved bounds for both schemes are comparable to the existing popular modes. The proofs are relatively straightforward but need some care for their unique structure to avoid trivial breaks, particularly for Manx2.

We implement Manx1 and Manx2 using AES-128 as the underlying block cipher and compare them with common modes on 8-bit AVR and 32-bit ARM microprocessors, which are widely deployed in many IoT use cases. Our implementation results show a clear advantage in favor of Manx over the other modes; for example, on ARM Cortex-M4, Manx2 with $(\nu, \alpha, \ell) = (64, 16, 44)$ runs around 5.2K cycles, whereas GCM and CCM run around 14K and 11K cycles, respectively. For more details, refer to Section 4.

2 Preliminaries

For integers $1 \leq i < j$, let $[i..j] \coloneqq \{i, i+1, \ldots, j\}$ and $[i] \coloneqq [1..i]$. Let $\{0,1\}^*$ be the set of all finite bit strings. For $X \in \{0,1\}^*$, $|X|$ is its length in bits. The empty string is denoted by ε and $|\varepsilon| = 0$. Let $\{0,1\}^{\leq b}$ denote $\bigcup_{i=0,1,\ldots,b}\{0,1\}^i$, where $\{0,1\}^0 = \{\varepsilon\}$. For two bit-strings X and Y, $X \,\|\, Y$ is their concatenation. We also write this as XY if it is clear from the context. Let 0^i be the string of i zero bits; for instance, we write 10^i for $1 \,\|\, 0^i$. For $X \in \{0,1\}^*$ with $|X| \geq i$, $\mathtt{msb}_i(X)$ is the first (left) i bits of X, and $\mathtt{lsb}_i(X)$ is the last (right) i bits of X. If X is uniformly chosen from the set $\mathcal{X}$, we write $X \xleftarrow{\$} \mathcal{X}$.

Let $\mathtt{pad}_{n'} : \{0,1\}^{\leq n} \to \{0,1\}^{n'}$ for any $n' \geq n$ denote a so-called one-zero (possibly non-injective) padding: $\mathtt{pad}_{n'}(X) = X \| 10^{n'-|X|-1}$ when $|X| < n$ and $\mathtt{pad}_{n'}(X) = X$ when $|X| = n$ and $n' = n$. We define the (pseudo) inversion $\mathtt{depad}_{n'} : \{0,1\}^{n'} \to \{0,1\}^{\leq n}$ by removing the last $100\ldots$ sequence of the input $Y \in \{0,1\}^{n'}$. If $Y = 0^{n'}$, let $\mathtt{depad}_{n'}(Y)$ be any fixed constant. Note that if we know that the input to $\mathtt{pad}_{n'}$ is shorter than n in advance or $n' > n$ is ensured, $\mathtt{pad}_{n'}$ is injective, and its inverse is uniquely determined by $\mathtt{depad}_{n'}$.

For any $X \in \{0,1\}^*$ and a positive integer n, $X[1], X[2], \ldots, X[m] \xleftarrow{n} X$ denotes the parsing into n-bits, i.e., $X[1] \,\|\, X[2] \,\|\, \cdots \,\|\, X[m] = X$ and $|X[i]| = n$ for all $i < m$, $|X[m]| \in [n]$. By extending the notation, we write $X[1], \ldots, X[a] \xleftarrow{l_1, l_2, \ldots, l_a} X$ such that $X[1] \,\|\, \cdots \,\|\, X[a] = X$ and $|X[i]| = l_i$ for all $i \in [a]$, assuming $\sum_{i \in [a]} l_i = |X|$.

Fields with 2^n points. We interchangeably view an element $a = (a_{n-1} \cdots a_1 a_0) \in \{0,1\}^n$ as a point in $\mathrm{GF}(2^n)$ as a coefficient vector of the corresponding polynomial: $a(\mathtt{x}) = \sum_{i=0}^{n-1} a_i \mathtt{x}$. Following [32], by writing $2a$ for $a \in \{0,1\}^s$, we mean a multiplication over $\mathrm{GF}(2^s)$ by the polynomial $\mathtt{x}$, also called doubling. Similarly, $3a$ means a multiplication by $\mathtt{x}+1$, i.e. $3a = 2a \oplus a$. As popularized by [20, 32], these operations are quite efficient. For example, by taking the lexicographically first irreducible polynomial for $n = 128$, which is $\mathtt{u}^{128} + \mathtt{u}^7 + \mathtt{u}^2 + \mathtt{u} + 1$, $2a$ means $a \ll 1$ if $a_{127} = 0$, and $(a \ll 1) \oplus 0^{120}10000111$ otherwise.

(Tweakable) Block Ciphers and Random Primitives. A tweakable block cipher (TBC) [27] is a keyed function $\widetilde{E} : \mathcal{K} \times \mathcal{T}_w \times \mathcal{M} \to \mathcal{M}$ such that for each $(K, T) \in \mathcal{K} \times \mathcal{T}_w$, $\widetilde{E}(K, T, \cdot)$ is a permutation over $\mathcal{M}$. Here, K is a key, and T is a public value called a tweak. The encryption of a plaintext $M \in \mathcal{M}$ with a key $K \in \mathcal{K}$ and a tweak $T \in \mathcal{T}_w$ is a ciphertext $C = \widetilde{E}(K, T, M)$. It is also written as $\widetilde{E}_K(T, X)$. Similarly, the decryption is written as $M = \widetilde{E}^{-1}(K, T, C)$ or $\widetilde{E}_K^{-1}(T, C)$. Note that a conventional block cipher $E : \mathcal{K} \times \mathcal{M} \to \mathcal{M}$ is equivalent to a TBC with $|\mathcal{T}_w| = 1$. We write $E_K^{-1}(*)$ to denote the decryption function.

Let $\mathrm{TPerm}(\mathcal{T}_w, \mathcal{M})$ denote the set of all tweakable permutations over $\mathcal{M}$ with tweak space $\mathcal{T}_w$, and let $\mathrm{Perm}(\mathcal{M})$ be the set of all permutations over $\mathcal{M}$. A tweakable uniform random permutation (TURP) of tweak space $\mathcal{T}_w$ and message space $\mathcal{M}$ is a random tweakable permutation uniformly sampled from $\mathrm{TPerm}(\mathcal{T}_w, \mathcal{M})$. It is denoted as $\widetilde{\mathsf{P}} : \mathcal{T}_w \times \mathcal{M} \to \mathcal{M}$. Similarly, a uniform random permutation (URP) of message space $\mathcal{M}$ is a random permutation uniformly sampled from $\mathrm{Perm}(\mathcal{M})$. It is denoted as $\mathsf{P} : \mathcal{M} \to \mathcal{M}$. Their inverses are denoted by $\widetilde{\mathsf{P}}^{-1}$ and P^{-1}, respectively, where $\widetilde{\mathsf{P}}^{-1}$ additionally takes a tweak.

2.1 Authenticated Encryption

We describe the syntax of nonce-based AE (NAE). Let $\mathsf{NAE} = (\mathsf{NAE}.\mathcal{E}, \mathsf{NAE}.\mathcal{D})$ be an NAE scheme. The (deterministic) encryption algorithm $\mathsf{NAE}.\mathcal{E}$ takes a key $K \in \mathcal{K}$ and a tuple (N, A, M) consisting of a nonce $N \in \mathcal{N}$, an associated data (AD) $A \in \mathcal{A}$, and a plaintext $M \in \mathcal{M}$ as input, and returns a ciphertext $C \in \mathcal{M}$. Note that $|C| > |M|$ must hold for authenticity. For some AE modes, the output may also be written as a tuple (C, T) where T denotes the fixed-length tag, but we adopt this unified syntax for notational compatibility with our schemes. The (deterministic) decryption algorithm $\mathsf{NAE}.\mathcal{D}$ takes $K \in \mathcal{K}$ and the tuple $(A, X) \in \mathcal{A} \times \mathcal{X}$ as input, where $\mathcal{X} = \{0,1\}^*$, and returns $M \in \mathcal{M}$ or the reject symbol $\perp$. We assume that when C is received by querying (N, A, M) to $\mathsf{NAE}.\mathcal{E}_K$, the *trivial* decryption query (A, X) is always uniquely determined by the tuple (N, A, M, C). By trivial, we mean that $\mathsf{NAE}.\mathcal{D}_K(A, X)$ returns M with probability one. Our proposals meet this assumption.

Note that our syntax for decryption is slightly more general than the usual one (which specifies the tuple (N, A, C) as input, so N is explicit). We use this syntax for its affinity with our proposals. Some of our proposals contain N as a part of X, but some do not, depending on the input length. We remark that the AD must be sent in clear (as this is the definitional requirement), but the nonce is not necessarily transmitted in clear to ensure the standard NAE security (Def. 1). We also remark that we do not consider security notions for nonce-hiding AEs [14]. We use the abovementioned point to save bandwidth in one of our proposals.

2.2 Security Notions

Let A be an adversary that queries an oracle $\mathcal{O}$. We say A is a distinguisher if it outputs $x \in \{0,1\}$ as an outcome. If the outcome is 1, we write $\mathsf{A}^{\mathcal{O}} = 1$ to

denote this event. It is a probabilistic event whose randomness comes from those of A and $\mathcal{O}$. Queries of A may be adaptive unless otherwise specified. If there are multiple oracles, $\mathcal{O}_1, \mathcal{O}_2, \ldots$ then $\mathsf{A}^{\mathcal{O}_1,\mathcal{O}_2,\cdots}$ means that A can query any oracle in $\mathbf{O}$ in an arbitrary order.

Let $\mathcal{O}$ and $\mathcal{O}'$ be the oracles. For an adversary A thats is a distinguisher for $\mathcal{O}$ and $\mathcal{O}'$ using adaptive queries, we define the indistinguishability as

$$\mathbf{Adv}^{\mathsf{ind}}_{\mathcal{O},\mathcal{O}'}(\mathsf{A}) \coloneqq |\Pr[\mathsf{A}^{\mathcal{O}} = 1] - \Pr[\mathsf{A}^{\mathcal{O}'} = 1]|.$$

For two (tuples of) oracles, $\mathbf{O} = (\mathcal{O}_1, \mathcal{O}_2, \ldots, \mathcal{O}_s)$ and $\mathbf{O}' = (\mathcal{O}'_1, \mathcal{O}'_2, \ldots, \mathcal{O}'_s)$, $\mathbf{Adv}^{\mathsf{ind}}_{\mathbf{O},\mathbf{O}'}(\mathsf{A})$ is defined as $|\Pr[\mathsf{A}^{\mathcal{O}_1,\mathcal{O}_2,\ldots,\mathcal{O}_s} = 1] - \Pr[\mathsf{A}^{\mathcal{O}'_1,\mathcal{O}'_2,\ldots,\mathcal{O}'_s} = 1]|$.

For a TBC: $\widetilde{E}_K : \mathcal{T}_w \times \mathcal{M} \to \mathcal{M}$, we define the tweakable strong pseudorandom permutation (TSPRP) advantage and the tweakable pseudorandom permutation (TPRP) advantage against an adversary A as

$$\begin{aligned}
\mathbf{Adv}^{\mathsf{tsprp}}_{\widetilde{E}}(\mathsf{A}) &\coloneqq \mathbf{Adv}^{\mathsf{ind}}_{(\widetilde{E}_K,\widetilde{E}_K^{-1}),(\widetilde{\mathsf{P}},\widetilde{\mathsf{P}}^{-1})}(\mathsf{A}),\\
\mathbf{Adv}^{\mathsf{tprp}}_{\widetilde{E}}(\mathsf{A}) &\coloneqq \mathbf{Adv}^{\mathsf{ind}}_{\widetilde{E}_K,\widetilde{\mathsf{P}}}(\mathsf{A})
\end{aligned}$$

where $\widetilde{\mathsf{P}}$ is a TURP with tweak space $\mathcal{T}_w$ and message space $\mathcal{M}$. For a block cipher $E_K : \mathcal{M} \to \mathcal{M}$, we similarly define SPRP and PRP advantages as

$$\begin{aligned}
\mathbf{Adv}^{\mathsf{sprp}}_{E}(\mathsf{A}) &\coloneqq \mathbf{Adv}^{\mathsf{ind}}_{(E_K,E_K^{-1}),(\mathsf{P},\mathsf{P}^{-1})}(\mathsf{A}),\\
\mathbf{Adv}^{\mathsf{prp}}_{E}(\mathsf{A}) &\coloneqq \mathbf{Adv}^{\mathsf{ind}}_{E_K,\mathsf{P}}(\mathsf{A}),
\end{aligned}$$

where P is a URP over $\mathcal{M}$.

We define the following privacy and authenticity notions for NAE. The definitions mostly follow the standard ones; we just need to reflect the change in the decryption syntax.

Definition 1. *Let* $\mathsf{NAE} = (\mathsf{NAE}.\mathcal{E}, \mathsf{NAE}.\mathcal{D})$ *be an NAE scheme. We define*

$$\begin{aligned}
\mathbf{Adv}^{\mathsf{priv}}_{\mathsf{NAE}}(\mathsf{A}_1) &\coloneqq |\Pr[\mathsf{A}_1^{\mathsf{NAE}.\mathcal{E}_K} = 1] - \Pr[\mathsf{A}_1^{\$} = 1]|,\\
\mathbf{Adv}^{\mathsf{auth}}_{\mathsf{NAE}}(\mathsf{A}_2) &\coloneqq |\Pr[\mathsf{A}_2^{\mathsf{NAE}.\mathcal{E}_K,\mathsf{NAE}.\mathcal{D}_K} \text{ forges}]|,
\end{aligned}$$

where $\$$ *denotes a random-bit oracle that returns a uniformly random string of* $|\mathsf{NAE}.\mathcal{E}_K(N, A, M)|$ *bits for any query* (N, A, M)*. The probability spaces are defined over the experiment* $K \xleftarrow{\$} \mathcal{K}$ *and the possible internal randomness of the adversary. We say* $\mathsf{A}_2^{\mathsf{NAE}.\mathcal{E}_K,\mathsf{NAE}.\mathcal{D}_K}$ *forges if* A_2 *makes a non-trivial decryption query* (A', X') *and receives any* $M \neq \perp$*, i.e., there is no previous encryption query* (N, A, M) *and its response* C *that determines* (A', X') *as a trivial decryption query. We require* A_1 *and* A_2 *to be nonce-respecting, i.e., using unique nonce for each encryption query. Note that the authenticity adversary* A_2 *has no restriction on the nonces used by the decryption queries.*

For a list of adversary parameters θ (such as the number of queries) and a security notion sec, we write θ-sec adversary to mean an adversary using θ that plays a game defined by the notion sec. In particular, for priv and auth notions of NAE, we use q_e and q_d to denote the number of encryption and decryption queries and t to denote the time complexity.

3 AE modes for very short inputs

3.1 Minimum calls of existing modes

Let us briefly summarize the minimum number of n-bit block cipher calls for any non-empty plaintext for the existing general purpose (i.e., supporting long inputs) modes. First, it is four for OCB (as of version 3 [25]); two for generating the masks and two for encryption and authentication. In the case of GCM, n is fixed to 128, and it needs three calls plus two $\mathrm{GF}(2^{128})$ multiplications when the nonce is 96 bits; otherwise, two more multiplications are required for any shorter nonce. Compared with them, COFB [9, 17] is a better scheme in this respect; it needs three calls to encrypt a single-block message[6]. CCM needs four calls. See also Table 1.

3.2 What can be done in 1 call?

Encode-then-Encipher [15] is the only viable approach if we use just one call. Using EtE, we encrypt the vector $V = (0^c, N, A, M)$ for some fixed $c > 0$ and obtain $C = E_K(f(V))$ using a one-to-one encoding function f, and send (N, A, C) to the receiver. The verification is done by checking if $\mathtt{msb}_c(E_K^{-1}(C))$ is 0^c. A slight improvement could be achieved by Khovratovich at CT-RSA 2014 [23]. What [23] shows is a permutation-based EtE for deterministic AE [33]. However, the core idea is also applicable to a block cipher-based NAE. The idea is to verify if $(0^c, \mathrm{hash}(A))$ is correctly recovered from $E_K^{-1}(C')$ for the received (A', C'), instead of just checking if 0^c is correctly recovered. This generally extends the possible input length for M as long as $c + |\mathrm{hash}(A)|$ is guaranteed not to be smaller than the required authenticity bit security.

EtE is ultimately simple. However, it is clearly impossible to handle the case of $|N| + |A| + |M| > n$.

3.3 What can be done in 2 calls?

For input (N, A, M), let $\sigma = \nu + \alpha + \ell$ where $|N| = \nu$, $|A| = \alpha$, and $|M| = \ell$. We explore the possibility for AE when σ may exceed n, allowing up to two n-bit block cipher calls. In contrast to the case of one-call schemes (Sect 3.2), the design space for two-call schemes significantly expands. To make the analysis feasible, we set the following assumptions: (1) ν is fixed, and (2) $0 \le \alpha \le \alpha_{\max}$ for some

[6] There are several versions, and we mean (the mode part of) GIFT-COFB [9], which uses GIFT [10] as the internal block cipher. It is one of the NIST LwC finalists.

predetermined $\alpha_{\max}$, irrespective of the plaintext length. Both are reasonable, e.g., $\nu = 96$ is a typical choice for GCM, and AD is often used as a protocol header having a short fixed length. We also impose several assumptions to exclude "cheating" constructions for efficiency consideration. We first assume that there are no cryptographic primitives other than the block cipher, assume the key is the single block cipher key, and exclude the use of a universal hash function, e.g., GHASH of GCM. We exclude any pre-computation beyond the block cipher's key schedule for efficiency and simplicity.

We remark that these limitations are still inherently not rigorous. Say, we can extend the nonce/AD space for our first proposal (Manx1) by one bit with little complexity using tripling ($\mathrm{GF}(2^n)$ multiplication by `x+1`). In more detail, we use either $2 \cdot 3S$ or $2S$ as the offset of block cipher input (at line 5 of the left of Fig. 1) depending on the extra bit. One can view this as a universal hash function of a single bit [32]. By using more constants in $\mathrm{GF}(2^n)$, we can significantly extend the nonce/AD space in principle, but this effectively implements a full field multiplication, which is costlier and conflicts with our assumption of the no-universal hash function. Similarly, small-input universal hash functions can be quite efficient (still, it needs an independent key), such as the stretch-then-shift function proposed by [25].

With these considerations, we keep our goal simple and do not try to specify the ultimately clear borderline on allowed operations beyond block cipher calls. Finally, to achieve the standard model security (as GCM or OCB), we require that the block cipher key is not changed during encryption/decryption. If we use AES-128 (thus $n = 128$), a typical setting would be $\nu \in [64..128]$, but our schemes support shorter value for ν. Whether small ν is acceptable or not is beyond our scope. For the security goal, we set $n/2$ and $\tau \in [n]$ as the desired security level in bits for privacy and authenticity notions, following GCM and OCB. It turns out that the achievable range of τ has some restrictions depending on the scheme and other parameters.

We must impose $\sigma \leq 2n$ since otherwise, the whole encryption query cannot be processed by the block cipher, implying the break of the privacy notion. Hence, we explore two-call AE modes within this $\sigma \leq 2n$ restriction.

3.4 Manx1 based on XEX

One natural way to extend the single-block EtE shown above is to add a *mask* to the input and output of EtE, by generating a mask using another block cipher call. The mask-generating call can extend the input space. More specifically, we can use a mode that turns a block cipher into a TBC, such as XEX [32]. Below we present an XEX-based two-call AE mode, Manx1. For generality, we introduce a vector encoding function $\mathsf{vencode} : \mathcal{N} \times \mathcal{A} \to \mathcal{M} \times \mathcal{V}$ for $\mathcal{M} = \{0,1\}^n$ and $\mathcal{V} \subseteq \{0,1\}^{\leq n}$.

Definition 2. *For* $\mathsf{vencode} : \mathcal{N} \times \mathcal{A} \to \mathcal{M} \times \mathcal{V}$*, let* $(V[1], V[2]) = \mathsf{vencode}(N, A)$ *for* $N \in \mathcal{N} (= \{0,1\}^\nu)$ *and* $A \in \mathcal{A}$*. For* $\mathcal{N}$ *and* $\mathcal{A}$*,* vencode *is sound with respect to* $\mathcal{N}$ *and* $\mathcal{A}$ *if* (N, A) *is uniquely determined by* $(V[1], V[2])$ *and* $\mathcal{V} = \{0,1\}^{v_2}$*,*

Algorithm Manx1.$\mathcal{E}[E_K](N, A, M)$

$(V[1], V[2]) \leftarrow \mathsf{vencode}(N, A)$
$v_2 \leftarrow |V[2]|$
$\overline{M} \leftarrow \mathtt{pad}_{n-v_2}(M)$
$S \leftarrow E_K(V[1])$
$S \leftarrow 2S$
$C \leftarrow E_K(S \oplus (V[2] \,\|\, \overline{M})) \oplus S$
return C

Algorithm Manx1.$\mathcal{D}[E_K](A, X)$

if $|X| \neq n + \nu$ **then**
  return $\perp$
$(N, C) \xleftarrow{\nu,n} X$
$(V[1], V[2]) \leftarrow \mathsf{vencode}(N, A)$
$v_2 \leftarrow |V[2]|$
$S \leftarrow E_K(V[1])$
$S \leftarrow 2S$
$Y \leftarrow E_K^{-1}(S \oplus C) \oplus S$
$(\widetilde{V}[2], \widetilde{M}) \xleftarrow{v_2, n-v_2} Y$
if $\widetilde{V}[2] \neq V[2]$ **then**
  return $\perp$
else
  $M \leftarrow \mathtt{depad}_{n-v_2}(\widetilde{M})$
  return M

Fig. 1: The algorithms of Manx1. The sender transmits (A, X) via the channel, where $X = (N \,\|\, C)$.

where v_2 is a fixed positive integer not smaller than τ, for any $(N, A) \in \mathcal{N} \times \mathcal{A}$. We say vencode *is sound if $\mathcal{N}$ and $\mathcal{A}$ are clear from the context.*

The vencode allows a flexible choice on ν and α. When AD can be of variable length ($\mathcal{A} = \{0,1\}^{\leq \alpha_{\max}}$), $\alpha_{\max} < 2n - \nu$ must hold as for the injectivity of padding, and when AD is fixed to $\alpha_{\max}$ bits, $\alpha_{\max} \leq 2n - \nu$ must hold. The existence of a sound vector encoding depends on the nonce and AD spaces. For example, when $\nu = n$ and $\mathcal{A} = \{0,1\}^{\alpha_{\max}}$ with some $\tau \leq \alpha_{\max} \leq 2n - \nu$ (i.e. AD length is fixed to $\alpha_{\max}$ bits), a simple encoding of $\mathsf{vencode}(N, A) = (V[1], V[2]) = (N, A)$ is sound. Another example is that $\nu \leq n$ and $\mathcal{A} = \{0,1\}^{\leq \alpha_{\max}}$ for some $0 \leq \alpha_{\max} < 2n - \nu$. In this case, a slightly more complex encoding works as $\overline{A} = \mathtt{pad}_s(A)$ for $s = \max\{n - \nu + \tau, \alpha_{\max}\}$ and $V[1] = N \,\|\, \mathtt{msb}_{n-\nu}(\overline{A})$ and $V[2] = \mathtt{lsb}_{s-(n-\nu)}(\overline{A})$. More complex cases might occur in practice, say $\mathcal{A}$ consisting of noncontiguous lengths (e.g., 2 or 4 bytes), but designing efficient vencode for such cases is beyond our scope.

Description of Manx1. The algorithms of Manx1 are as follows. For encryption, we first encode (N, A) via a sound encoding vencode to obtain $(V[1], V[2])$. We encrypt $(V[2] \,\|\, \mathtt{pad}_{n-v_2}(M))$ by XEX mode using $V[1]$ as a tweak to obtain $C \in \{0,1\}^n$, where $v_2 = |V[2]|$ is a fixed value (Def. 2). The tuple (A, X) for $X = N \,\|\, C$ is sent to the receiver. The decryption is done by checking the correctness of $V[2]$. See Fig. 1 for the pseudocode. Note that the multiplication by 2 (the generator of the field, $\mathtt{x}$) applied to S is needed for security [29,32]. For any input (N, A, M), it must be ensured (at the protocol level) that $|M| < n - v_2$ where $v_2 = |V[2]|$ and $(V[1], V[2]) = \mathsf{vencode}(N, A)$. We assume vencode is sound (Def. 2) and fixed in advance. The scheme is pretty simple while introducing vencode allows more flexible choices for the possible parameter choices.

3.5 Security of Manx1

We present the security bounds for Manx1.

Theorem 1. *Let* A_1 *be a* (q_e, t)*-priv adversary and let* A_2 *be a* (q_e, q_d, t)*-auth adversary against* Manx1 *using a block cipher* $E : \mathcal{K} \times \mathcal{M} \to \mathcal{M}$ *for* $\mathcal{M} = \{0,1\}^n$*. Then, assuming a sound vector encoding* vencode *and* $q_e \le 2^{n-1}$ *for* A_2*, we have*

$$\mathbf{Adv}^{\mathsf{priv}}_{\mathsf{Manx1}[E]}(\mathsf{A}_1) \le \mathbf{Adv}^{\mathsf{prp}}_{E}(\mathsf{A}'_1) + \frac{5q_e^2}{2^n}$$
$$\mathbf{Adv}^{\mathsf{auth}}_{\mathsf{Manx1}[E]}(\mathsf{A}_2) \le \mathbf{Adv}^{\mathsf{sprp}}_{E}(\mathsf{A}'_2) + \frac{4.5(q_e+q_d)^2}{2^n} + \frac{2q_d}{2^\tau}$$

for some A'_1 *using* q_e *encryption queries with* $t + O(q_e)$ *time, and some* A'_2 *using* $2q_e$ *encryption and* $2q_d$ *decryption queries with* $t + O(2q_e + 2q_d)$ *time.*

Proof. We derive the bounds for $\mathbf{Adv}^{\mathsf{priv}}_{\mathsf{Manx1}[\mathsf{P}]}(\mathsf{A}_1^*)$ and $\mathbf{Adv}^{\mathsf{auth}}_{\mathsf{Manx1}[\mathsf{P}]}(\mathsf{A}_2^*)$ for n-bit URP P against (q_e, ∞)-priv adversary A_1^* and (q_e, q_d, ∞)-auth adversary A_2^*. Using TURP $\widetilde{\mathsf{P}} : \mathcal{T}_w \times \mathcal{M} \to \mathcal{M}$ with $\mathcal{T}_w = \{0,1\}^n$, we define an idealized version of Manx1, i-Manx1: its encryption returns $C = \widetilde{\mathsf{P}}(V[1], (V[2] \,\|\, \overline{M}))$. The decryption is defined similarly. Then, from TSPRP advantage of XEX [29, Corollary 1], we have

$$\mathbf{Adv}^{\mathsf{priv}}_{\mathsf{Manx1}[\mathsf{P}]}(\mathsf{A}_1^*) \le \mathbf{Adv}^{\mathsf{priv}}_{\mathsf{i\text{-}Manx1}[\widetilde{\mathsf{P}}]}(\mathsf{A}_1^*) + \frac{4.5q_e^2}{2^n} \tag{1}$$
$$\mathbf{Adv}^{\mathsf{auth}}_{\mathsf{Manx1}[\mathsf{P}]}(\mathsf{A}_2^*) \le \mathbf{Adv}^{\mathsf{auth}}_{\mathsf{i\text{-}Manx1}[\widetilde{\mathsf{P}}]}(\mathsf{A}_2^*) + \frac{4.5(q_e+q_d)^2}{2^n} \tag{2}$$

We observe that N and A in any (A, X) are uniquely determined as ν is fixed. Thanks to the soundness of vencode (Def. 2), the tuple (N, A) effectively works as a nonce, that is, the tuple $(V[1], V[2])$ never repeats in encryption queries, and the correct nonce and AD are always retrieved. For the privacy notion, the first term of the right-hand side of (1) is at most $q_e^2/2^{n+1}$ which is achieved when $V[1]$ is entirely determined by AD and thus can be fixed (i.e., $V[2]$ contains the entire nonce). This proves the first (privacy) claim of the theorem. For the authenticity claim, we first consider the case $q_d = 1$ for the first term of the right-hand side of (2). A simple analysis shows that this is at most $1/(2^{v_2} - q_e) \le 1/(2^\tau - q_e)$ since $v_2 \ge \tau$ from Def. 2. To see this, let $(A', X' = N' \,\|\, C')$ be the decryption query and let $(V'[1], V'[2]) = \mathsf{vencode}(N', A')$. The worst case is achieved when, again, $V[1]$ is fixed for all encryption queries[7] and N' is used in an encryption query. The soundness of vencode guarantees that the "target" $v_2(\ge \tau)$-bit value obtained by decrypting C' with tweak $V'[1]$ must be matched with $V'[2]$. Hence, the first term of the right hand side of (2) is at most $2^{n-\tau}/(2^n - q_e) \le 2/2^\tau$ from $q_e \le 2^{n-1}$. Note that the case where depad takes the all-zero string (hence not correctly decrypting) only occurs if the forgery is successful. Applying the standard technique from single to multiple decryption queries [12], we obtain

[7] This can happen e.g. $\mathsf{vencode}(N, A) = (A, N)$ with $|A|$ fixed to n.

Algorithm Manx2.$\mathcal{E}[E_K](N, A, M)$

$\overline{A} \leftarrow \mathtt{encode}(A)$
$\alpha^* \leftarrow |\overline{A}|$
$r \leftarrow n - (\nu + \alpha^* + 2)$
if $|M| < r$ **then** //tiny message
  $C \leftarrow E_K(N \,\|\, 10 \,\|\, \overline{A} \,\|\, \mathtt{pad}_r(M))$
else if $|M| = r$ **then** //tiny message
  $C \leftarrow E_K(N \,\|\, 11 \,\|\, \overline{A} \,\|\, M)$
else //short message
  $(M[1], M[2]) \xleftarrow{r,|M|-r} (M)$
  $r' \leftarrow n - (\nu + 2)$
  $C[1] \leftarrow E_K(N \,\|\, 00 \,\|\, \overline{A} \,\|\, M[1])$
  $C[2] \leftarrow E_K(N \,\|\, 01 \,\|\, \mathtt{pad}_{r'}(M[2]))$
  $C \leftarrow C[1] \,\|\, C[2]$
return C

Algorithm Manx2.$\mathcal{D}[E_K](A, X)$

$\overline{A} \leftarrow \mathtt{encode}(A)$
$\alpha^* \leftarrow |\overline{A}|$
if $|X| = \nu + n$ **then** //tiny message
  $r \leftarrow n - (\nu + \alpha^* + 2)$
  $(N, C) \xleftarrow{\nu,n} X$
  $S \leftarrow E_K^{-1}(C)$
  $(\widetilde{N}, \widetilde{b}, \widetilde{A}, \widetilde{M}) \xleftarrow{\nu,2,\alpha^*,r} S$
  if $(\widetilde{N}, \widetilde{b}, \widetilde{A}) = (N, 10, \overline{A})$ **then**
    $M \leftarrow \mathtt{depad}_r(\widetilde{M})$
    return M
  else if $(\widetilde{N}, \widetilde{b}, \widetilde{A}) = (N, 11, \overline{A})$ **then**
    $M \leftarrow \widetilde{M}$
    return M
  else
    return $\perp$
else if $|X| = 2n$ //short message
  $r' \leftarrow n - (\nu + 2)$
  $(C[1], C[2]) \xleftarrow{n,n} X$
  $S[1] \leftarrow E_K^{-1}(C[1])$
  $S[2] \leftarrow E_K^{-1}(C[2])$
  $(\widetilde{N}[1], \widetilde{b}[1], \widetilde{A}, \widetilde{M}[1]) \xleftarrow{\nu,2,\alpha^*,r} S[1]$
  $(\widetilde{N}[2], \widetilde{b}[2], \widetilde{M}[2]) \xleftarrow{\nu,2,r'} S[2]$
  if $(\widetilde{N}[1] \neq \widetilde{N}[2])$ **or** $(\widetilde{b}[1], \widetilde{b}[2]) \neq (00, 01)$ **or** $\widetilde{A} \neq \overline{A}$ **then**
    return $\perp$
  else
    $M \leftarrow \widetilde{M}[1] \,\|\, \mathtt{depad}_{r'}(\widetilde{M}[2])$
    return M
else //unsupported length
  return $\perp$

Fig. 2: The algorithms of Manx2. The sender transmits (A, X) via the channel, where $X = (N \,\|\, C)$ when $|M| \leq n - (\nu + \alpha^* + 2)$ and $X = C$ otherwise, where $\alpha^* = |\mathtt{encode}(A)|$ for an injective `encode` function over $\mathcal{A}$. For encryption to work, we must ensure that $|M| < 2n - 2\nu - 4 - \alpha^*$ for any AD $A \in \mathcal{A}$ in advance.

$2q_d/2^\tau$ for the general case of $q_d \geq 1$. This proves the authenticity bound of (2). To conclude the proof, the final step is to obtain the computational counterparts, which is standard [11]. □

3.6 Limitations of Manx1 and our solution, Manx2

Manx1 is pretty simple. However, it incurs several drawbacks. Most importantly, the message length ℓ is at most $(n - \tau - 1)$ no matter how AD is short, and it needs two calls irrespective of ℓ. As τ cannot be arbitrarily small (otherwise, the scheme effectively reduces to unauthenticated encryption), we cannot employ Manx1 in case $\ell \approx n$. Moreover, the two calls are not parallelizable.

We present an alternative scheme that solves these problems, which we call Manx2. It accepts the message length ℓ about $2n - 2\nu - \alpha_{\max}$, and needs just one

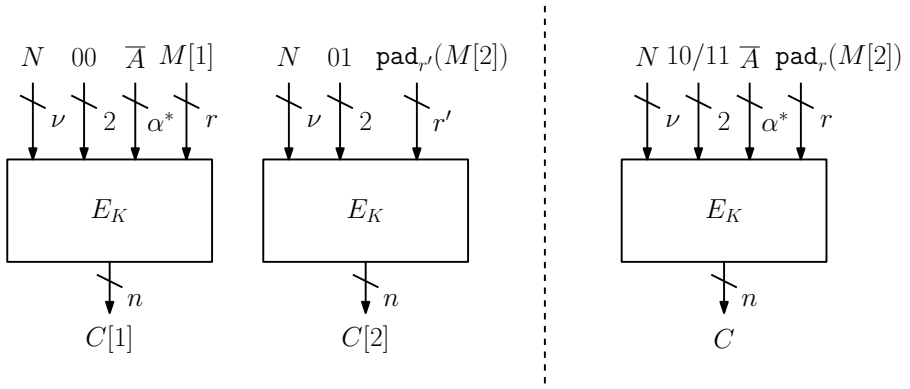

Fig. 3: Encryption of Manx2. (Left) Short message case, (Right) Tiny message case.

call when ℓ is smaller than about $n - (\nu + \alpha_{\max})$, and two calls otherwise. For convention, we call the former and the latter cases *tiny message case* and *short message case*, respectively (see Figs. 2 and 3). For simplicity, we assume $\alpha_{\max}$ is at most about $n - \nu$ because longer ADs are already supported by Manx1. The exact limits of $\alpha_{\max}$ and ℓ depend on the internal encoding of A (see below). See Fig. 2 for the algorithms of Manx2.

For example, when $\nu = n/2$, Manx2 enables encrypting a plaintext of about n bits, which was impossible with Manx1. Interestingly, Manx2 has some similarities to RPC mode by Katz and Yung [22], which was one of the earliest designs of AE and has been largely overlooked since the proposal. Unfortunately, RPC fails to meet our goal: it needs $\lceil \ell/(n-\nu) \rceil + 2$ calls for any $\ell > 0$, hence 4 calls when $\nu = n/2$ and $\ell = n$. Moreover, there is no mechanism to absorb AD.

Moreover, Manx2 has smaller bandwidth than RPC. Assuming AD is absent, the output bandwidth of RPC is $\nu + n \cdot (\lceil \ell/(n-\nu) \rceil + 2)$ bits, hence $\nu + 3n$ bits for the tiny message case, and $\nu + 4n$ bits for the short message case. In contrast, Manx2 has output bandwidth $\nu + \alpha + n$ bits for the tiny message case and $\nu + \alpha + 2n$ bits for the short message case. As a result, Manx2 saves $2n$ bits in both cases, which is non-negligible. In practice, saving bandwidth is important for IoT use cases from the power consumption perspective.

Fig. 4 shows the achievable parameter areas of (ν, ℓ) for Manx1 and Manx2, assuming (for simplicity) $\alpha_{\max} = 0$ and $\tau = n/2$. We remark that τ is the minimum authenticity level we accept. As we mentioned, Manx1 allows very long nonce; however, the message length ℓ must be significantly smaller than n, and Manx2 enables to extend ℓ close to n. Note that $\nu > n$ is not very common when $n = 128$ (thus AES), and too small ν also severely limits usability. Hence, this figure highlights the practical usefulness of Manx2 over Manx1 when the nonce has a reasonable length.

Description. Manx2 for the tiny message case is similar to the improved version of EtE described at Sect. 3.2; it encrypts as $C = E_K(N, A, M)$ and sends (A, X) to the receiver where $X = N \,\|\, C$ The decryption routine verifies the tuple $(A, X = N \,\|\, C)$ by checking if N is correctly recovered from $\mathtt{msb}_\nu(E_K^{-1}(C))$. It also checks the domain separation bits to recover M correctly. For the short message case, Manx2 first parses M into two parts, $M[1]$ and $M[2]$, where $|M| \approx n - \nu - \alpha$ and $|M[2]| \approx n - \nu$, and encrypts as $C[1] = E_K(N, A, M[1])$ and $C[2] = E_K(N, M[2])$, and sends (A, X) where $X = C[1] \,\|\, C[2]$. The decryption of a tiny message case is similar to EtE decryption, while in the case of a short message, we verify the ciphertext by comparing the first ν bits of $E_K^{-1}(C[1])$ with $E_K^{-1}(C[2])$. Note that this is an intuitive description. The exact algorithms are shown in Fig. 2. Also, Fig. 3 depicts the encryption. It turns out that the algorithms have to incorporate domain separations and an encoding function for AD to make it secure, keeping efficiency. For example, we define the encoding function $\mathtt{encode} : \mathcal{A} \to \{0,1\}^*$ that is injective with respective to $\mathcal{A}$ as in the same manner to vencode for Manx1. Such encode function can be realized by $\mathtt{encode}(A) = \mathtt{pad}_{\alpha_{\max}+1}(A)$ when $\mathcal{A} = \{0,1\}^{\leq \alpha_{\max}}$ or $\mathtt{encode}(A) = A$ when $\mathcal{A} = \{0,1\}^{\alpha_{\max}}$. The former allows $\alpha_{\max} < n - \nu - 2$, and the latter allows $\alpha_{\max} \leq n - \nu - 2$. The encryption can accept a message of length ℓ as long as $\ell < 2n - 2\nu - 4 - |\mathtt{encode}(A)|$ for any $A \in \mathcal{A}$. Note that these conditions are determined by fixing $\mathcal{M}$, $\mathcal{A}$, and encode, thus cannot be manipulated by the adversary.

In Manx2, the first block cipher call takes $\mathtt{encode}(A)$ instead of plain A, as otherwise, a simple authenticity attack would be possible when AD has a variable length. Moreover, we optimize the design to maximize the input space and minimize the bandwidth. Specifically, we utilize the 2-bit domain separation for separating the tiny and short message cases. At the same time, these 2 bits are also used to extend the possible message length of the tiny message case by a bit (lines 5 and 8 of the left part of Fig. 2). We do not explicitly send N for the short message case to reduce the bandwidth consumption by ν bits (see also the caption of Fig. 2).

3.7 Security of Manx2

We present the security bounds of Manx2. For the tiny message case, the proof basically follows EtE, while for the short message case, the way it guarantees security (in particular authenticity) is somewhat unusual. The security proof is rather intuitive; however, some careful analysis is needed, mainly due to the complexity around unifying the tiny and short message cases without explicit authentication of input lengths.

Theorem 2. *Let* A_1 *be a* (q_e, t)*-priv adversary and let* A_2 *be a* (q_e, q_d, t)*-auth adversary against* Manx2*. We assume the* encode *function is injective. Then,*

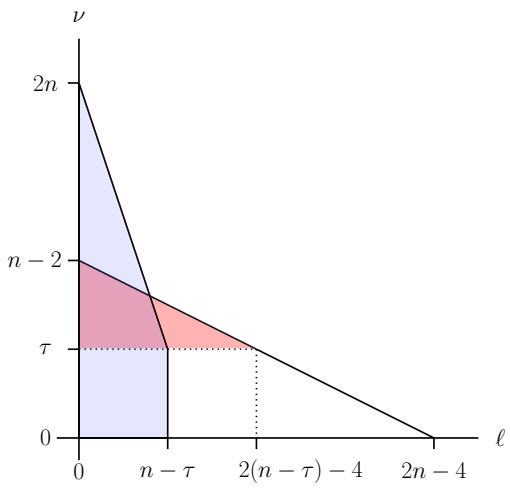

Fig. 4: Achievable parameter areas of (ν, ℓ) for Manx1 (blue) and Manx2 (red) when AD is empty ($\alpha = 0$) and $\tau = n/2$.

assuming $q_e, q_d \le 2^{n-1}$ for A_2, we have

$$\mathbf{Adv}^{\mathsf{priv}}_{\mathsf{Manx2}[E]}(\mathsf{A}_1) \le \mathbf{Adv}^{\mathsf{prp}}_{E}(\mathsf{A}'_1) + \frac{2q_e^2}{2^n}$$
$$\mathbf{Adv}^{\mathsf{auth}}_{\mathsf{Manx2}[E]}(\mathsf{A}_2) \le \mathbf{Adv}^{\mathsf{sprp}}_{E}(\mathsf{A}'_2) + \frac{2q_d}{2^\nu}$$

for some A'_1 using q_e encryption queries with $t + O(q_e)$ time, and some A'_2 using $2q_e$ encryption and $2q_d$ decryption queries with $t + O(2q_e + 2q_d)$ time.

Theorem 2 tells that, by setting $\nu \ge \tau$, our security goal ($n/2$-bit privacy and τ-bit authenticity) is achieved.

Proof. We consider the idealized version, Manx2[P], that uses an n-bit URP P instead of a block cipher. We prove

$$\mathbf{Adv}^{\mathsf{priv}}_{\mathsf{Manx2}[\mathsf{P}]}(\mathsf{A}^*_1) \le \frac{2q_e^2}{2^n}, \tag{3}$$
$$\mathbf{Adv}^{\mathsf{auth}}_{\mathsf{Manx2}[\mathsf{P}]}(\mathsf{A}^*_2) \le \frac{2q_d}{2^\nu}. \tag{4}$$

Let $q_e^{(1)}$ ($q_e^{(2)}$) be the number of encryption queries of the short (tiny) message case. Here, $q_e = q_e^{(1)} + q_e^{(2)}$ holds. The privacy claim of (3) is straightforward: as we have a nonce in every P call and all the block inputs in the game are unique thanks to the domain separation $b \in \{0,1\}^2$. Eq. (3) holds from the hybrid argument involving the PRP-PRF switching lemma, which adds at most $(2q_e^{(1)})^2/2^{n+1} = 2(q_e^{(1)})^2/2^n \le 2q_e^2/2^n$ to the bound. Note that the privacy notion requires the pseudorandomness of the output of the encryption routine, i.e.,

$C \in \{0,1\}^n \bigcup \{0,1\}^{2n}$, and not that of X (which will contain N in case of the tiny message). This is not a problem as the privacy notion does not require hiding the message length or nonce.

To prove the authenticity claims of (4), as in the case of Manx1, we start with the case $q_d = 1$ and assume the adversary makes the decryption query after q_e encryption queries, which is optimal. Let $\Theta_e = \{(N^{(i)}, A^{(i)}, M^{(i)}, C^{(i)}) \mid i \in [q_e]\}$ be the encryption transcript, where $(N^{(i)}, A^{(i)}, M^{(i)})$ and $C^{(i)}$ denote the i-th encryption query and its response. Let $Q_1 \subseteq [q_e]$ be the index sets for the encryption queries of short message case, and let $Q_2 = [q_e] \setminus Q_1$ be those for tiny message case, where $|Q_1| = q_e^{(1)}$ and $|Q_2| = q_e^{(2)}$. For convenience, we may write $(\widetilde{N}^{(i)}, \widetilde{A}^{(i)}, \widetilde{M}^{(i)}, \widetilde{C}^{(i)})$ when $i \in Q_2$. Let $\Theta_e^1 = \{(N^{(i)}, A^{(i)}, M^{(i)}, C^{(i)}) \mid i \in Q_1\}$ and $\Theta_e^2 = \{(\widetilde{N}^{(i)}, \widetilde{A}^{(i)}, \widetilde{M}^{(i)}, \widetilde{C}^{(i)}) \mid i \in Q_2\}$. Note that $\Theta_e = \Theta_e^1 \cup \Theta_e^2$. For any $i \in Q_1$, $|C^{(i)}| = 2n$, and for any $j \in Q_2$, $|\widetilde{C}^{(j)}| = n$. We write $C^{(i)}[1] = \mathtt{msb}_n(C^{(i)})$ and $C^{(i)}[2] = \mathtt{lsb}_n(C^{(i)})$. Observe that, thanks to the domain separation and nonce, all ciphertext blocks in Θ_e are distinct. That is, the three sets, $\mathcal{C}_k \coloneqq \{C^{(i)}[k] \mid i \in Q_k\}$ for $k = 1$ and $k = 2$, and $\widetilde{\mathcal{C}} \coloneqq \{\widetilde{C}^{(j)} \mid j \in Q_2\}$, have no intersections and each set has no repeating elements. We use $\mathcal{C}$ to denote $\mathcal{C}_1 \cup \mathcal{C}_2 \cup \widetilde{\mathcal{C}}$.

Let (A', X') be the decryption query. We first consider when the decryption query falls into the short message case, i.e., $|X'| = 2n$. We write C' for X' and let $\mathtt{msb}_n(C') = C'[1]$ and $\mathtt{lsb}_n(C') = C'[2]$. Let $S'[1] = \mathsf{P}^{-1}(C'[1])$ and $S'[2] = \mathsf{P}^{-1}(C'[2])$. Following the pseudocode, we define

$$(\widetilde{N}'[1], \widetilde{b}'[1], \widetilde{A}', \widetilde{M}'[1]) \xleftarrow{\nu,2,\alpha^*,r} S'[1]$$

for $\alpha^* = |\overline{A}'|$ where $\overline{A}' = \mathtt{encode}(A')$, and

$$(\widetilde{N}'[2], \widetilde{b}'[2], \widetilde{M}'[2]) \xleftarrow{\nu,2,n-(\nu+2)} S'[2].$$

When $C'[1] = C'[2]$, it means $\widetilde{b}'[1] = \widetilde{b}'[2]$, hence it never successes in forgery. So we assume $C'[1] \neq C'[2]$. Let p_{f} be the probability of successful forgery, i.e., the probability of receiving $\neq \perp$ from the decryption oracle. We provide a case analysis.

- **Case 1-1.** If $\exists i \in Q_1$ and $C' = C^{(i)}$, we have $A' \neq A^{(i)}$. From the injectiveness of $\mathtt{encode}$, $\overline{A^{(i)}} \neq \widetilde{A}'$ holds thus $p_{\mathrm{f}} = 0$.
- **Case 1-2.** If $C' \neq C^{(i)}$ for all $i \in Q_1$, we have further sub-cases. If $C'_1 \notin \mathcal{C}$, $\mathsf{P}^{-1}(C'_1)$ is uniform over a set of size $(2^n - q_e)$, thus $\Pr[\widetilde{N}[1] = \mathtt{msb}_\nu(C'_2)]$ is at most $2^{n-\nu}/(2^n - q_e) \leq 2/2^\nu$ by the assumption.
- **Case 1-3.** If $C'[1] \in \mathcal{C}_2 \cup \widetilde{\mathcal{C}}$, it holds that $\widetilde{b}'[1] \neq 00$, hence $p_{\mathrm{f}} = 0$.
- **Case 1-4.** If $C'[1] = C^{(i)}[1]$ for some $i \in Q_1$, we have $C'[2] \neq C^{(i)}[2]$. We have sub-cases: (1) if $C'[2] \notin \mathcal{C}$ then $\mathsf{P}^{-1}(C'[2])$ is uniform over a set of size $(2^n - q_e)$ and thus $p_{\mathrm{f}} \leq 2/2^\nu$ as in **Case 1-2**. The remaining cases are (2) $C'[2] \in \mathcal{C}_1$ and (3) $C'[2] = C^{(h)}[2]$ for some $h \in Q_1$, $h \neq i$, and (4) $C'[2] \in \widetilde{\mathcal{C}}$. Any sub-cases have $p_{\mathrm{f}} = 0$ due to the domain separation or a difference in the decrypted nonce.

We consider the tiny message case, i.e., $|X'| = n + \nu$. Let C' be $\mathtt{lsb}_n(X')$ and N' be $\mathtt{msb}_\nu(X')$.

- **Case 2-1.** Suppose $C' = \widetilde{C}^{(j)}$ for some $j \in Q_2$. We have either $A' \neq \widetilde{A}^{(j)}$ or $N' \neq \widetilde{N}^{(j)}$, hence $p_{\mathrm{f}} = 0$.
- **Case 2-2.** If $C' \in \mathcal{C}_1 \cup \mathcal{C}_2$, the domain separation bits guarantee $p_{\mathrm{f}} = 0$.
- **Case 2-3.** If $C' \notin \mathcal{C}$, $\mathsf{P}^{-1}(C')$ is uniform over a set of size $(2^n - q_e)$, hence the probability $\Pr[\mathtt{msb}_\nu(\mathsf{P}^{-1}(C')) = N']$ is at most $2/2^\nu$ as in **Case 1-2**.

Overall, when $q_d = 1$, we have $p_{\mathrm{f}} \leq 2/2^\nu$. Combining with the standard technique by Bellare et al. [12], we prove the authenticity bound of (4). The derivation of the computational counterpart is also standard [11]. This concludes the proof. □

4 Implementations

This section reports software implementation results of Manx in order to measure its benefits over existing modes when processing short inputs. Since Manx aims to be deployed on embedded devices, we run benchmarks on 8-bit and 32-bit microprocessors for several parameters sets using AES-128 as the underlying block cipher. Our Manx implementations are publicly available at `www.github.com/aadomn/manx_ae`.

4.1 Benchmark settings

Platforms. We consider two popular microprocessors for the IoT: the 8-bit AVR ATmega128 and the 32-bit ARM Cortex-M4. For benchmarks on ATmega128, we used Microchip Studio `7.0.2594` in debugging mode with `avr-gcc 12.1.0`. For benchmarks on ARM Cortex-M4, we used an STM32F407VG microcontroller with `arm-none-eabi-gcc 10.3.1`. Both environments allow us to accurately measure the number of clock cycles required to complete the encryption process.

AES implementations. For both platforms, we consider the fastest constant-time AES implementations that are publicly available. On AVR, we use the RijndaelFast variant from [30] which requires around 2.4K clock cycles to encrypt a 128-bit block (using pre-computed round keys) and around 800 cycles to run the key schedule. It implements the S-box using a look-up table which is considered safe against timing attacks since AVR microcontrollers do not embed any cache memory. On ARM Cortex-M4 we use the fixsliced implementation from Adomnicai and Peyrin [6] which currently constitutes the fastest constant-time AES implementation on this platform. It requires around 2.8K cycles to encrypt two blocks at a time (with pre-computed round keys) and around 1.5K cycles to run the key schedule. However its performance are reduced by a factor of 2 when combined with a sequential mode of operation since the second block is computed for nothing (it can actually be used for side-channel countermeasures if needed). Therefore, on top of providing performance insights on both 8-bit

and 32-bit architectures, our benchmark also highlights the discrepancies that may arise when using a serial versus a parallel implementation of the underlying block cipher.

Reference modes. As reference, we consider the following four AE modes of operation: GCM, CCM, OCB and COFB. All modes are implemented in C while the AES implementations mentioned above are both written in assembly. For the hash function GHASH in GCM, we use the 32-bit constant-time implementation from BearSSL[8]. For each mode, the round key material is calculated only once.

Parameter sets. Our benchmark consider the following three parameter sets for (ν, α, ℓ) to cover different cases. The first case is $(64, 0, 120)$ for the largest input message (in terms of nibbles) that can be handled by Manx2, keeping the capability of 2^{64} messages for its 64-bit security. The second case $(96, 0, 56)$ follows the same motivation, but with $\nu = 96$ to avoid two additional GHASH calls in GCM, which is the common choice for this mode. The third case $(64, 16, 44)$ considers tiny messages with tiny associated data.

4.2 Results

As detailed in Table 1, our benchmark shows that the Manx family of AE modes outperforms all other reference modes, for all parameters sets on both platforms.

8-bit AVR. On ATmega128, when considering tiny messages with associated data, Manx2 runs around 240% faster than COFB, which is the fastest option among the reference modes. However, the improvement is less pronounced for short messages mainly because we are only saving a single call to AES-128 instead of two. Also, when $\nu \bmod 8 = 0$, Manx2 requires many bitshifts to concatenate $\overline{A}$ and M into the input blocks $N \,\|\, 00 \,\|\, \overline{A} \,\|\, M[1]$ and $N \,\|\, 01 \,\|\, \mathsf{pad}_{r'}(M[2])$ since the 2-bit domain separator introduces a misalignment (i.e. the block is not byte-aligned anymore). Since the shift instruction on AVR can only shift by a single bit a time, this can result in a non-negligible overhead in terms of performance. Note that when ν and α are fixed at the protocol level, the amount of bits to shift is known in advance and the corresponding code can optimized using dedicated assembly routines [26]. For instance, by fixing ν and α such that $\nu \bmod 8 = \alpha \bmod 8 = 0$ and hard coding the bitshifts accordingly, Manx2 now requires 7 466 cycles instead of 8 411 for $(\nu, \alpha, \ell) = (64, 44, 16)$.

All in all, the performance gain on 8-bit AVR is close to the number of calls to the internal block cipher since the AES-128 implementation processes a single block at a time on this platform. Note that GCM is clearly not relevant on AVR because of the challenge of efficiently implementing $\mathrm{GF}(2^{128})$ multiplications due to 8-bit multiplications and single bit shift instructions. An optimized assembly implementation could definitely improve its performance, but presumably not to the extent of competing with the other modes.

[8] `https://bearssl.org/`

Parameters (bits)			**Mode**	AES-128 calls	**Speed** (clock cycles)	
ν	α	ℓ			ATmega128	Cortex-M4
64	0	120	GCM	3* (3)	147 871	13 208
			CCM	4 (2+2)	12 029	7 905
			OCB	4 (2+2)	14 933	8 371
			COFB	3 (1+1+1)	10 768	11 322
			Manx1	2 (1+1)	-	-
			Manx2	2 (2)	**8 411**	**5 379**
96	0	56	GCM	3* (3)	53 898	10 468
			CCM	4 (2+2)	11 679	7 842
			OCB	4 (2+2)	14 540	8 280
			COFB	3 (1+1+1)	10 990	10 821
			Manx1	2 (1+1)	**6 525**	7 817
			Manx2	2 (2)	7 597	**5 179**
64	16	44	GCM	3* (3)	159 912	14 551
			CCM	5 (2+2+1)	14 355	10 919
			OCB	5 (2+2+1)	17 661	11 392
			COFB	3 (1+1+1)	11 144	11 649
			Manx1	2 (1+1)	6 586	7 858
			Manx2	1	**4 643**	**5 008**

* GCM needs additional GF(2^{128}) multiplications (2 when $\nu = 96$ and 4 when $\nu = 64$)

Table 1: Benchmark on 8-bit AVR ATmega128 and 32-bit ARM Cortex-M4 microprocessors when encrypting/authenticating messages with different parameter sets. The number of calls to the internal block cipher indicates the degree of parallelism provided by the mode (e.g. 2+1 means two calls can be processed in parallel except the last one). The AES-128 implementation on 8-bit AVR processes a single block at a time while the one on 32-bit ARM processes two blocks in parallel. No results are reported for Manx1 when $(\nu, \alpha, \ell) = (64, 0, 120)$ since it cannot handle such long inputs.

32-bit ARM. On Cortex-M4, the results are now correlated to the degree of parallelism provided by the mode since the AES-128 implementation reaches its best performance when processing two blocks at once. This explains why CCM and OCB are faster than COFB on this platform: although it requires more calls to the internal block cipher, they provide the ability to process blocks in parallel while COFB is fully sequential. When omitting associated data, Manx2 runs approximately 30% faster than CCM, which is the fastest option among

the reference modes. However, CCM requires an additional call to AES-128 when processing associated data, which makes Manx2 around twice faster in this setting.

All things considered, the Manx family allows to reduce the overhead of AE based on software AES-128 from 30% to 240% over previous solutions on AVR ATmega128 and ARM Cortex-M4, depending on parameters sets and the degree of parallelism which can be fully exploited. Note that the gain should be even more significant when the internal primitive embeds side-channel countermeasures (e.g. masking), which may decrease its performance manyfold. While we chose AES as the standard cipher, expanding the benchmarking using other lightweight block ciphers, say GIFT [10], and comparing with NIST LwC candidates would be interesting future work.

5 Concluding remarks

We studied the problem of AE for very short messages, say smaller than the block size of the block cipher we use. Based on the observation that the known popular modes need at least 3 to 5 calls for any non-empty messages, we explored the design space for AE with up to two block cipher calls. We proposed a family of AE modes, Manx, that can handle total input space at most $2n$ bits with additional restrictions and have comparable security as existing AE modes. Notably, Manx2 is the first proposal to encrypt about n-bit plaintext using two calls and achieve comparable security to the standard AE modes. Our microprocessor benchmark showcases the significant advantages of Manx2 over the known popular modes.

By design, Manx cannot handle long messages. Hence its scope is niche. However, if we want to support long messages, it can be combined with an existing mode, say by using different keys or using domain separation by AD. For applications where message lengths are widely distributed (e.g., few bytes to few kilobytes), such a combination may improve the average speed from using a single existing mode, say GCM. A formal analysis of the security/efficiency of such a combination would be a future topic. Further design investigation to expand the achievable domain of input parameters within two calls and extend the problem to TBC/permutation-based constructions are also interesting directions.

Acknowledgements

We thank Yoshinori Aono and Takenobu Seito for the fruitful discussions.
This research was in part conducted under a contract of "Research and development on IoT malware removal / make it non-functional technologies for effective use of the radio spectrum" among "Research and Development for Expansion of Radio Wave Resources (JPJ000254)", which was supported by the Ministry of Internal Affairs and Communications, Japan. This work was in part supported by JSPS KAKENHI Grant Number JP22K19773.

References

1. Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC. NIST Special Publication 800-38D (2007), National Institute of Standards and Technology.
2. Recommendation for Block Cipher Modes of Operation: the CCM Mode for Authentication and Confidentiality. NIST Special Publication 800-38C (2007), National Institute of Standards and Technology.
3. Sigfox Technical Overview (2017), https://www.ismac-nc.net/wp/wp-content/uploads/2017/08/sigfoxtechnicaloverviewjuly2017-170802084218.pdf, accessed: 2023-01-23
4. Submission Requirements and Evaluation Criteria for the Lightweight Cryptography Standardization Process (2018), https://csrc.nist.gov/CSRC/media/Projects/Lightweight-Cryptography/documents/final-lwc-submission-requirements-august2018.pdf, accessed: 2023-01-23
5. EnOcean Serial Protocol 3 (ESP3) Specification (2020), https://www.enocean.com/wp-content/uploads/Knowledge-Base/EnOceanSerialProtocol3.pdf, accessed: 2023-01-23
6. Adomnicai, A., Peyrin, T.: Fixslicing AES-like ciphers. IACR TCHES **2021**(1), 402–425 (2021). https://doi.org/10.46586/tches.v2021.i1.402-425, https://tches.iacr.org/index.php/TCHES/article/view/8739
7. Andreeva, E., Lallemand, V., Purnal, A., Reyhanitabar, R., Roy, A., Vizár, D.: ForkAE. A submission to NIST Lightweight Cryptography (2019)
8. Andreeva, E., Lallemand, V., Purnal, A., Reyhanitabar, R., Roy, A., Vizár, D.: Forkcipher: A new primitive for authenticated encryption of very short messages. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019, Part II. LNCS, vol. 11922, pp. 153–182. Springer, Heidelberg (Dec 2019). https://doi.org/10.1007/978-3-030-34621-8_6
9. Banik, S., Chakraborti, A., Inoue, A., Iwata, T., Minematsu, K., Nandi, M., Peyrin, T., Sasaki, Y., Sim, S.M., Todo, Y.: GIFT-COFB. A submission to NIST Lightweight Cryptography (2019)
10. Banik, S., Pandey, S.K., Peyrin, T., Sasaki, Y., Sim, S.M., Todo, Y.: GIFT: A small present - towards reaching the limit of lightweight encryption. In: Fischer, W., Homma, N. (eds.) CHES 2017. LNCS, vol. 10529, pp. 321–345. Springer, Heidelberg (Sep 2017). https://doi.org/10.1007/978-3-319-66787-4_16
11. Bellare, M., Desai, A., Jokipii, E., Rogaway, P.: A Concrete Security Treatment of Symmetric Encryption. In: FOCS. pp. 394–403. IEEE Computer Society (1997)
12. Bellare, M., Goldreich, O., Mityagin, A.: The power of verification queries in message authentication and authenticated encryption. Cryptology ePrint Archive, Report 2004/309 (2004), https://eprint.iacr.org/2004/309
13. Bellare, M., Namprempre, C.: Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 531–545. Springer, Heidelberg (Dec 2000). https://doi.org/10.1007/3-540-44448-3_41
14. Bellare, M., Ng, R., Tackmann, B.: Nonces are noticed: AEAD revisited. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part I. LNCS, vol. 11692, pp. 235–265. Springer, Heidelberg (Aug 2019). https://doi.org/10.1007/978-3-030-26948-7_9
15. Bellare, M., Rogaway, P.: Encode-then-encipher encryption: How to exploit nonces or redundancy in plaintexts for efficient cryptography. In: Okamoto, T. (ed.) ASI-

ACRYPT 2000. LNCS, vol. 1976, pp. 317–330. Springer, Heidelberg (Dec 2000). https://doi.org/10.1007/3-540-44448-3_24
16. Bos, J.W., Osvik, D.A., Stefan, D.: Fast Implementations of AES on Various Platforms. Cryptology ePrint Archive, Paper 2009/501 (2009), https://eprint.iacr.org/2009/501
17. Chakraborti, A., Iwata, T., Minematsu, K., Nandi, M.: Blockcipher-based authenticated encryption: How small can we go? In: Fischer, W., Homma, N. (eds.) CHES 2017. LNCS, vol. 10529, pp. 277–298. Springer, Heidelberg (Sep 2017). https://doi.org/10.1007/978-3-319-66787-4_14
18. Dobraunig, C., Eichlseder, M., Mendel, F., Schläffer, M.: Ascon. A submission to NIST Lightweight Cryptography (2019)
19. Guo, C., Iwata, T., Khairallah, M., Minematsu, K., Peyrin, T.: Romulus. A submission to NIST Lightweight Cryptography (2019)
20. Iwata, T., Kurosawa, K.: OMAC: One-key CBC MAC. In: Johansson, T. (ed.) FSE 2003. LNCS, vol. 2887, pp. 129–153. Springer, Heidelberg (Feb 2003). https://doi.org/10.1007/978-3-540-39887-5_11
21. Iwata, T., Minematsu, K., Guo, J., Morioka, S.: CLOC: Authenticated encryption for short input. In: Cid, C., Rechberger, C. (eds.) FSE 2014. LNCS, vol. 8540, pp. 149–167. Springer, Heidelberg (Mar 2015). https://doi.org/10.1007/978-3-662-46706-0_8
22. Katz, J., Yung, M.: Unforgeable encryption and chosen ciphertext secure modes of operation. In: Schneier, B. (ed.) FSE 2000. LNCS, vol. 1978, pp. 284–299. Springer, Heidelberg (Apr 2001). https://doi.org/10.1007/3-540-44706-7_20
23. Khovratovich, D.: Key wrapping with a fixed permutation. In: Benaloh, J. (ed.) CT-RSA 2014. LNCS, vol. 8366, pp. 481–499. Springer, Heidelberg (Feb 2014). https://doi.org/10.1007/978-3-319-04852-9_25
24. Kim, Y., Seo, S.C.: Efficient Implementation of AES and CTR_DRBG on 8-Bit AVR-Based Sensor Nodes. IEEE Access **9**, 30496–30510 (2021). https://doi.org/10.1109/ACCESS.2021.3059623
25. Krovetz, T., Rogaway, P.: The software performance of authenticated-encryption modes. In: Joux, A. (ed.) FSE 2011. LNCS, vol. 6733, pp. 306–327. Springer, Heidelberg (Feb 2011). https://doi.org/10.1007/978-3-642-21702-9_18
26. van Laethem, A.: Optimizing constant bitshifts on AVR (2021), https://aykevl.nl/2021/02/avr-bitshift, accessed: 2023-01-2023
27. Liskov, M., Rivest, R.L., Wagner, D.: Tweakable block ciphers. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 31–46. Springer, Heidelberg (Aug 2002). https://doi.org/10.1007/3-540-45708-9_3
28. Malik, H., Alam, M.M., Moullec, Y.L., Kuusik, A.: NarrowBand-IoT Performance Analysis for Healthcare Applications. In: ANT/SEIT. Procedia Computer Science, vol. 130, pp. 1077–1083. Elsevier (2018)
29. Minematsu, K.: Improved security analysis of XEX and LRW modes. In: Biham, E., Youssef, A.M. (eds.) SAC 2006. LNCS, vol. 4356, pp. 96–113. Springer, Heidelberg (Aug 2007). https://doi.org/10.1007/978-3-540-74462-7_8
30. Poettering, B.: AVRAES: The AES block cipher on AVR controllers. http://point-at-infinity.org/avraes/, accessed: 2023-01-23
31. Rogaway, P.: Authenticated-encryption with associated-data. In: Atluri, V. (ed.) ACM CCS 2002. pp. 98–107. ACM Press (Nov 2002). https://doi.org/10.1145/586110.586125
32. Rogaway, P.: Efficient instantiations of tweakable blockciphers and refinements to modes OCB and PMAC. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS,

vol. 3329, pp. 16–31. Springer, Heidelberg (Dec 2004). https://doi.org/10.1007/978-3-540-30539-2_2
33. Rogaway, P., Shrimpton, T.: A provable-security treatment of the key-wrap problem. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 373–390. Springer, Heidelberg (May / Jun 2006). https://doi.org/10.1007/11761679_23