





Provably Secure Online Authenticated Encryption and Bidirectional Online Channels*

Arghya Bhattacharjee^{1,2}  Ritam Bhaumik^{1,2,3}  Daniel Collins^{3,4}  Mridul Nandi¹ 

¹ Indian Statistical Institute, Kolkata, India

² Technology Innovation Institute, Abu Dhabi, UAE

³ EPFL, Lausanne, Switzerland

⁴ Purdue University, West Lafayette, USA

bhattacharjeearghya29@gmail.com, bhaumik.ritam@gmail.com, colli594@purdue.edu,
mridul.nandi@gmail.com

Abstract

In this work, we examine online authenticated encryption with variable expansion. We follow a notion where both encryption and decryption are online, and security is ensured in the RUP (Release of Unverified Plaintext) setting. Then we propose a generic way of obtaining an online authenticated encryption mode from a tweakable online encryption mode based on the encode-then-encipher paradigm (Bellare and Rogaway, Asiacrypt 2000). To instantiate our generic scheme, we start with proposing a provably-secure tweakable online encryption mode called t-OleF, a tweakable version of OleF (Bhaumik and Nandi, ToSC 2016(2)), and then plug it into our generic scheme to obtain OI/EF, a provably-secure online authenticated encryption mode. As an application, we propose a primitive we call a bidirectional online channel suited for communication between lightweight devices.

1 Introduction

Authenticated encryption is a widely-used mode of cryptographic operation which provides both privacy and authenticity to the ciphertext. Initially, Goldwasser and Micali [GM84] formalised encryption schemes as stateful or probabilistic. Later, Rogaway [Rog02, Rog04] unified them by defining a cryptographic scheme as a deterministic algorithm that takes a unique user-supplied *nonce*. In the case of *nonce-based authenticated encryption* (nAE), the adversary is not allowed to repeat the nonce between any two encryption queries. If such repetition is observed, then the security guarantees (both privacy and authenticity) disappear. To overcome this strong requirement regarding nonce repetition, Rogaway and Shrimpton [RS06] proposed the notion of *misuse resistant authenticated encryption* (MRAE). In the MRAE game, the adversary is allowed to repeat the nonce between encryption queries. This repetition of the nonce has no adverse effect on authenticity, while privacy is compromised only to the extent that the repetition of the entire encryption query is detectable.

With all its advantages, there is one major drawback of MRAE, namely that an MRAE scheme can not be *online*. When we say an AE scheme is online, we mean that the encryption is done in a single pass over the message so that it can be realised with constant memory. In the case of an MRAE scheme, its definition implies that every bit of the ciphertext depends on every bit of the message. So it can not output the first ciphertext bit before it gets the last message bit. So, by definition, an online MRAE scheme is impossible to design [HRRV15].

*An extended abstract of this paper is included in the proceedings of SAC 2024.

Online AE. To overcome this extreme degradation of security or efficiency in either case, Fleischmann et al. [FFL12] proposed the notion which we call OAE1 (following [HRRV15]). Their definition builds on the idea of an online cipher [BBKN01]. An online cipher with block-size n is a cipher $\mathcal{E} : \mathcal{K} \times \{\{0, 1\}^n\}^* \rightarrow \{\{0, 1\}^n\}^*$ (i.e., $\mathcal{E}(K, \cdot)$ is a permutation for every $K \in \mathcal{K}$) where the i -th ciphertext block depends only on the first i message blocks; and an OAE1 output is the output of a tweakable online cipher with (N, A) as the tweak and M as the input, followed by a random (N, A, M) dependent tag, with N , A and M denoting the nonce, the associated data, and the message respectively.

Weakness of OAE1. Fleischmann et al. [FFL12] claimed that OAE1 supports MRAE security and online AE. Hoang et al. [HRRV15] showed first with a trivial attack on small-block OAE1 and then with the chosen-prefix/secret-suffix (CPSS) attack (the idea of which was introduced in a different setting with the BEAST attack [DR11]) on OAE1 with any block-size that the MRAE security claim is wrong, i.e., an adversary can exploit the nonce-misuse power to mount those attacks. Also, they made a series of arguments against their claim of constructing online AE. One of them was that even though encryption is online, decryption is not, i.e., the message cannot be released until and unless it is verified. For that, it needs additional security in the RUP (*Releasing Unverified Plaintext*) setting. Finally, in the same work, the authors proposed an alternate notion named OAE2. It is worth noting that there is a downside to the argument regarding the CPSS attack. Firstly, many real-world settings don't enable a CPSS attack. Also, OAE2 is vulnerable to the CPSS attack as well. Both of these facts are acknowledged in [HRRV15]. Later, Andreeva et al. [ABV20] extended the notion of OAE2 to deal with messages that have an incomplete last block.

RUP Setting. Security notion for AE schemes in the RUP setting was initially formalised by Andreeva et al. [ABL⁺14]. They proposed the notions IND-CPA + PA1/PA2 for privacy and INT-RUP (INT-CTXT in RUP setting) for authenticity. Among other things, the authors showed that OCB [KR11] and COPA [ABL⁺13] are not INT-RUP secure. Chang et al. [CDD⁺19] proposed the notion AERUP which unified IND-CPA + PA1 and INT-RUP. In the meantime, Ashur et al. [ADL17] also proposed an alternate notion named RUPAE which focused on nAE schemes. Unfortunately, it turned out that PA1 cannot be achieved by online AE schemes [ZW16]. PA2, being a stronger security notion than PA1, is trivially not helpful to the cause. Earlier, Abed et al. [AFF⁺14] used OPRP-CCA notion in the context of privacy of online AE schemes in RUP setting. Later, Zhang and Wu [ZW16] introduced OPA notion, and showed that OPA coupled with OPRP-CPA is equivalent to OPRP-CCA. Note that OPRP-CPA and OPRP-CCA were introduced by Bellare et al. [BBKN01] in the context of online ciphers.

Secure Channels. In practice, an encryption mode can be used to construct secure communication channels [AMPS22, BKN02, HDL21, MP17] between two or more parties. In the two-party case, Marson and Poettering [MP17] noted the importance of modelling *bidirectional* channels (where both parties send and receive) instead of unidirectional channels due to subtleties that arise when composing two unidirectional channels. Some channels consider different security properties and support different message patterns like out-of-order message delivery. Generally, the RUP setting is not considered in existing formalism: [DF18] is one notable exception, albeit they work in the Universal Composability (UC) framework [Can01].

Our Contributions.

The primary contribution of this work is two-fold: (1) a generic Encode-then-Encipher (EtE) scheme to obtain an online authenticated encryption from a tweakable online cipher, and (2) a new primitive called bidirectional online channel which is essentially an application of online authenticated encryption. Also,

to further elaborate the EtE scheme, we design a new tweakable online cipher $t\text{-OleF}$, and instantiate the EtE scheme with that to obtain $\text{OI}\mathcal{A}\mathcal{E}\mathcal{F}$. In more detail:

1. First, we discuss the notion of tweakable online cipher, and propose a secure tweakable online cipher $t\text{-OleF}$, based on the online cipher OleF [BN16].
2. Next, we discuss the notion of online authenticated encryption, and propose a generic EtE scheme [BR00] to obtain an online authenticated encryption scheme from a tweakable online cipher, which is the first primary contribution of this work. Towards this aim, we prove that a random tweakable online permutation composed with an injective suffix-pad results in a random tweakable online injection.

An important design feature of the EtE scheme is that it allows an optional use of a nonce. In the nonce-based setting, the nonce can be used as a prefix of the associated data using an injective padding. Since the nonceless setting can be seen as a nonce-misuse setting, this implies that the EtE scheme is MRAE secure up to the leakage of common prefix, which we understand is the best one can achieve in the online setting.

3. Next, we propose a concrete instantiation of the EtE scheme. We use $t\text{-OleF}$ as the underlying tweakable online cipher to obtain an online authenticated encryption scheme which we call $\text{OI}\mathcal{A}\mathcal{E}\mathcal{F}$.

$\text{OI}\mathcal{A}\mathcal{E}\mathcal{F}$ has a couple of important design features. $\text{OI}\mathcal{A}\mathcal{E}\mathcal{F}$ matches the bound for the number of block cipher calls needed for security in the non-online setting shown by Nandi [Nan15]: namely, it uses two block cipher calls per block (or more precisely, four per *diblock* i.e., a sequence of two consecutive blocks). Similar to OleF or $t\text{-OleF}$, $\text{OI}\mathcal{A}\mathcal{E}\mathcal{F}$ is block cipher based and *inverse-free*, i.e., it depends only on the encryption circuit of the underlying block cipher. This enables us to use a weak security assumption (i.e., PRF security) on the block cipher (instead of the stronger SPRP security). Also, we avoid the extra cost of implementing the inverse circuit of the block cipher.

4. Finally, we arrive at the second primary contribution of this work. As an application of online authenticated encryption, we introduce a new primitive which we call *bidirectional online channel*. We define appropriate correctness notions, in particular capturing message sending that is *online* and in batches of L blocks (such that authentication is performed for every L blocks). Our security notion captures the RUP setting, and while our primitive is stateful, ‘state resets’ are allowed by the security game, in which case we provide online security (and otherwise provide ‘full’ security). We construct a secure bidirectional online channel from an online authenticated encryption scheme.

Related Work.

In recent years, online AE with INT-RUP security (which ensures that both encryption and decryption are online) has drawn significant research interest, and a few designs have already been proposed. Some of the important names are CAESAR [CAE19] candidates POET [AFF⁺14], APE [ABB⁺14] and Minalpher [STA⁺b] (second-round candidates), NIST LWC [NIS18] candidates TinyJAMBU [WH] (a finalist), SAEAES [NMS⁺] (instantiation of SAEB [NMSS18] with AES-128), LOCUS and LOTUS [CDJ⁺19], Oribatida [BLLN21] and SAEF [ALP⁺19] (all but TinyJAMBU were second-round candidates). Other examples include Lynx [HC23]. One can look into the following works examining the INT-RUP security of different designs: [AFF⁺] for POET, [ABB⁺] for APE, [STA⁺a] for Minalpher, [DDG22] for SAEB and TinyJAMBU, and [ABV21] for SAEF.

AE with INT-RUP security has garnered independent attention as well, because there are other motivations for INT-RUP security besides making the design online. Some AE schemes have been proven to be INT-RUP secure, though they are not online, e.g., NIST LWC candidates Romulus-M [IKMP20] (one

of the finalists) and ESTATE [CDJ+20] (a second-round candidate). Other examples include MONDAE [CDD+19]. [ABV21] considers the INT-RUP security of Romulus-M and ESTATE.

Outline.

We start with notations and a few security and cryptographic notions in Section 2 that we use in the rest of the work. Section 3 talks about t-OleF and its security. Section 4 introduces the generic scheme to obtain an online authenticated encryption from a tweakable online cipher, does its security analysis, plugs in t-OleF into it to obtain OlÆF, and proposes its instantiation. Section 5 covers our bidirectional online channels and their security. Finally, Section 6 concludes.

2 Preliminaries

Notation. For a positive integer r , $[r]$ denotes the set $\{1, \dots, r\}$. For a bit-string x , $\|x\|$ denotes its length. For $t, s \in [\|x\|]$ and $t \leq s$, $x_{t..s}$ denotes the $(s - t + 1)$ -bit sub-string of x starting at the t -th bit. When $y = x_{1..s}$ for some $s \in [\|x\|]$, we say y is the s -bit prefix of x ; if in addition $s < \|x\|$, we say y is a *proper* prefix of x . For two strings x and y , $x\|y$ denotes their concatenation. For a finite set S , its size is denoted by $|S|$. For two sets S_1 and S_2 of strings, we define their cardinal product as $S_1 \times S_2 := \{x\|y \mid x \in S_1, y \in S_2\}$. A set S of strings is called *prefix-free* if there does not exist a pair of strings in S such that one is a prefix of the other. For a positive integer m , let $\{0, 1\}^{\leq m} := \bigcup_{t=1}^m \{0, 1\}^t$ denote the set of bit-strings with at most m bits.

For a non-negative integer τ , a function $f : \mathcal{D} \rightarrow \mathcal{R}$ is said to be τ -*expanding* if for every $x \in \mathcal{D}$, $\|f(x)\| = \|x\| + \tau$; and τ is called the expansion of f . f is called length-preserving if it is 0-expanding. An injection $f : \mathcal{D} \rightarrow \mathcal{R}$ is called prefix-free if the set $\{f(x) \mid x \in \mathcal{D}\}$ is prefix-free. An injection from a domain \mathcal{D} onto itself is called a permutation over \mathcal{D} . Given a set S and a distribution μ over it, we write $x \leftarrow_{\mu} S$ to denote that x is sampled from S according to distribution μ . When μ is the uniform distribution over S , we simply write $x \leftarrow_{\mathcal{S}} S$.

2.1 Security Notions

Distinguishing Game. For two oracles \mathcal{O}_0 and \mathcal{O}_1 , let \mathcal{A} be an algorithm which tries to distinguish between \mathcal{O}_0 and \mathcal{O}_1 . A *distinguishing game* is an interactive game that \mathcal{A} plays with \mathcal{O}_b ($b \in \{0, 1\}$) where b is unknown to \mathcal{A} , and then outputs a guess for b . \mathcal{A} wins when the guessed bit matches b . A *strong distinguishing game* is a special kind of distinguishing game where \mathcal{O}_b can handle both forward and inverse queries by accepting an extra bit to indicate the type of the query. \mathcal{A} is called a (*strong*) *distinguishing adversary*. Sometimes, we say \mathcal{G} -adversary to denote that \mathcal{A} plays the (strong) distinguishing game \mathcal{G} .

Distinguishing Advantage. We use the notation $\Pr_{\mathcal{O}_b}[\cdot]$ to denote probability under oracle \mathcal{O}_b . For an adversary \mathcal{A} playing a (strong) distinguishing game with \mathcal{O}_b ($b \in \{0, 1\}$), we define the (*strong*) *distinguishing advantage* of \mathcal{A} as

$$\mathbf{Adv}_{\mathcal{O}_1, \mathcal{O}_0}(\mathcal{A}) := \left| \Pr_{\mathcal{O}_0}[\mathcal{A} \Rightarrow 1] - \Pr_{\mathcal{O}_1}[\mathcal{A} \Rightarrow 1] \right|,$$

where $\mathcal{A} \Rightarrow b$ denotes the event that \mathcal{A} outputs b . Sometimes, we say \mathcal{G} -advantage or write $\mathbf{Adv}_{\mathcal{O}_1, \mathcal{O}_0}^{\mathcal{G}}(\mathcal{A})$ to denote that \mathcal{A} plays the (strong) distinguishing game \mathcal{G} . Sometimes we just write $\mathbf{Adv}_{\mathcal{O}_1}^{\mathcal{G}}(\mathcal{A})$ or $\mathbf{Adv}_{\mathcal{O}_1}(\mathcal{A})$ when \mathcal{O}_0 is clear from the context. If the (strong) distinguishing advantage is upper-bounded by ϵ for any adversary, we say that \mathcal{O}_0 and \mathcal{O}_1 are $(1 - \epsilon)$ -*(strong) indistinguishable*. Sometimes we simply say that \mathcal{O}_0 and \mathcal{O}_1 are (strong) indistinguishable, when ϵ is “negligible”.

Pseudorandomness. Consider a class \mathcal{C} of functions from \mathcal{D} to \mathcal{R} , and a key space \mathcal{K} . Consider the keyed family of functions

$$\mathcal{F} = \{f_k \mid k \in \mathcal{K}\} \subset \mathcal{C}.$$

We say \mathcal{F} is pseudorandom (or strong pseudorandom) in \mathcal{C} if for $f \leftarrow_{\S} \mathcal{C}$ and $k \leftarrow_{\S} \mathcal{K}$, oracles \mathcal{O}_0 simulating f (called the ideal oracle) and \mathcal{O}_1 simulating f_k (called the real oracle) are indistinguishable (or strong indistinguishable respectively). Common examples of (strong) pseudorandom families are pseudorandom function, pseudorandom permutation, and strong pseudorandom permutation, which we discuss below.

Security notions: prf, prp and sprp. Suppose \mathcal{F} is pseudorandom in \mathcal{C} . When \mathcal{F} is pseudorandom in Func , the set of all functions from \mathcal{D} to \mathcal{R} , it is called a *pseudorandom function* (prf) from \mathcal{D} to \mathcal{R} . When \mathcal{F} is pseudorandom in Perm_m , the set of all permutations from $\{0, 1\}^m$ to $\{0, 1\}^m$, \mathcal{F} is called an *m-bit pseudorandom permutation* (prp). Finally, when \mathcal{F} is strong pseudorandom in Perm_m , it is called an *m-bit strong pseudorandom permutation* (sprp).

2.2 Cryptographic Notions

Deterministic Encryption Scheme. Formally, with a deterministic encryption scheme \mathfrak{E} , we associate three finite spaces: the message space \mathcal{M} , the ciphertext space \mathcal{C} , and the key space \mathcal{K} . \mathfrak{E} consists of two deterministic functions: the encryption function $\mathfrak{e} : \mathcal{K} \times \mathcal{M} \rightarrow \mathcal{C}$ and the decryption function $\mathfrak{d} : \mathcal{K} \times \mathcal{C} \rightarrow \mathcal{M}$, such that for any $K \in \mathcal{K}, M \in \mathcal{M}$, we have $\mathfrak{d}(K, \mathfrak{e}(K, M)) = M$. In addition, we associate with \mathfrak{E} a key distribution μ_K (usually uniform over the key-space). For a fixed $K \in \mathcal{K}$, we write $\mathfrak{e}(K, \cdot)$ as $\mathfrak{e}_K(\cdot)$ and $\mathfrak{d}(K, \cdot)$ as $\mathfrak{e}_K^{-1}(\cdot)$.

Block Cipher. A *block cipher* E is a deterministic encryption scheme with $\mathcal{M} = \mathcal{C} = \{0, 1\}^n$ for some fixed n . We call n the *block length* of the E . More specifically, we write E_K , or in short E , to denote the encryption function of the block cipher, and E_K^{-1} , or in short E^{-1} , to denote the decryption function of the block cipher. E is usually modelled as an n -bit prp. It is a well-known result that for any family \mathcal{F} of block functions, the prf-advantage of any adversary \mathcal{A} making at most q queries cannot exceed its prp-advantage by more than $q^2/2^n$. This result, known as the *prp-prf switching lemma* [BR04], lets us treat a block cipher as a prf.

Almost-XOR-Universal Hash Functions. A keyed hash-function $\mathcal{H} : \mathcal{K} \times \mathcal{X} \rightarrow \mathcal{Y}$ takes a key $K \in \mathcal{K}$ and a message $X \in \mathcal{X}$ and outputs an element $Y \in \mathcal{Y}$. \mathcal{H} is said to be ϵ -almost-XOR-universal, or in short ϵ -AXU, if for any $X_1, X_2 \in \mathcal{X}$ and any $Y \in \mathcal{Y}$, on sampling $K \leftarrow_{\S} \mathcal{K}$, we have $\Pr[\mathcal{H}_K(X_1) \oplus \mathcal{H}_K(X_2) = Y] \leq \epsilon$. Sometimes we omit the term ‘ ϵ ’ and only say “AXU hash” to denote an ϵ -AXU hash with “good” value of ϵ . Hash functions based on univariate polynomials, e.g., Poly1305 [Ber05] and GHASH [MV04] are examples of AXU hashes.

3 t-OleF: A Tweakable Online Cipher

We first specify the notion of “online” as used in this paper, and other relevant notions, before describing the new construction t-OleF.

3.1 Tweakable Online Permutation and Cipher

Blocks and Block-Prefixes. Let $\mathbf{B} := \{0, 1\}^n$ be the *block space*, the set of complete blocks; let $\mathbf{IB} := \{0, 1\}^{\leq n-1}$, the set of incomplete blocks. Let $\mathcal{M}_D := \{x \mid 2n \leq \|x\| \leq 2nl\}$ be the message space.¹ We'll parse a message as a sequence of two or more complete blocks optionally followed by an incomplete block. For any $x \in \mathcal{M}_D$, denote the number of blocks in x as $\langle x \rangle := \lceil \|x\|/n \rceil$. For $r \in [\langle x \rangle - 1]$, let the r -th block of x be defined as $x[r] := x_{n(r-1)+1..nr}$; let the final block of x be defined as $x[\langle x \rangle] := x_{n(\langle x \rangle - 1) + 1.. \|x\|}$. For $1 \leq i \leq j \leq \langle x \rangle$, let $x[i..j]$ denote the sequence of $(j - i + 1)$ blocks from $x[i]$ to $x[j]$.

```

Module encrypt[K, K']
input  : ℓ' AD blocks A[1], ..., A[ℓ'], ℓ plaintext diblocks (L[1], R[1]), ..., (L[ℓ], R[ℓ])
output: ℓ ciphertext diblocks (L'[1], R'[1]), ..., (L'[ℓ], R'[ℓ])
begin
  T[1] ← HK'(A[1], ..., A[ℓ'])
  if |R[ℓ]| = n then
    for j ← 1 to ℓ do
      (T[j+1], L'[j], R'[j]) ← round[K](j, ℓ, T[j], L[j], R[j])
    end for
  else
    for j ← 1 to ℓ - 2 do
      (T[j+1], L'[j], R'[j]) ← round[K](j, ℓ, T[j], L[j], R[j])
    end for
    if |L[ℓ]| < n then
      LTEMP ← L[ℓ - 1] ⊕ b4 · E(K, pad(L[ℓ]))
      (T[ℓ], L'[ℓ - 1], R'TEMP) ← round[K](ℓ - 1, ℓ, T[ℓ - 1], LTEMP, R[ℓ - 1])
      L'[ℓ] ← L[ℓ] ⊕ chop(E(K, T[ℓ]))
      R'[ℓ - 1] ← R'TEMP ⊕ b4 · E(K, pad(L'[ℓ]))
    else
      LTEMP ← L[ℓ - 1] ⊕ b4 · E(K, L[ℓ]) ⊕ E(K, pad(R[ℓ]))
      (T[ℓ], L'[ℓ - 1], R'TEMP) ← round[K](ℓ - 1, ℓ, T[ℓ - 1], LTEMP, R[ℓ - 1])
      L'[ℓ] ← L[ℓ] ⊕ E(K, T[ℓ])
      R'[ℓ] ← R[ℓ] ⊕ chop(E(K, T[ℓ] ⊕ 1))
      R'[ℓ - 1] ← R'TEMP ⊕ b4 · E(K, L[ℓ]) ⊕ E(K, pad(R'[ℓ]))
    end if
  end if
end

Module round[K]
input  : j, ℓ, T, L, R
output: TNEXT, L', R'
begin
  if j = ℓ then
    bT ← b3 · T
  else
    bT ← b2 · T
  end if
  X ← E(K, L) ⊕ R ⊕ bT
  Y ← E(K, X) ⊕ L
  R' ← E(K, Y) ⊕ X
  L' ← b1 · E(K, R') ⊕ Y ⊕ bT
  TNEXT ← X ⊕ Y
end

```

Algorithm 1: The algorithm of t-OleF[E, \mathcal{H}]. b_1, b_2, b_3, b_4 are distinct small constants that allow efficient multiplication.

Tweakable Online Permutation. We call $\tilde{g} : \mathcal{T} \times \mathcal{M}_D \rightarrow \mathcal{M}_D$ a *tweakable online permutation* if for every $T \in \mathcal{T}$, $\tilde{g}(T, \cdot)$ is a permutation over \mathcal{M}_D such that for every $x, y \in \mathcal{M}_D$ and for every r with $\|x\| - 2nr \geq 2n, \|y\| - 2nr \geq 2n$,

$$x[1..2r] = y[1..2r] \implies \tilde{g}(T, x)[1..2r] = \tilde{g}(T, y)[1..2r].$$

¹For sampling purposes we need a theoretical upper limit on the message length, but the results in this paper hold for arbitrarily large l .

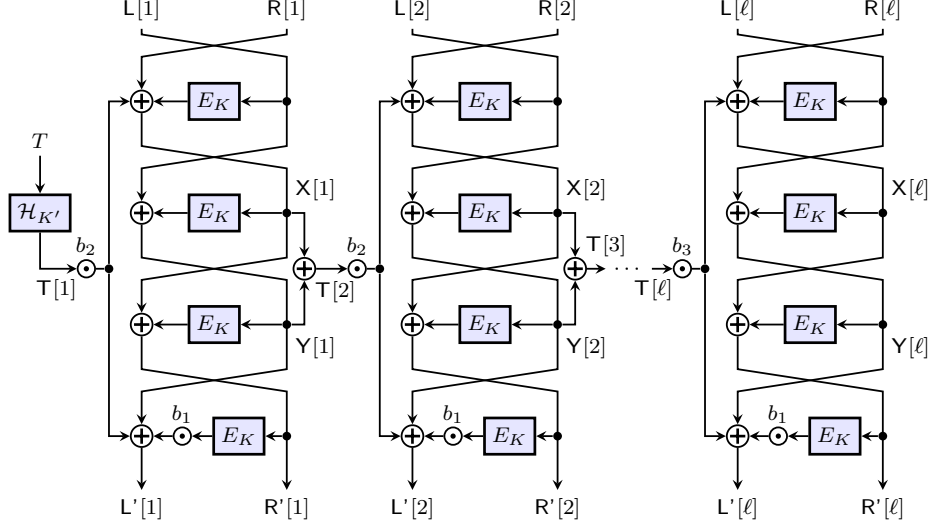


Figure 1: $\text{t-OleF}[E, \mathcal{H}]_{K, K'}$ processing ℓ complete diblocks where \odot denotes field multiplication

We write $\tilde{g}^{-1} : \mathcal{T} \times \mathcal{M}_D \rightarrow \mathcal{M}_D$ to denote the tweakable online permutation satisfying $\tilde{g}^{-1}(T, \cdot) = \tilde{g}(T, \cdot)^{-1}$ for each $T \in \mathcal{T}$. Note that in this definition, the online property does not extend to the last $2n$ bits of the output, which are allowed to behave arbitrarily, even when the corresponding input bits form part of a common prefix.

It will often be convenient to treat an odd-numbered block and the next even-numbered block together as a single entity, which we will call a *diblock*; elements of \mathcal{B}^2 will be called *complete diblocks* and those of $\text{IB} \cup \mathcal{B} \cup (\mathcal{B} \times \text{IB})$ will be called *incomplete diblocks* (which include incomplete single blocks and complete single blocks to account for messages with an odd number of blocks in total). Messages will be parsed into a sequence of complete diblocks optionally followed by an incomplete diblock.

Tweakable Online Cipher. Formally, a *tweakable online cipher* consists of an encryption function $\epsilon : \mathcal{K} \times \mathcal{T} \times \mathcal{M}_D \rightarrow \mathcal{M}_D$ and a decryption function $\vartheta : \mathcal{K} \times \mathcal{T} \times \mathcal{M}_D \rightarrow \mathcal{M}_D$ satisfying the following:

- It must be *correct*. For every $(K, T, x) \in \mathcal{K} \times \mathcal{T} \times \mathcal{M}_D$,
$$\vartheta(K, T, \epsilon(K, T, x)) = x;$$
- It must be *online*. For every $(K, T, x, y) \in \mathcal{K} \times \mathcal{T} \times \mathcal{M}_D^2$, and for every r with $\|x\| - 2nr \geq 2n$, $\|y\| - 2nr \geq 2n$,
$$x[1..2r] = y[1..2r] \implies \epsilon(K, T, x)[1..2r] = \epsilon(K, T, y)[1..2r].$$

The sprtop game. Let TOPerm be the set of all tweakable online permutations over \mathcal{M}_D with tweak-space \mathcal{T} . In the *strong pseudorandom tweakable online permutation (sprtop)* game, the adversary tries to distinguish between \mathcal{O}_0 representing $\tilde{g} \leftarrow_{\S} \text{TOPerm}$, and \mathcal{O}_1 representing $\mathfrak{o}\mathcal{E}_K$ for $K \leftarrow_{\S} \mathcal{K}$ where $\mathfrak{o}\mathcal{E}$ is a tweakable online cipher.

3.2 Specifications and Security of t-OleF

We now propose a tweakable online cipher, which we call **t-OleF**. It is a tweakable variant of the online cipher **OleF** [BN16]. **Figure 1** shows the construction of **t-OleF** for a message with ℓ complete diblocks.

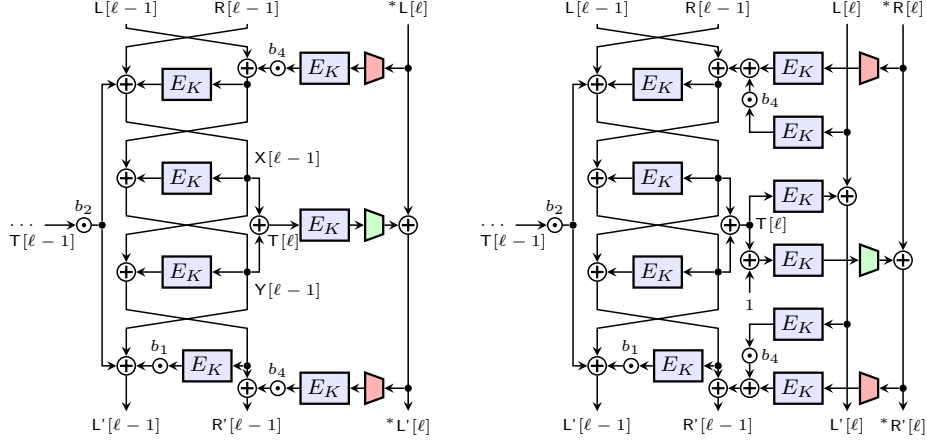


Figure 2: $t\text{-OleF}[E, \mathcal{H}]_{K, K'}$ processing an incomplete diblock, when **(left)** $\|*L[\ell]\| = t < n$, and when **(right)** $\|*R[\ell]\| = t < n$. The $\rightarrow \text{green arrow} \leftarrow$ nodes indicate truncation to t bits, and the $\leftarrow \text{red arrow} \leftarrow$ nodes indicate suffix padding with 10^{n-t-1} .

$E : \mathcal{K} \times \mathcal{B} \rightarrow \mathcal{B}$ is a block cipher, and $\mathcal{H} : \mathcal{K} \times \mathcal{T} \rightarrow \mathcal{B}$ is an ϵ -AXU hash-function that takes the tweak as input, and outputs a block. The keys of E and \mathcal{H} are sampled independently. For partial-diblock messages, we carry over the technique used in OleF, as shown in Figure 2. Algorithm 1 gives the complete specification of $t\text{-OleF}$.

Slightly modifying the proof used in [BN16] to show the security of OleF, we can show the following sprtop security bound for $t\text{-OleF}[E, \mathcal{H}]$ for any block cipher E and any ϵ -AXU hash function \mathcal{H} . A sketch of the modified proof is included in Appendix A.

Theorem 1. *For any sprtop -adversary \mathcal{A} of $t\text{-OleF}[E, \mathcal{H}]$ limited to q queries consisting of up to σ query blocks in total, there is a prf -adversary \mathcal{A}' of E making at most $10\sigma/3$ queries such that*

$$\text{Adv}_{t\text{-OleF}[E, \mathcal{H}]}^{\text{sprtop}}(\mathcal{A}) \leq \frac{7\sigma^2}{2^n} + \text{Adv}_E^{\text{prf}}(\mathcal{A}') + 3q^2\epsilon.$$

4 OIÆF: An Online AE scheme

We begin this section by building towards our proposed notion of online authenticated encryption.

4.1 A Variable-Expansion Tweakable Online Injection

Definitions. Fix an *expansion-cap* $t \leq 2n(l-1)$. Define $\mathbb{T} := \{0, 1\}^{\leq t}$, and $\mathcal{M}_D^- := \{x \mid 2n-1 \leq \|x\| \leq 2nl-t\}$. An injection $\gamma : [t] \times \mathcal{M}_D^- \rightarrow \mathcal{M}_D^- \times \mathbb{T}$ is called a *variable-expansion injection* if for every $\tau \in [t]$, $\gamma_\tau := \gamma(\tau, \cdot)$ is τ -expanding. $\gamma_\tau(x)_{\|x\|+1.. \|x\|+\tau}$ is called the *tag* associated to x by γ . We call γ a *variable-expansion online injection* if, for every $\tau \in [t]$, for every $x, y \in \mathcal{M}_D^-$, and for every r with $\|x\| - 2nr \geq 2n - \tau$ and $\|y\| - 2nr \geq 2n - \tau$,

$$x[1..2r] = y[1..2r] \implies \gamma_\tau(x)[1..2r] = \gamma_\tau(y)[1..2r].$$

A function $\tilde{\gamma} : \mathcal{T} \times [t] \times \mathcal{M}_D^- \rightarrow \mathcal{M}_D^- \times \mathbb{T}$ is called a *variable-expansion tweakable online injection* if for every $T \in \mathcal{T}$, $\tilde{\gamma}_T := \tilde{\gamma}(T, \cdot, \cdot)$ is a variable-expansion online injection. It is useful to note that $\mathcal{M}_D^- \times \mathbb{T}$ coincides with \mathcal{M}_D .

Injective Suffix-Pad. A function $\phi : [\mathfrak{t}] \times \mathcal{M}_D^- \rightarrow \mathcal{M}_D$ is called an *injective suffix-pad* if ϕ is an injection such that for any $\tau \in [\mathfrak{t}]$ and any $x \in \mathcal{M}_D^-$, we can find $y \in \{0, 1\}^\tau$ such that $\phi(\tau, x) = x\|y$. For an injective suffix-pad ϕ , for $\tau \in [\mathfrak{t}]$, define $\phi_\tau := \phi(\tau, \cdot)$, and let $\mathcal{M}_D^\phi[\tau] := \{\phi_\tau(x) \mid x \in \mathcal{M}_D^-\}$ be the range of ϕ_τ . Then ϕ_τ is a bijection from \mathcal{M}_D^- to $\mathcal{M}_D^\phi[\tau]$, with an inverse $\phi_\tau^{-1} : \mathcal{M}_D^\phi[\tau] \rightarrow \mathcal{M}_D^-$. We can define the *unverified inverse* $\phi^{[-1]} : [\mathfrak{t}] \times \mathcal{M}_D \rightarrow \mathcal{M}_D^-$ of ϕ as $\phi^{[-1]}(\tau, z) := z_{1.. \|z\| - \tau}$; the name indicates the fact that when $z \in \mathcal{M}_D^\phi[\tau]$, $\phi^{[-1]}(\tau, \cdot)$ exactly coincides with ϕ_τ^{-1} . A common example of an injective suffix-pad is ϕ_{10^*} defined by $\phi_{10^*}(\tau, x) := x\|10^{\tau-1}$.

An Equivalence Result. We now come to an important equivalence result. Let TOInj be the set of all tweakable online injections with the parameters as above.

Lemma 1. *Sample $\tilde{g} \leftarrow_{\S} \text{TOPerm}$ and $\tilde{\gamma} \leftarrow_{\S} \text{TOInj}$. Fix a tweak $T \in \mathcal{T}$ and define $h_T := \tilde{g}_T \circ \phi$ for some injective suffix-pad ϕ . Then for any q distinct pairs $(\tau_1, x_1), \dots, (\tau_q, x_q) \in [\mathfrak{t}] \times \mathcal{M}_D^-$, $(h_T(\tau_1, x_1), \dots, h_T(\tau_q, x_q))$ and $(\tilde{\gamma}_T(\tau_1, x_1), \dots, \tilde{\gamma}_T(\tau_q, x_q))$ are identically distributed.*

In other words, if we take a random tweakable online permutation, and compose it with an injective suffix-pad, we get a random tweakable online injection.

Proof. Consider the set \mathcal{P} of all distinct complete-diblock proper prefixes in $\{x_1, \dots, x_q\}$. We observe that under any diblock-online function f every r -diblock $x \in \mathcal{P}$ can be associated with a unique diblock, which is the r -th diblock of $f(x\|y)$ for any y with $\|y\| \geq 2n$. Thus, once T is fixed, from each of these prefixes we have one ciphertext diblock from h_T and one from $\tilde{\gamma}_T$, which are identically distributed. Since ϕ is a suffix-pad, the block associated with a prefix $x \in \mathcal{P}$ by h_T is exactly the block associated with x by \tilde{g}_T , the distribution of which is identical to that of the one associated with x by $\tilde{\gamma}_T$. Similarly, for both, the tag will be identically distributed. The only boundary case is when the online-but-last effect comes into play. To resolve that, we simply note that h_T is also an online injection, since for every $\tau \in [\mathfrak{t}]$, for every $x, y \in \mathcal{M}_D$, and for every r with $\|x\| - 2nr \geq 2n - \tau$, $\|y\| - 2nr \geq 2n - \tau$,

$$x[1..2r] = y[1..2r] \implies h_T(\tau, x)[1..2r] = h_T(\tau, y)[1..2r].$$

Since the blocks assigned to the diblock-prefixes, along with the tags, constitute the ciphertexts, the result follows. □

4.2 Online Authenticated Encryption

Definition. Formally, an *online authenticated encryption (AE) scheme* consists of an encryption function $\tilde{\mathfrak{e}} : \mathcal{K} \times \mathcal{T} \times [\mathfrak{t}] \times \mathcal{M}_D^- \rightarrow \mathcal{M}_D$, a decryption function $\tilde{\mathfrak{d}} : \mathcal{K} \times \mathcal{T} \times [\mathfrak{t}] \times \mathcal{M}_D \rightarrow \mathcal{M}_D^-$, and a verification function $\tilde{\mathfrak{v}} : \mathcal{K} \times \mathcal{T} \times [\mathfrak{t}] \times \mathcal{M}_D \rightarrow \{\text{accept}, \text{reject}\}$ satisfying the following:

- It must be *correct*. For any $(K, T, \tau, x) \in \mathcal{K} \times \mathcal{T} \times [\mathfrak{t}] \times \mathcal{M}_D^-$,

$$\tilde{\mathfrak{d}}(K, T, \tau, \tilde{\mathfrak{e}}(K, T, \tau, x)) = x, \quad \tilde{\mathfrak{v}}(K, T, \tau, \tilde{\mathfrak{e}}(K, T, \tau, x)) = \text{accept}.$$

- It must be *online*. For any $K \in \mathcal{K}$, the function $\tilde{\mathfrak{e}}_K := \tilde{\mathfrak{e}}(K, \cdot, \cdot, \cdot)$ is a variable-expansion tweakable online injection.

```

Module Enc [ $K, K'$ ]
input : associated data  $A$ , plaintext  $M$ , expansion  $\tau$ 
output: ciphertext  $C$ 

begin
  |  $P \leftarrow \phi_{10^*}(M, \tau)$ 
  |  $C \leftarrow \text{t-OleF}[E, \mathcal{H}](K, K', A, P)$ 
end

Module Dec [ $K, K'$ ]
input : associated data  $A$ , ciphertext  $C$ , expansion  $\tau$ 
output: unverified plaintext  $M$ 

begin
  |  $P \leftarrow \text{t-OleF}^{-1}[E, \mathcal{H}](K, K', A, C)$ 
  |  $M \leftarrow \phi_{10^*}^{[-1]}(P)$ 
end

Module Ver [ $K, K'$ ]
input : associated data  $A$ , ciphertext  $C$ , expansion  $\tau$ 
output: flag  $F$ 

begin
  |  $P \leftarrow \text{t-OleF}^{-1}[E, \mathcal{H}](K, K', A, C)$ 
  | if  $P_{\|P\|-\tau+1..\|P\|} = 10^{\tau-1}$  then
  |   |  $F \leftarrow \text{accept}$ 
  | else
  |   |  $F \leftarrow \text{reject}$ 
  | end if
end

```

Algorithm 2: The algorithm of $\text{OI}\mathcal{A}\text{EF}[E, \mathcal{H}]$. The messages are encoded using ϕ_{10^*} .

Security Game. In the OAE security game, the real oracle \mathcal{O}_1 has three functions associated with it: encryption Enc_1 , decryption Dec_1 , and verification Ver_1 , and the ideal oracle \mathcal{O}_0 has three functions associated with it: encryption Enc_0 , decryption Dec_0 , and verification Ver_0 . The adversary \mathcal{A} can make queries to all the functions of the oracle it is interacting with. Enc_0 is a random tweakable online injection $\tilde{\gamma} \leftarrow_{\S} \text{TOInj}$. However, by [Lemma 1](#), we can take an injective suffix-pad ϕ and a random tweakable online permutation $\tilde{g} \leftarrow_{\S} \text{TOPerm}$, and compose them, and what we arrive at is identical in distribution to $\tilde{\gamma}$. Thus, we can assume either of the two to be Enc_0 , and we choose the latter because it makes our proof easier. [Figure 3](#) summarises the oracles and the corresponding functions mentioned above.

An online authenticated encryption scheme strives to attain two major security goals: *privacy*—the ciphertexts should “look like” random bit-strings, and *integrity*—it should be difficult to *forge* a valid ciphertext. We formalise these two notions as two different security games with the oracles defined above:

- *Privacy* The privacy game (*priv*) is a distinguishing game played with either \mathcal{O}_0 or \mathcal{O}_1 .
- *Integrity* The integrity game (*int*) is played with \mathcal{O}_1 and expansion τ^* . \mathcal{A} wins if one of the verification queries with expansion τ^* returns *accept*.

In both games we add the stipulation that \mathcal{A} is not allowed to feed the output of an encryption query unchanged into a verification query with the same tweak (otherwise the integrity game would be trivial

Ideal Oracle \mathcal{O}_0	Real Oracle \mathcal{O}_1
$\tilde{g} \leftarrow_{\S} \text{TOPerm}$	$K \leftarrow_{\S} \mathcal{K}$
<u>$\text{Enc}_0(T, \tau, x)$</u> return $\tilde{g}(T, \phi(\tau, x))$	<u>$\text{Enc}_1(T, \tau, x)$</u> return $\epsilon(K, T, \phi(\tau, x))$
<u>$\text{Dec}_0(T, \tau, y)$</u> $z \leftarrow \tilde{g}^{-1}(T, y)$ return $\phi^{[-1]}(\tau, z)$	<u>$\text{Dec}_1(T, \tau, y)$</u> $z \leftarrow \mathfrak{d}(K, T, y)$ return $\phi^{[-1]}(\tau, z)$
<u>$\text{Ver}_0(T, \tau, y)$</u> $z \leftarrow \tilde{g}^{-1}(T, y)$ if $z \in \mathcal{M}_D^\phi[\tau]$: return accept else : return reject	<u>$\text{Ver}_1(T, \tau, y)$</u> $z \leftarrow \mathfrak{d}(K, T, y)$ if $z \in \mathcal{M}_D^\phi[\tau]$: return accept else : return reject

Figure 3: Oracles for online authenticated encryption security games

to win). Note that in the integrity game the target expansion τ^* is only used to check for a valid forging attempt— \mathcal{A} is free to make queries with other expansions.

4.3 A Generic Construction

We’re now in a position to put everything together and propose a generic technique of constructing an online authenticated encryption scheme from a tweakable online cipher.

For our construction, we use the encode-then-encipher paradigm, where the message is first put through a public encoding and then encrypted using a keyed primitive. The encoding is done with $\phi : [\mathfrak{t}] \times \mathcal{M}_D^- \rightarrow \mathcal{M}_D$, and the enciphering is done with the encryption function $\epsilon : \mathcal{K} \times \mathcal{T} \times \mathcal{M}_D \rightarrow \mathcal{M}_D$ of a tweakable online encryption scheme $\mathfrak{oE} = (\epsilon, \mathfrak{d})$. Formally, the encryption, decryption, and verification functions of our online authenticated encryption scheme $\mathfrak{oaE} = (\tilde{\epsilon}, \tilde{\mathfrak{d}}, \tilde{\mathfrak{v}})$ are defined as follows:

$$\begin{aligned}
\tilde{\epsilon}(K, T, \tau, x) &:= \epsilon(K, T, \phi(\tau, x)), \\
\tilde{\mathfrak{d}}(K, T, \tau, y) &:= \phi^{[-1]}(\tau, \mathfrak{d}(K, T, y)), \\
\tilde{\mathfrak{v}}(K, T, \tau, y) &:= \text{accept}, & \text{if } \mathfrak{d}(K, T, y) \in \mathcal{M}_D^\phi[\tau], \\
&= \text{reject}, & \text{otherwise.}
\end{aligned}$$

Security Analysis. We next prove the following security result for \mathfrak{oaE} .

Theorem 2. *For any priv-adversary \mathcal{A} of \mathfrak{oaE} making q queries with a maximum of σ query blocks, we can find an sprtop-adversary \mathcal{A}' of \mathfrak{oE} making q queries with a maximum of σ query blocks such that*

$$\mathbf{Adv}_{\mathfrak{oaE}}^{\text{priv}}(\mathcal{A}) \leq \mathbf{Adv}_{\mathfrak{oE}}^{\text{sprtop}}(\mathcal{A}').$$

Also, for any int-adversary \mathcal{A} of oaE with target expansion $\tau^* \leq 2n$ making q queries with a maximum of σ query blocks, among which there are q' forging attempts (i.e., decryption queries with expansion τ^*), we can find an sprtop -adversary \mathcal{A}' of oE making q queries with a maximum of σ query blocks such that

$$\text{Adv}_{\text{oaE}}^{\text{int}}(\mathcal{A}) \leq \text{Adv}_{\text{oE}}^{\text{sprtop}}(\mathcal{A}') + \frac{q'}{2^{\tau^*}}.$$

Proof. Privacy This is a standard reduction proof. \mathcal{A}' uses \mathcal{A} to play an sprtop -game against oE as follows: for every encryption query (T, τ, x) of \mathcal{A} , \mathcal{A}' makes an encryption query with $(T, \phi(\tau, x))$ and passes on the response to \mathcal{A} as the output of Enc_b . For every decryption query (T, τ, y) of \mathcal{A} , \mathcal{A}' makes the decryption query (T, y) to receive z as response; and passes on $\phi^{[-1]}(\tau, z)$ to \mathcal{A} as the output of Dec_b . Finally, for every verification query (T, τ, y) of \mathcal{A} , \mathcal{A}' makes the decryption query (T, y) to receive z as response; and passes on **accept** if $z \in \mathcal{M}_D^\phi[\tau]$, and **reject** otherwise, to \mathcal{A} as the output of Ver_b . At the end \mathcal{A}' outputs the output bit of \mathcal{A} , and wins if \mathcal{A} wins, thus proving the claimed bound.

Integrity We first replace \mathcal{O}_1 by \mathcal{O}_0 using the upper-bound of the priv -advantage. Thus in order to win \mathcal{A} must produce a ciphertext (T, y) such that $\tilde{g}_T^{-1}(y)$ is in the range of $\phi(\tau^*, \cdot)$, where $\tilde{g} \leftarrow_{\S} \text{TOPerm}$. The online-but-last property ensures that at least the last $2n$ bits of $\tilde{g}_T^{-1}(y)$ always have fresh randomness for all distinct y , and $\tau^* \leq 2n$, so the last τ^* bits for any forging attempt can only match the required suffix with a probability of $1/2^{\tau^*}$. This completes the proof of the claimed bound. \square

A Word on Integrity. We've just shown that our scheme when used with a fully secure oE gives an integrity of τ^* bits (for $\tau^* \leq 2n$). This does not, however, guarantee a practically useful integrity, because for very low values of τ^* , the scheme can be forged by brute-force. We do not see this as a weakness of the scheme; indeed it is rather an advantage of variable-expansion authenticated encryption schemes that the user can select τ^* to suit his integrity requirements. Note that a user only needs to worry about forge attacks with the expansion of his choice; any variable-expansion authenticated encryption scheme can be forged for low expansion values, and that does not imply a weakness in the scheme. Thus, it is useful to talk about integrity of a scheme for specific expansions, instead of a general integrity. We also point out that the privacy of the scheme remains intact even for very low values of τ^* .

4.4 OIÆF: Security

Now we propose OIÆF, a provably secure online authenticated encryption scheme. To obtain OIÆF, the tweakable online cipher $\text{t-OleF}[E, \mathcal{H}]$ from [Section 3](#) is plugged into the generic construction from [Section 4.3](#). [Algorithm 2](#) gives the complete specification of OIÆF. We can show the following security bound for OIÆF using the bounds from [Theorem 1](#) and [Theorem 2](#).

Theorem 3. *For any priv-adversary \mathcal{A} of OIÆF making q queries with a maximum of σ query blocks, we can find a prf-adversary \mathcal{A}' of E making at most $10\sigma/3$ queries such that*

$$\text{Adv}_{\text{OIÆF}}^{\text{priv}}(\mathcal{A}) \leq \frac{7\sigma^2}{2^n} + \text{Adv}_E^{\text{prf}}(\mathcal{A}') + 3q^2\epsilon.$$

For any int-adversary \mathcal{A} of OIÆF with target expansion $\tau^ \leq 2n$ making q queries with a maximum of σ query blocks, among which there are q' forging attempts (i.e., decryption queries with expansion τ^*), we can find a prf-adversary \mathcal{A}' of E making at most $10\sigma/3$ queries such that*

$$\text{Adv}_{\text{OIÆF}}^{\text{int}}(\mathcal{A}) \leq \frac{7\sigma^2}{2^n} + \text{Adv}_E^{\text{prf}}(\mathcal{A}') + 3q^2\epsilon + \frac{q'}{2^{\tau^*}}.$$

In the above statement, we assume that in the prf -game against E the adversary \mathcal{A}' can make encryption queries with chosen keys and chosen plaintexts. The birthday-term in the second bound implies that τ^* can be at most $\approx n/2$ in order to guarantee around τ^* bits of integrity.

As an injective suffix-padding scheme, we propose ϕ_{10^*} . Note that while hashing the associated data, since inverse-free is not a relevant property in the context of hashing (it being itself a non-invertible function), we do not need to handle diblocks, and can process it block-wise.

5 An Application: Bidirectional Online Channels

In this section, we build a formalism and construct a primitive for communication between parties that we call a *bidirectional online channel*. Our channels are particularly suited for communication between lightweight devices that send and receive messages in a streaming fashion due to memory constraints. To this end, our primitive ensures online security even in the presence of ‘state resets’; a special case of this is when the state is ‘reset’ after every send/receive operation for both parties, i.e., when the local state is not updatable.

5.1 Online Channels

We first define a syntax and correctness notions for an online channel. In our formulation, we consider two parties, A and B, who can both send and receive messages between each other. We assume that parties initially have performed key exchange or have otherwise agreed upon initial keying material (abstracted into the function init in our formalism).

Definition. Formally, we associate an online channel scheme with the following sets: the randomness space \mathcal{R} , the state space for both parties \mathcal{S} , the message space $\mathcal{M} = \mathcal{M}_D^-$, the ciphertext space $\mathcal{C} = \mathcal{M}_D$ and the associated data space \mathcal{T} . Let $\mathbb{Z}_{>0}$ denote the set comprised of all integers greater than 0.

We assume that a state st_P for party $P \in \{\text{A}, \text{B}\}$ has a variable $ooo \in \{0, 1\}$; semantically, this variable is set to 1 whenever a message is delivered out-of-order (at least when there are no state resets).

We also assume that ciphertexts can be split into two components c_{core} and c_{aux} with the implementation-specific helper function $(c_{core}, c_{aux}) \leftarrow \text{getaux}(c)$.² Then, an online channel consists of an initialisation function

$$\text{init} : [t] \times \mathcal{R} \longrightarrow \mathcal{S} \times \mathcal{S},$$

that takes as inputs an integer L (the block limit) and randomness r and outputs initial states for A and B; a send function

$$\text{send} : \mathcal{M} \times [t] \times \mathcal{T} \times \mathcal{S} \longrightarrow \mathcal{C} \times \mathbb{Z}_{>0} \times \mathcal{S},$$

that takes as inputs a message, expansion, associated data and the caller’s state, and outputs a ciphertext, an index and a new state; and a receive function

$$\text{receive} : \mathcal{C} \times [t] \times \mathcal{T} \times \mathcal{S} \longrightarrow \mathcal{M} \cup \{\perp\} \times \mathbb{Z}_{>0} \times \mathcal{S},$$

that takes as inputs a ciphertext, expansion, associated data and the caller’s state, and outputs a message and index (or (\perp, \perp) , denoting failure), and a new state; satisfying the following:

- Online Consider $(\text{st}_A, \text{st}_B) \leftarrow \text{init}(L; r)$. Then, for all $(L, r) \in [t] \times \mathcal{R}$ and for all sequences of send and receive calls, let st_P be the state of some $P \in \{\text{A}, \text{B}\}$ at some point during protocol execution.

²Looking ahead, we split the ciphertext into two components so that security will ensure c_{core} is indistinguishable from random.

Then the function $f_i := [(c, \cdot, \cdot) \leftarrow \text{send}(\cdot, \cdot, \cdot, \text{st}_P); (c_1, c_2, \dots, c_{aux}) = (c_{1..(Ln+\tau)}, c_{(Ln+\tau+1)..(2(Ln+\tau))}, \dots, c_{aux}) \leftarrow c; \text{return } c_i]$ is a tweakable online injection for all $i \in [|\!|c|\!|/(Ln + \tau)]$.

- **Good-Case Sequentiality** Consider $(\text{st}_A, \text{st}_B) \leftarrow \text{init}(L; r)$. Then, for all $(L, r, P) \in [l] \times \mathcal{R} \times \{A, B\}$ and for all sequences of **send** and **receive** calls, consider a call $(m, i, \text{st}_B) \leftarrow \text{receive}(c, \tau, a, \text{st}_B)$ made at some point in time by P such that $m \neq \perp$ and $\text{st}_B.\text{ooo} = 0$ (if such a call exists). Let $(m', i', \text{st}_P') \leftarrow \text{receive}(c', \tau', a', \text{st}_P')$ be the next call made by P such that $m' \neq \perp$ and $\text{st}_P'.\text{ooo} = 0$ (if it exists). Then for all such pairs of **receive** calls, $i' = i + 1$.
- **Correctness** Consider $(\text{st}_A, \text{st}_B) \leftarrow \text{init}(L; r)$. Then, for all $(L, r) \in [l] \times \mathcal{R}$ and for all sequences of **send** and corresponding (sequentially made) **receive** calls using the output of **send** and the same associated data a as input (where such **receive** calls are only made once), we have for all $P \in \{A, B\}$, $m, \tau: (c, i, \text{st}_P) \leftarrow \text{send}(m, \tau, a, \text{st}_P) \wedge (m', i', \text{st}_{\bar{P}}) \leftarrow \text{receive}(c, \tau, a, \text{st}_{\bar{P}}) \Rightarrow (i, m) = (i', m')$ (where \bar{P} denotes P 's counterpart).

Primitive design choices. Our primitive is stateful: this is required for sequentiality in the good case scenario that the adversary does not deliver ciphertexts out-of-order to parties.

Algorithm **init** is parametrised by L , which enforces that encryption is performed in batches of L blocks (such that authentication checks are certainly performed every L blocks). Our online condition requires that each ciphertext component corresponding to L plaintext blocks corresponds to a tweakable online injection (as in the online condition for online authenticated encryption). Note that we consider blocks for simplicity but diblocks could easily be considered.

Good-case sequentiality requires the index output by **receive** to increase monotonically (in particular by 1 after each call **receive** that outputs $m \neq \perp$), except when messages are delivered out-of-order or when the state is reset (which we elaborate on below when describing security). Some channels in the literature vary here by, e.g., allowing messages to be received in a ‘sliding window’ or out-of-order (captured in more general robustness predicates in [FGJ20, AMPS22]).

As done for online authenticated encryption, our **send** call (and corresponding **receive** call) supports variable-length expansion and thus takes $\tau \leq t$ input, which corresponds to the expansion of each L -block message sequence (rather than the entire ciphertext).

Security Definitions. We describe the security definition’s oracles in Figure 4, and let $\text{Adv}_{\mathcal{Ch}}^{\text{bocs}}(\mathcal{A})$ denote adversary \mathcal{A} 's advantage in the bidirectional online channels distinguishing security game **bocs** w.r.t. channels scheme \mathcal{Ch} . For simplicity of explanation, we assume that messages consist of kL blocks for positive integer k and block limit L . Our indistinguishability game captures both privacy and ciphertext integrity and is a real-or-random game. The adversary can make oracle queries of the form (x, δ) , where $x \in \{s, r, \ell, sr\}$ corresponds to **send**, **receive**, **leak** and **state reset** queries, respectively, and δ to the relevant input.

In the ideal game (\mathcal{O}_0) , **Send** queries are such that each L block ciphertext component corresponds to the output of a random tweakable online injection (except for the auxiliary output), which, as in Section 4.3, is constructed by composing a random tweakable online permutation \tilde{g} and an injective suffix-pad ϕ . For queries of the form $\text{Receive}_1(c, \tau, \dots)$, we require that $\tau = \tau^*$ for forgery target tag length τ^* .

The tweak in the ideal game comprises both the input associated data and the sender’s current index (i.e., how many times Send_b has been called for party $P \in \{A, B\}$ since their state was reset). We define the leakage function Leak_b as an extension of the unverified decryption Dec_b of online authenticated encryption (Figure 3) to account for the fact that encryption is performed blockwise (we assume that input $|\!|c|\!| = k(nL + \tau) + aux$ for positive integer k and $aux = |\!|c_{aux}|\!|$). We define for \mathcal{O}_1 the concrete leak function for our online channel scheme to be introduced in the next subsection.

Ideal Oracle \mathcal{O}_0	Real Oracle \mathcal{O}_1
$\tilde{g} \leftarrow_{\$} \text{TOPerm}; i_A, i_B \leftarrow 0$	$(\text{st}_A, \text{st}_B) \leftarrow_{\$} \text{init}(\tau, L)$ $(\text{st}_A^*, \text{st}_B^*) \leftarrow (\text{st}_A, \text{st}_B)$
<u>Send₀($m, \tau, a, P \in \{A, B\}$)</u>	$\text{sent}_A, \text{sent}_B \leftarrow \emptyset$
$e \leftarrow \lceil \lceil m \rceil / nL \rceil$ $(m_1, \dots, m_e) \leftarrow (m[1:L], \dots, m[(Le-1)+1:Le])$	<u>Send₁($m, \tau, a, P \in \{A, B\}$)</u>
for $i \in [e]$:	$(c, i, \text{st}_P) \leftarrow \text{send}(m, \tau, a, \text{st}_P)$
$y_i \leftarrow \tilde{g}((a, i_P, e, i), \phi(\tau, m_i))$	$\text{sent}_P \leftarrow \text{sent}_P \cup \{c, \tau, a\}$
$y \leftarrow (y_1, \dots, y_e); i_P \leftarrow i_P + 1$	return (c, i)
return $(y, i_P - 1)$	
<u>Receive₀($c, \tau, a, P \in \{A, B\}$)</u>	<u>Receive₁($c, \tau, a, P \in \{A, B\}$)</u>
return \perp	$(m, i, \text{st}_P) \leftarrow \text{receive}(c, \tau, a, \text{st}_P)$
	if $(c, a) \in \text{sent}_{\bar{P}} \vee (\tau \neq \tau^*)$:
	return \perp
	return (m, i)
<u>Leak₀($c, \tau, a, P \in \{A, B\}$)</u>	<u>Leak₁($c, \tau, a, P \in \{A, B\}$)</u>
$(c_{\text{core}}, c_{\text{aux}}) \leftarrow \text{getaux}(c); (c_1, \dots, c_e) \leftarrow c_{\text{core}}$	$(m, i, \text{st}_P') \leftarrow \text{leak}(c, \tau, a, \text{st}_P)$
$x \leftarrow \min\{i \in [e] : c_i \text{ is invalid}, e + 1\}$	return (m, i)
for $i \in [x - 1]$:	
$z \leftarrow \tilde{g}^{-1}((a, i_P, e, i), c_i)$	
$m_i \leftarrow \phi^{-1}(\tau, z)$	
if $x \leq e$:	<u>Reset₁($P \in \{A, B\}$)</u>
$m_x \leftarrow \tilde{g}^{-1}((a, i_P, e, i), c_i); m_{x+1}, \dots, m_e \leftarrow \perp$	$\text{st}_P \leftarrow \text{st}_P^*$
return $(m_1 \dots m_e, i_P)$	return
<u>Reset₀($P \in \{A, B\}$)</u>	
$i_P \leftarrow 0$	
return	

Figure 4: Online channels security game defined with respect to forgery target tag length τ^* and assuming an encode-then-encipher cipher used for sending/receiving; injective suffix-pad ϕ is defined as in [Section 4.3](#). `getaux` is a construction-dependent helper that separates the ‘core’ ciphertext and auxiliary data that need not be indistinguishable from random. `leak` is a construction-dependent leakage function that captures what kind of release of unverified plaintext is allowed.

Our game allows the state of each party to be adaptively ‘reset’ over the lifetime of the game’s execution. This captures a setting where a device’s memory is erased or reset over time. This is captured in the Reset_b oracle, which in the ideal world simply sets the send counter for P to 0, and in the real world sets the state of P to the state st_{P} originally output by init . We require that the cryptographic portion of the ciphertext comprises random tweakable online injections in this setting or when messages are delivered out-of-order (given neither of these occur, each L -block ciphertext portion corresponds to a random tweakable injection).

Our game departs from some previous work which considers a left-or-right indistinguishability game (i.e., where Send_b would take messages (m_0, m_1) with $m_0 = m_1$ as input and encrypt m_b), rather than a real-or-random game, as we feel it is more meaningful to define Leak_b in a real-or-random game.

5.2 Online Channels from Online Authenticated Encryption

Construction. We present our online channels construction $\mathcal{C}\mathfrak{h}$ in [Algorithm 3](#) which uses online authenticated encryption scheme $\text{oa}\mathcal{E}$. Essentially, our construction uses $\text{oa}\mathcal{E}$ to encrypt/decrypt (via $\tilde{\mathfrak{e}}/\tilde{\mathfrak{d}}$) in L block batches. Note that we include several values in the tweak to prevent trivial forgeries from an adversary who tries to mix and match different ciphertext values. It otherwise keeps track of indices to enforce our correctness requirements. We define helpers leak and getaux as follows:

- **leak:** We define leak as running receive except if the first i (but not all) $\tilde{\mathfrak{d}}$ calls were successful and output (m_1, \dots, m_i) , the function outputs $(m_1, \dots, m_i, m^*, \perp, \dots, \perp)$ where m^* denotes the unverified plaintext associated with the $(i + 1)$ -th call to $\tilde{\mathfrak{d}}$ if it exists.
- **getaux(c):** On input $c \leftarrow (c_{\text{core}}, c_{\text{aux}})$, getaux returns c_{aux} as defined in [Algorithm 3](#) (which is just an integer).

Our channels are ‘robust’ in the sense of [\[BSJ⁺17\]](#) (and unlike those of [\[MP17\]](#)) in that a single incorrect ciphertext will not result in the caller’s state being set to \perp , as the caller’s state is unchanged after a decryption failure. Correctness follows from the correctness of $\text{oa}\mathcal{E}$, the online property follows from the online property of $\text{oa}\mathcal{E}$ and good-case sequentiality by inspection of how indices are managed in the construction.

Security. We state the following security result for $\mathcal{C}\mathfrak{h}$ proven in [Section B.1](#).

Theorem 4. *For any bocS-adversary \mathcal{A} of $\mathcal{C}\mathfrak{h}$ making q queries with a maximum of σ query blocks, we can find an int-adversary \mathcal{A}' of $\text{oa}\mathcal{E}$ and a priv-adversary \mathcal{A}'' of $\text{oa}\mathcal{E}$ each making at most σ queries w.r.t. messages/ciphertexts of at most L blocks such that*

$$\text{Adv}_{\mathcal{C}\mathfrak{h}}^{\text{bocS}}(\mathcal{A}) \leq \text{Adv}_{\text{oa}\mathcal{E}}^{\text{int}}(\mathcal{A}') + \text{Adv}_{\text{oa}\mathcal{E}}^{\text{priv}}(\mathcal{A}'').$$

6 Conclusion and Future Work

We have proposed new definitions for tweakable online encryption and online authenticated encryption under the release of unverified plaintext, and provided provably secure instantiations for both. We also detailed a salient application of our online mode, namely to construct online bidirectional channels. Our online authenticated encryption scheme $\text{OI}\mathfrak{AE}\mathfrak{F}$ uses two independent keys, and a possible future work could explore whether the same security can be achieved with a single-keyed construction. Another avenue of work could be beyond-birthday constructions satisfying these definitions. Even though we haven’t provided a proof of this, it seems to suggest that t-OleF , the online sprp used in $\text{OI}\mathfrak{AE}\mathfrak{F}$, is optimal in the number of block cipher calls for an inverse-free construction of an online sprp, due to a result by Nandi [\[Nan15\]](#), which in future work should be confirmed or contradicted. Another direction is to adapt our online channel formalism to better suit different settings, which we elaborate on in [Section B.2](#).


```

Module init
input : block limit  $L$ 
output: initial states  $st_A$  and  $st_B$ 
begin
   $k \leftarrow_{\mathcal{S}} \mathcal{K}$ 
   $t_A, t_B \leftarrow (0, 0)$ 
   $ooo \leftarrow 0$ 
   $st_A \leftarrow (k, L, t_A, t_B)$ 
   $st_B \leftarrow st_A$ 
end

Module send
input : message  $m$ , expansion  $\tau$ , associated data  $a$  and state  $st_P = (k, L, t_A, t_B, ooo)$  for  $P \in \{A, B\}$ 
output: ciphertext  $c$ , index  $i$  and updated state  $st_P'$ 
begin
   $e \leftarrow \lceil \ell/L \rceil$  // encryption count
  parse  $B_1 || \dots || B_\ell \leftarrow m$  // parsing  $m$  in blocks of length  $n$ 
   $c_0 \leftarrow \perp$ 
  // encrypting in  $L$  block batches
  for  $i \leftarrow 1$  to  $e$  do
     $T_i \leftarrow \text{combine}(t_P, e, i, a, c_{i-1}, P)$ 
     $m_i \leftarrow B_{(i-1)L+1} || \dots || B_{iL}$ 
     $c_i \leftarrow \bar{e}(k, T, \tau, m_i)$ 
  end for
   $c_{core} \leftarrow (c_1, \dots, c_e)$ ;  $(i, c_{aux}) \leftarrow (t_P, t_P)$ ;  $c \leftarrow (c_{core}, c_{aux})$ 
   $t_P \leftarrow t_P + 1$ ;  $st_P' \leftarrow (k, L, t_A, t_B)$ 
end

Module receive
input : ciphertext  $c$ , expansion  $\tau$ , associated data  $a$  and state  $st_P = (k, L, t_A, t_B)$  for  $P \in \{A, B\}$ 
output: message  $m$  (or  $\perp$ ), index  $i$  (or  $\perp$ ) and (possibly) updated state  $st_P'$ 
begin
  parse  $((c_1, \dots, c_e), j) \leftarrow c$ ;  $c_0 \leftarrow \perp$ 
  // decrypting in  $L$  block batches
  success  $\leftarrow 1$ 
  for  $i \leftarrow 1$  to  $e$  do
    if  $\|c_i\| \neq nL + \tau$  then
      success  $\leftarrow 0$ 
      break
    end if
     $T_i \leftarrow \text{combine}(j, e, i, a, c_{i-1}, \bar{P})$ 
     $m_i \leftarrow \bar{d}(k, T, \tau, c_i)$ 
    if  $m_i = \perp$  then
      success  $\leftarrow 0$ 
      break
    end if
  end for
  if success = 1 then
    // successful decryption
    if  $j \neq t_{\bar{P}} + 1$  then
       $ooo \leftarrow 1$ 
    end if
     $m \leftarrow (m_1 || \dots || m_e)$ ;  $i \leftarrow t_{\bar{P}}$ 
     $t_{\bar{P}} \leftarrow t_{\bar{P}} + 1$ ;  $st_P' \leftarrow (k, L, t_A, t_B)$ 
  else
    // decryption failure
     $m \leftarrow \perp$ ;  $i \leftarrow \perp$ ;  $st_P' \leftarrow st_P$ 
    break
  end if
end

```

Algorithm 3: Online channels scheme $\mathcal{C}\mathfrak{h}$ from online authenticated encryption scheme $oa\mathcal{E}$. Function `combine` is an injection that maps integers t_P , e and i , ciphertext c_{i-1} or bottom value \perp , associated data a and party identifier P to associated data T .

References

- [ABB⁺] Elena Andreeva, Begül Bilgin, Andrey Bogdanov, Atul Luykx, Florian Mendel, Bart Mennink, Nicky Mouha, Qingju Wang, and Kan Yasuda. Primates v1.02. *Submission to CAESAR Competition*, 2013 - 2019. <https://competitions.cr.yp.to/round2/primatesv102.pdf>. (Cited on p. 3.)
- [ABB⁺14] Elena Andreeva, Begül Bilgin, Andrey Bogdanov, Atul Luykx, Bart Mennink, Nicky Mouha, and Kan Yasuda. APE: authenticated permutation-based encryption for lightweight cryptography. In Carlos Cid and Christian Rechberger, editors, *FSE 2014*, volume 8540 of *LNCS*, pages 168–186. Springer, 2014. (Cited on p. 3.)
- [ABL⁺13] Elena Andreeva, Andrey Bogdanov, Atul Luykx, Bart Mennink, Elmar Tischhauser, and Kan Yasuda. Parallelizable and authenticated online ciphers. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013*, volume 8269 of *LNCS*, pages 424–443. Springer, 2013. (Cited on p. 2.)
- [ABL⁺14] Elena Andreeva, Andrey Bogdanov, Atul Luykx, Bart Mennink, Nicky Mouha, and Kan Yasuda. How to securely release unverified plaintext in authenticated encryption. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014*, volume 8873 of *LNCS*, pages 105–125. Springer, 2014. (Cited on p. 2.)
- [ABV20] Elena Andreeva, Amit Singh Bhati, and Damian Vizár. Nonce-misuse security of the SAEF authenticated encryption mode. In Orr Dunkelman, Michael J. Jacobson Jr., and Colin O’Flynn, editors, *Selected Areas in Cryptography - SAC 2020 - 27th International Conference, Halifax, NS, Canada (Virtual Event), October 21-23, 2020, Revised Selected Papers*, volume 12804 of *Lecture Notes in Computer Science*, pages 512–534. Springer, 2020. (Cited on p. 2.)
- [ABV21] Elena Andreeva, Amit Singh Bhati, and Damian Vizár. RUP security of the SAEF authenticated encryption mode. *IACR Cryptol. ePrint Arch.*, page 103, 2021. (Cited on pp. 3 and 4.)
- [ADL17] Tomer Ashur, Orr Dunkelman, and Atul Luykx. Boosting authenticated encryption robustness with minimal modifications. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017*, volume 10403 of *LNCS*, pages 3–33. Springer, 2017. (Cited on p. 2.)
- [AFF⁺] Farzaneh Abed, Scott Fluhrer, John Foley, Christian Forler, Eik List, Stefan Lucks, David McGrew, and Jakob Wenzel. Poet v2.0. *Submission to CAESAR Competition*, 2013 - 2019. <https://competitions.cr.yp.to/round2/poetv20.pdf>. (Cited on p. 3.)
- [AFF⁺14] Farzaneh Abed, Scott R. Fluhrer, Christian Forler, Eik List, Stefan Lucks, David A. McGrew, and Jakob Wenzel. Pipelineable on-line encryption. In Carlos Cid and Christian Rechberger, editors, *FSE 2014*, volume 8540 of *LNCS*, pages 205–223. Springer, 2014. (Cited on pp. 2 and 3.)
- [ALP⁺19] Elena Andreeva, Virginie Lallemand, Antoon Purnal, Reza Reyhanitabar, Arnab Roy, and Damian Vizár. Forkcipher: A new primitive for authenticated encryption of very short messages. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019*, volume 11922 of *LNCS*, pages 153–182. Springer, 2019. (Cited on p. 3.)
- [AMPS22] Martin R. Albrecht, Lenka Mareková, Kenneth G. Paterson, and Igors Stepanovs. Four attacks and a proof for telegram. In *43rd IEEE Symposium on Security and Privacy, SP 2022, San Francisco, CA, USA, May 22-26, 2022*, pages 87–106. IEEE, 2022. (Cited on pp. 2 and 14.)

- [BBKN01] Mihir Bellare, Alexandra Boldyreva, Lars R. Knudsen, and Chanathip Namprempre. Online ciphers and the hash-cbc construction. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 292–309. Springer, 2001. (Cited on p. 2.)
- [Ber05] Daniel J. Bernstein. The poly1305-aes message-authentication code. In Henri Gilbert and Helena Handschuh, editors, *FSE 2005*, volume 3557 of *LNCS*, pages 32–49. Springer, 2005. (Cited on p. 5.)
- [BKN02] Mihir Bellare, Tadayoshi Kohno, and Chanathip Namprempre. Authenticated encryption in ssh: provably fixing the ssh binary packet protocol. In *CCS*, pages 1–11, 2002. (Cited on p. 2.)
- [BLLN21] Arghya Bhattacharjee, Cuauhtemoc Mancillas López, Eik List, and Mridul Nandi. The oribatida v1.3 family of lightweight authenticated encryption schemes. *J. Math. Cryptol.*, 15(1):305–344, 2021. (Cited on p. 3.)
- [BN16] Ritam Bhaumik and Mridul Nandi. Olef: an inverse-free online cipher. an online SPRP with an optimal inverse-free construction. *IACR Trans. Symmetric Cryptol.*, 2016(2):30–51, 2016. (Cited on pp. 3, 7, 8, 22, 23, and 24.)
- [BR00] Mihir Bellare and Phillip Rogaway. Encode-then-encipher encryption: How to exploit nonces or redundancy in plaintexts for efficient cryptography. In Tatsuaki Okamoto, editor, *ASIACRYPT 2000*, volume 1976 of *LNCS*, pages 317–330. Springer, 2000. (Cited on p. 3.)
- [BR04] Mihir Bellare and Phil Rogaway. The game-playing technique. 2004. (Cited on p. 5.)
- [BSJ⁺17] Mihir Bellare, Asha Camper Singh, Joseph Jaeger, Maya Nyayapati, and Iqors Stepanovs. Ratcheted encryption and key exchange: The security of messaging. In *CRYPTO 2017*, pages 619–650. Springer, 2017. (Cited on p. 16.)
- [BY03] Mihir Bellare and Bennet Yee. Forward-security in private-key cryptography. In *CT-RSA 2003*, pages 1–18. Springer, 2003. (Cited on p. 24.)
- [CAE19] CAESAR: Competition for Authenticated Encryption: Security, Applicability, and Robustness, 2013 - 2019. <https://competitions.cr.yt.to/caesar-submissions.html>. (Cited on p. 3.)
- [Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, pages 136–145. IEEE, 2001. (Cited on p. 2.)
- [CDD⁺19] Donghoon Chang, Nilanjan Datta, Avijit Dutta, Bart Mennink, Mridul Nandi, Somitra Sanadhya, and Ferdinand Sibleyras. Release of unverified plaintext: Tight unified model and application to ANYDAE. *IACR Trans. Symmetric Cryptol.*, 2019(4):119–146, 2019. (Cited on pp. 2 and 4.)
- [CDJ⁺19] Avik Chakraborti, Nilanjan Datta, Ashwin Jha, Cuauhtemoc Mancillas-López, Mridul Nandi, and Yu Sasaki. INT-RUP secure lightweight parallel AE modes. *IACR Trans. Symmetric Cryptol.*, 2019(4):81–118, 2019. (Cited on p. 3.)
- [CDJ⁺20] Avik Chakraborti, Nilanjan Datta, Ashwin Jha, Cuauhtemoc Mancillas-López, Mridul Nandi, and Yu Sasaki. ESTATE: A lightweight and low energy authenticated encryption mode. *IACR Trans. Symmetric Cryptol.*, 2020(S1):350–389, 2020. (Cited on p. 4.)

- [DDG22] Nilanjan Datta, Avijit Dutta, and Shibam Ghosh. INT-RUP security of SAEB and tinyjambu. In Takanori Isobe and Santanu Sarkar, editors, *INDOCRYPT 2022*, volume 13774 of *LNCS*, pages 146–170. Springer, 2022. (Cited on p. 3.)
- [DF18] Jean Paul Degabriele and Marc Fischlin. Simulatable channels: Extended security that is universally composable and easier to prove. In *ASIACRYPT 2018*, pages 519–550. Springer, 2018. (Cited on p. 2.)
- [DR11] Thai Duong and Juliano Rizzo. Here come the \oplus ninjas. 2011. (Cited on p. 2.)
- [FFL12] Ewan Fleischmann, Christian Forler, and Stefan Lucks. Mcoe: A family of almost foolproof on-line authenticated encryption schemes. In Anne Canteaut, editor, *FSE 2012*, volume 7549 of *LNCS*, pages 196–215. Springer, 2012. (Cited on p. 2.)
- [FGJ20] Marc Fischlin, Felix Günther, and Christian Janson. Robust channels: Handling unreliable networks in the record layers of quic and dtls 1.3. *Cryptology ePrint Archive*, Paper 2020/718, 2020. (Cited on p. 14.)
- [GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984. (Cited on p. 1.)
- [HC23] Munawar Hasan and Donghoon Chang. Lynx: Family of lightweight authenticated encryption schemes based on tweakable blockcipher. *IACR Cryptol. ePrint Arch.*, page 241, 2023. (Cited on p. 3.)
- [HDL21] Loïs Huguenin-Dumittan and Iraklis Leontiadis. A message franking channel. In *Inscrypt 2021*, pages 111–128. Springer, 2021. (Cited on p. 2.)
- [HRRV15] Viet Tung Hoang, Reza Reyhanitabar, Phillip Rogaway, and Damian Vizár. Online authenticated-encryption and its nonce-reuse misuse-resistance. In Rosario Gennaro and Matthew Robshaw, editors, *CRYPTO 2015*, volume 9215 of *LNCS*, pages 493–517. Springer, 2015. (Cited on pp. 1 and 2.)
- [IKMP20] Tetsu Iwata, Mustafa Khairallah, Kazuhiko Minematsu, and Thomas Peyrin. Duel of the titans: The romulus and remus families of lightweight AEAD algorithms. *IACR Trans. Symmetric Cryptol.*, 2020(1):43–120, 2020. (Cited on p. 3.)
- [KR11] Ted Krovetz and Phillip Rogaway. The software performance of authenticated-encryption modes. In Antoine Joux, editor, *FSE 2011*, volume 6733 of *LNCS*, pages 306–327. Springer, 2011. (Cited on p. 2.)
- [MP17] Giorgia Azzurra Marson and Bertram Poettering. Security notions for bidirectional channels. *IACR Trans. Symmetric Cryptol.*, 2017(1):405–426, 2017. (Cited on pp. 2 and 16.)
- [MV04] David A. McGrew and John Viega. The security and performance of the galois/counter mode (GCM) of operation. In Anne Canteaut and Kapalee Viswanathan, editors, *INDOCRYPT 2004*, volume 3348 of *LNCS*, pages 343–355. Springer, 2004. (Cited on p. 5.)
- [Nan15] Mridul Nandi. On the optimality of non-linear computations of length-preserving encryption schemes. *IACR Cryptology ePrint Archive*, 2015:414, 2015. (Cited on pp. 3 and 16.)

- [NIS18] NIST. 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>. (Cited on p. 3.)
- [NMS⁺] Yusuke Naito, Mitsuru Matsui, Yasuyuki Sakai, Daisuke Suzuki, Kazuo Sakiyama, and Takeshi Sugawara. Saeaes. *Submission to NIST Lightweight Cryptography Standardisation Process*, 2018 - 2023. <https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/spec-doc-rnd2/SAEAEs-spec-round2.pdf>. (Cited on p. 3.)
- [NMSS18] Yusuke Naito, Mitsuru Matsui, Takeshi Sugawara, and Daisuke Suzuki. SAEB: A lightweight blockcipher-based AEAD mode of operation. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(2):192–217, 2018. (Cited on p. 3.)
- [Rog02] Phillip Rogaway. Authenticated-encryption with associated-data. In Vijayalakshmi Atluri, editor, *Proceedings of the 9th ACM Conference on Computer and Communications Security, CCS 2002, Washington, DC, USA, November 18-22, 2002*, pages 98–107. ACM, 2002. (Cited on p. 1.)
- [Rog04] Phillip Rogaway. Nonce-based symmetric encryption. In Bimal K. Roy and Willi Meier, editors, *FSE 2004*, volume 3017 of *LNCS*, pages 348–359. Springer, 2004. (Cited on p. 1.)
- [RS06] Phillip Rogaway and Thomas Shrimpton. A provable-security treatment of the key-wrap problem. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 373–390. Springer, 2006. (Cited on p. 1.)
- [STA⁺a] Yu Sasaki, Yosuke Todo, Kazumaro Aoki, Yusuke Naito, Takeshi Sugawara, Yumiko Murakami, Mitsuru Matsui, and Shoichi Hirose. Minalpher. *DIAC 2014: Directions in Authenticated Ciphers*, August 23-24, 2014, Santa Barbara, USA. <https://info.isl.ntt.co.jp/crypt/minalpher/files/minalpher-diac2014.pdf>. (Cited on p. 3.)
- [STA⁺b] Yu Sasaki, Yosuke Todo, Kazumaro Aoki, Yusuke Naito, Takeshi Sugawara, Yumiko Murakami, Mitsuru Matsui, and Shoichi Hirose. Minalpher v1.1. *Submission to CAESAR Competition*, 2013 - 2019. <https://competitions.cr.yt.to/round2/minalpherv11.pdf>. (Cited on p. 3.)
- [WH] Hongjun Wu and Tao Huang. Tinyjambu v2. *Submission to NIST Lightweight Cryptography Standardisation Process*, 2018 - 2023. (Cited on p. 3.)
- [YVCC22] Hailun Yan, Serge Vaudenay, Daniel Collins, and Andrea Caforio. Optimal symmetric ratcheting for secure communication. *Computer Journal*, 2022. (Cited on p. 24.)
- [ZW16] Jian Zhang and Wen-Ling Wu. Security of online AE schemes in RUP setting. In Sara Foresti and Giuseppe Persiano, editors, *CANS 2016*, volume 10052 of *LNCS*, pages 319–334, 2016. (Cited on p. 2.)

Appendices

A Proof Sketch of **Theorem 1**

We first use the standard hybrid reduction to replace E_K with a random function $f^* : \mathbb{B} \rightarrow \mathbb{B}$, by allowing \mathcal{A}' to play a PRF game against E_K using \mathcal{A} ; since there are at most $10\sigma/3$ calls to E_K in the game (worst case: 10 for a 3-block query with a tiny 4-th block), this accounts for the term $\mathbf{Adv}_E^{\text{prf}}(\mathcal{A}')$ in the advantage. This brings us to the setting of [BN16, Theorem 2], and we can follow the same proof with a few changes.

One can observe that the first message block is processed differently in OleF and t-OleF, while the processing of the rest of the message blocks are similar. As a result, **badB** of OleF requires dedicated analysis in t-OleF whenever the first message block is involved.

We consider the tweaks $T^i (i \in [q])$ as part of the transcript. We also take the tweaks into account when we compute prefixes (thus affecting in particular the definitions of \mathcal{F} , \mathcal{F}' , **badA**, and **badB**). Here we need to consider some additional bad events. We follow the notation of the proof of [BN16, Theorem 2]. As all of the additional bad events fall into the category **badB** of [BN16, Theorem 2], we state all the bad events from that category below, and bound their probabilities. The rest of the security analysis of OleF and t-OleF are identical, and we omit that to avoid tedious repetition.

Instead of sampling the basis elements suggested in [BN16], we sample directly the outputs corresponding to \mathcal{D} (as defined in [BN16, Sec. 4.1]): we call the outputs corresponding to $\mathbb{L}^i[j]$ and $\mathbb{R}^i[j]$ respectively $\widehat{\mathbb{L}}^i[j]$ and $\widehat{\mathbb{R}}^i[j]$.

1. **badB1**: For some $(i_1, j_1), (i_2, j_2) \in \mathcal{F}$, with $(i_1, j_1) \neq (i_2, j_2)$, $\mathbb{X}^{i_1}[j_1] = \mathbb{X}^{i_2}[j_2]$.
 - (a) **badB11**: $j_1 > 1, j_2 > 1$. This event is a sub-event of the first event under the category **badB** of [BN16, Theorem 2].
 - (b) **badB12**: $j_1 > 1, j_2 = 1$. Expanding the expressions for the random variables, we obtain

$$\widehat{\mathbb{L}}^{i_1}[j_1] + \mathbb{R}^{i_1}[j_1] + b_2 \cdot \mathbb{T}^{i_1}[j_1] = \widehat{\mathbb{L}}^{i_2}[1] + \mathbb{R}^{i_2}[1] + b_2 \cdot \mathcal{H}(T^{i_2}).$$

For $\mathcal{H}(T^{i_2}) = 0$, this event is a sub-event of the first event under the category **badB** of [BN16, Theorem 2]. And even if $\mathcal{H}(T^{i_2}) \neq 0$, the same bound holds as all the required randomness are still present in the event.

- (c) **badB13**: $j_1 = j_2 = 1$. Expanding the expressions for the random variables, we obtain

$$\widehat{\mathbb{L}}^{i_1}[1] + \mathbb{R}^{i_1}[1] + b_2 \cdot \mathcal{H}(T^{i_1}) = \widehat{\mathbb{L}}^{i_2}[1] + \mathbb{R}^{i_2}[1] + b_2 \cdot \mathcal{H}(T^{i_2}).$$

If $T^{i_1} \neq T^{i_2}$, then for a fixed value of the indices, the probability of the event is upper bounded by ϵ as \mathcal{H} is ϵ -AXU. If $T^{i_1} = T^{i_2}$ but $\mathbb{L}^{i_1}[1] \neq \mathbb{L}^{i_2}[1]$, then for a fixed value of the indices, the probability of the event is upper bounded by $1/N$ due to the randomness of $\widehat{\mathbb{L}}^{i_1}[1]$ or $\widehat{\mathbb{L}}^{i_2}[1]$. And if $T^{i_1} = T^{i_2}$ and $\mathbb{L}^{i_1}[1] = \mathbb{L}^{i_2}[1]$ but $\mathbb{R}^{i_1}[1] \neq \mathbb{R}^{i_2}[1]$, then for a fixed value of the indices, the probability of the event is 0. Applying union bound over all possible values of the indices, we obtain

$$\Pr[\text{badB13}] \leq q^2 \epsilon + \frac{q^2}{N}.$$

2. **badB2**: For some $(i_1, j_1), (i_2, j_2) \in \mathcal{F}'$, with $(i_1, j_1) \neq (i_2, j_2)$, $\mathbb{Y}^{i_1}[j_1] = \mathbb{Y}^{i_2}[j_2]$.

- (a) **badB21**: $j_1 > 1, j_2 > 1$. This event is a sub-event of the second event under the category **badB** of [BN16, Theorem 2].
- (b) **badB22**: $j_1 > 1, j_2 = 1$. The probability of this event can be bounded following similar argument to **badB12**.
- (c) **badB23**: $j_1 = j_2 = 1$. Expanding the expressions for the random variables, we obtain

$$L^{i_1}[1] + b_1 \cdot \widehat{R}^{i_1}[1] + b_2 \cdot \mathcal{H}(T^{i_1}) = L^{i_2}[1] + b_1 \cdot \widehat{R}^{i_2}[1] + b_2 \cdot \mathcal{H}(T^{i_2}).$$

If $T^{i_1} \neq T^{i_2}$, then for a fixed value of the indices, the probability of the event is upper bounded by ϵ as \mathcal{H} is ϵ -AXU. If $T^{i_1} = T^{i_2}$ but $R^{i_1}[1] \neq R^{i_2}[1]$, then for a fixed value of the indices, the probability of the event is upper bounded by $1/N$ due to the randomness of $\widehat{R}^{i_1}[1]$ or $\widehat{R}^{i_2}[1]$. And if $T^{i_1} = T^{i_2}$ and $R^{i_1}[1] = R^{i_2}[1]$ but $L^{i_1}[1] \neq L^{i_2}[1]$, then for a fixed value of the indices, the probability of the event is 0. Applying union bound over all possible values of the indices, we obtain

$$\Pr[\text{badB23}] \leq q^2 \epsilon + \frac{q^2}{N}.$$

3. **badB3**: For some $(i_1, j_1) \in \mathcal{F}, (i_2, j_2) \in \mathcal{F}', X^{i_1}[j_1] = Y^{i_2}[j_2]$.

- (a) **badB31**: $j_1 > 1, j_2 > 1$. This event is a sub-event of the third event under the category **badB** of [BN16, Theorem 2].
- (b) **badB32**: $j_1 = 1, j_2 > 1$. The probability of this event can be bounded following similar argument to **badB12**.
- (c) **badB33**: $j_1 > 1, j_2 = 1$. The probability of this event can be bounded following similar argument to **badB12**.
- (d) **badB34**: $j_1 = j_2 = 1$. Expanding the expressions for the random variables, we obtain

$$\widehat{L}^{i_1}[1] + R^{i_1}[1] + b_2 \cdot \mathcal{H}(T^{i_1}) = L^{i_2}[1] + b_1 \cdot \widehat{R}^{i_2}[1] + b_2 \cdot \mathcal{H}(T^{i_2}).$$

If $T^{i_1} \neq T^{i_2}$, then for a fixed value of the indices, the probability of the event is upper bounded by ϵ as \mathcal{H} is ϵ -AXU. If $T^{i_1} = T^{i_2}$ but $L^{i_1}[1] \neq R^{i_2}[1]$, then for a fixed value of the indices, the probability of the event is upper bounded by $1/N$ due to the randomness of $\widehat{L}^{i_1}[1]$ or $\widehat{R}^{i_2}[1]$. Even if $T^{i_1} = T^{i_2}$ and $L^{i_1}[1] = R^{i_2}[1]$, for a fixed value of the indices, the probability of the event is upper bounded by $1/N$ because we can still use the randomness of $\widehat{L}^{i_1}[1]$ as it doesn't cancel out in the equation. Applying union bound over all possible values of the indices, we obtain

$$\Pr[\text{badB34}] \leq q^2 \epsilon + \frac{q^2}{N}.$$

4. **badB4**: For some $(i, j) \in \mathcal{F}, X^i[j] \in \mathcal{D}$.

- (a) **badB41**: $j > 1$. This event is a sub-event of the fourth event under the category **badB** of [BN16, Theorem 2].
- (b) **badB42**: $j = 1$. The probability of this event can be bounded following similar argument to **badB12**.

5. **badB5**: For some $(i, j) \in \mathcal{F}'$, $Y^i[j] \in \mathcal{D}$.

- (a) **badB51**: $j > 1$. This event is a sub-event of the fifth event under the category **badB** of [BN16, Theorem 2].
- (b) **badB52**: $j = 1$. The probability of this event can be bounded following similar argument to **badB12**.

Of the probability terms above, the only terms not subsumed in the bound of **OleF** are the three $q^2\epsilon$ terms in the bounds for $\Pr[\text{badB13}]$, $\Pr[\text{badB23}]$, and $\Pr[\text{badB34}]$, which add up to an additional term of $3q^2\epsilon$ in the final bound.

B Deferred Material on Online Channels

B.1 Proof of **Theorem 4**

Proof. Let G be the online channels security game played w.r.t. \mathcal{Ch} except where the output of Receive_1 with target expansion τ^* is always \perp . By a standard bad-event hybrid argument, we can first hop from the original game to G by excluding the event where Receive_1 outputs $(m, i) \neq \perp$ by a reduction to the integrity of oaE . In particular, \mathcal{A}' queries Enc to simulate Send , Dec to simulate Leak and Ver to partially simulate Receive (note the game aborts before Receive_1 ever outputs a non- \perp value) where the third argument x/y to $\text{Enc}/\text{Dec}/\text{Ver}$ comprise of L blocks or less. \mathcal{A}' otherwise simulates locally, and in particular keeps track of Reset queries to know what indices to simulate Send queries. In the worst case, \mathcal{A}' (and \mathcal{A}'' below) makes at most σ oracle queries (but could make as few as $\lceil \sigma/L \rceil$ if \mathcal{A} makes e.g. a single long Send query). Note in general that more than one oaE forgery (i.e., non- \perp output from Dec) may be required to break \mathcal{Ch} (i.e., non- \perp output from Receive), e.g. if the forged ciphertext consists of several L -block batches. Consequently, the advantage loss is at most $\text{Adv}_{\text{oaE}}^{\text{int}}(\mathcal{A}')$ for efficient \mathcal{A}' . It is then straightforward for oaE privacy adversary \mathcal{A}'' to simulate perfectly for a G adversary, in particular simulating Send calls via Enc , Receive calls by outputting \perp , Leak via Dec , and Reset locally. \square \square

B.2 Extensions and Variants

Our formalism in **Section 5** could be conceivably extended to support *forward security* [BY03, YVCC22]. That is, if the secret key is exposed (i.e., given to the adversary) after i e.g. encryption queries, then the key cannot be used to decrypt the first i ciphertexts. This attack vector is especially important for long-lived communication sessions. A two key (rather than our one key) construction would be more suited to this purpose for handling concurrency/asynchrony, where each party encrypts using a different key. Here, the relevant key would be hashed (using an online PRP, for instance) after every encryption.

Our security notion captures the particular case where, after every operation, the state of both parties is reset/cleared, corresponding to the scenario where their states are never updated. Our formalism can be adapted and simplified to this ‘stateless’ setting, since, e.g., we no longer require good-case sequentiality and indices output by $\text{send}/\text{receive}$ are not so meaningful in this case.

Our security game (and security notions in the previous sections) are such that Send and Receive queries are one-shot, i.e., that they encrypt/decrypt the *entire* message at once. It would therefore be more general to allow for *per-block*, rather than *per-message* oracle queries.