Breaking the confidentiality of OCB2

Bertram Poettering
Royal Holloway, University of London
`bertram.poettering@rhul.ac.uk`

2018-Nov-12

Abstract. OCB2 is a widely standardized mode of operation of a blockcipher that aims at providing authenticated encryption. A recent report by Inoue and Minematsu (IACR EPRINT report 2018/1040) indicates that OCB2 does not meet this goal. Concretely, by describing simple forging attacks the authors evidence that the (sub)goal of authenticity is not reached. The report does not question the confidentiality offered by OCB2.

In this note we show how the attacks of Inoue and Minematsu can be extended to also break the confidentiality of OCB2. We do this by constructing both IND-CCA and plaintext recovering adversaries, all of which require minimal resources and achieve overwhelming success rates.

1 Introduction

In symmetric-key cryptography, a primitive providing authenticated encryption (AE) is one that allows for encrypting messages into ciphertexts, and decrypting ciphertexts into messages, such that both the confidentiality and the integrity of the messages are protected. A classic approach towards achieving this is through the hybrid encrypt-then-mac construction [1], but a line of research that started about two decades ago [4] and is now more active than ever [9] put forward several integrated AE modes that jointly achieve the two security goals in a more efficient way. As of today, authenticated encryption (possibly enriched with the option to take into account an associated-data string when performing the encryption and decryption operations; this variant is commonly referred to as AEAD) is a core component of many real-world cryptographic constructions.

While AES-GCM [5] has likely been the most widely used AE scheme for the last decade, a family of independent constructions is known by the name of OCB [6]. The three members of this family (OCB1, OCB2, OCB3) are blockcipher-based designs and effectively get along with a single blockcipher invocation per message block ('rate-1'). While OCB1 is a plain AE mode, OCB2 and OCB3 are AEAD modes, and all three modes are among the most efficient (generic blockcipher based) designs that the market has to offer.[1]

In their recent report [3], Inoue and Minematsu (IM) showed that the OCB2 authenticated encryption scheme does not achieve the promised goal of authenticity. More precisely, the authors give four different attacks on OCB2 that allow for forging ciphertexts that validly decrypt to unauthentic messages without flagging an error. All four attacks succeed with overwhelming probability, they require minimal time and memory resources, and some of them allow for a specific level of control over the message to which a forged ciphertext will decrypt. The only realistic conclusion that one can draw from this seems to be that the authenticity of OCB2 is fully broken.[2]

Contribution. In this light it seems natural to ask whether the IM attacks on *authenticity* also have implications on the *confidentiality* of OCB2. In the abstract of their report, IM indicate that their attacks "do not break the privacy of OCB2" [3]. In the current article we thus provide a fresh IM-inspired assessment of the confidentiality of OCB2, and our main result is that the findings of IM indeed *can* be leveraged to yield effective message distinguishing attacks (in the IND-CCA sense) as well as

[1] All versions of OCB were, and some still are, covered by intellectual property claims. This likely contributes to the clear real-world dominance of (the royalty-free) AES-GCM. The performance of AES-GCM could catch up with that of OCB only when CPU manufacturers started incorporating hardware support for certain GCM operations into their products (e.g., the `PCLMULQDQ` instruction in Westmere).

[2] As IM point out, it seems that fixing OCB2 is not too complicated. However, as ciphertexts of the original and the fixed version are not compatible with each other, implementers likely will, instead of applying the fix, take the opportunity to switch to AES-GCM in the first place.

powerful plaintext recovery attacks. We show this by giving concrete attacks that, just like the ones of IM, consume minimal resources and have an overwhelming success rate. As a by-product we further show that OCB2 is indeed universally forgeable. Our conclusion is that the confidentiality of OCB2 is as broken as its authenticity.

2 Preliminaries

2.1 Notation

Symbols. If A is a set we write $a \leftarrow_\$ A$ for the operation of picking an element of A uniformly at random and assigning it to the variable a. If B, B' are sets we write $B \overset{\cup}{\leftarrow} B'$ shorthand for $B \leftarrow B \cup B'$. For fixed-length strings $C_1, C_2 \in \{0,1\}^n$ we write $C \leftarrow C_1 \parallel C_2$ for their concatenation to a single ($2n$-long) string C. Such a string C can again be split up into its components C_1, C_2 with the $\overset{n}{\leftarrow}$ operator. For example $C \leftarrow C_1 \parallel C_2 \parallel C_3$ followed by $C_1' \parallel C_2' \parallel C_3' \overset{n}{\leftarrow} C$ yields $C_1' = C_1$, $C_2' = C_2$, $C_3' = C_3$. As further detailed in Section 2.3, we write $+$ for the xor operation. In program code we might use the if-then-else ternary operator `_ ? _ : _` known from programming languages like C and Java: If C is a Boolean condition, the expression C `?` a `:` b evaluates to a if C is true; otherwise, it evaluates to b.

Games. We define security notions via games played between a challenger and the adversary. Such games are formalized with pseudo-code. The execution of a game stops when it runs into a 'Stop' instruction. If the latter has an argument (e.g., 'Stop with x'), then the argument is considered the output of the game. For a game G we write $\Pr[G \Rightarrow 1]$ for the probability (over all random coins drawn by the game and the adversary) that the game terminates by running into a 'Stop with x' instruction with $x = 1$. We further use the instruction 'Require C', where C is a Boolean condition, as a shortcut for 'If $\neg C$: Stop with 0'. (This is usually used to penalize the adversary for posing 'illegal queries'; note that all our security definitions are such that such a penalty does not increase the formal attack advantage.)

2.2 Nonce-based AEAD

Syntax. We formalize a syntactical framework for authenticated encryption with associated data (AEAD). A corresponding scheme specifies a key space $\mathcal{K}$, a nonce space $\mathcal{N}$, an associated-data space $\mathcal{AD}$, a message space $\mathcal{M}$, a ciphertext space $\mathcal{C}$, and the (deterministic) algorithms enc and dec. The encryption algorithm enc takes a key $K \in \mathcal{K}$, a nonce $N \in \mathcal{N}$, an associated-data string $AD \in \mathcal{AD}$, and a message $M \in \mathcal{M}$, and outputs a ciphertext $C \in \mathcal{C}$. The decryption algorithm dec takes a key $K \in \mathcal{K}$, a nonce $N \in \mathcal{N}$, an associated-data string $AD \in \mathcal{AD}$, and a ciphertext $C \in \mathcal{C}$, and outputs either a message $M \in \mathcal{M}$ or the special symbol $\perp \notin \mathcal{M}$. If dec outputs a message, i.e., an element of $\mathcal{M}$, then we say it *accepts* (the ciphertext C); otherwise, if it outputs $\perp$, we say it *rejects*. For correctness we require that if keys, nonces, and associated data are provided consistently to enc and dec, then messages encrypted with enc are recovered by dec. Precisely, we require that for all $K \in \mathcal{K}, N \in \mathcal{N}, AD \in \mathcal{AD}, M \in \mathcal{M}$ we have $\mathsf{enc}(K, N, AD, M) = C \Rightarrow \mathsf{dec}(K, N, AD, C) = M$.

Security. We consider two security notions for AEAD: one for authenticity and one for confidentiality. While articles that aim at establishing the *security* of an AEAD candidate tend to do so using rather strong notions (e.g., in the IND\$ or SUF spirit), in this article we aim at analyzing the *insecurity* of an AEAD scheme and thus deliberately focus on rather weak notions. This only strengthens our results: If a scheme does not meet a weak notion, in particular it also does not meet any stronger notion. Note that the security goals that we formalize below assume nonce-respecting adversaries (that use for each encryption query a fresh nonce).

Our authenticity notion is formalized using the INT game from Figure 1 (left); the focus is on the integrity protection of associated-data strings and messages. (In contrast to [3] our definition disregards attacks that merely consist of manipulating nonces or ciphertexts.) We define the authenticity advantage of an adversary A as per $\mathbf{Adv}^{\text{int}}(\mathsf{A}) := \Pr[\text{INT}(\mathsf{A}) \Rightarrow 1]$. Intuitively, an AEAD scheme offers authenticity if $\mathbf{Adv}^{\text{int}}(\mathsf{A})$ is negligible for all realistic adversaries A.

Our confidentiality notion is formalized using the IND^b games from Figure 1 (right); note that this is a classic left-or-right IND-CCA definition and thus captures a notion of confidentiality against active adversaries. (In contrast to [3] our definition does not require that ciphertexts look like random bitstrings.) We define the confidentiality advantage of an adversary B as per $\mathbf{Adv}^{\mathrm{ind}}(\mathsf{B}) := |\Pr[\mathrm{IND}^1(\mathsf{B}) \Rightarrow 1] - \Pr[\mathrm{IND}^0(\mathsf{B}) \Rightarrow 1]|$. Intuitively, an AEAD scheme offers confidentiality (against active attacks) if $\mathbf{Adv}^{\mathrm{ind}}(\mathsf{B})$ is negligible for all realistic adversaries B.

```
**Game** INT(A)
00 $\mathrm{NS} \leftarrow \emptyset$; $\mathrm{ADM} \leftarrow \emptyset$
01 $K \leftarrow_{\$} \mathcal{K}$
02 $\mathsf{A}^{\mathcal{E}(\cdot,\cdot,\cdot),\mathcal{D}(\cdot,\cdot,\cdot)}$
03 Stop with 0

**Oracle** $\mathcal{E}(N, AD, M)$
04 Require $N \notin \mathrm{NS}$
05 $C \leftarrow \mathsf{enc}(K, N, AD, M)$
06 $\mathrm{NS} \xleftarrow{\cup} \{N\}$
07 $\mathrm{ADM} \xleftarrow{\cup} \{(AD, M)\}$
08 Return $C$

**Oracle** $\mathcal{D}(N, AD, C)$
09 $M \leftarrow \mathsf{dec}(K, N, AD, C)$
10 If $M = \perp$: Return $\perp$
11 If $(AD, M) \notin \mathrm{ADM}$:
12     Stop with 1
13 Return $M$
```

```
**Game** IND^b(B)
14 $\mathrm{NS} \leftarrow \emptyset$; $\mathrm{NADC} \leftarrow \emptyset$
15 $K \leftarrow_{\$} \mathcal{K}$
16 $b' \leftarrow \mathsf{B}^{\mathcal{E}(\cdot,\cdot,\cdot,\cdot),\mathcal{D}(\cdot,\cdot,\cdot)}$
17 Stop with $b'$

**Oracle** $\mathcal{E}(N, AD, M^0, M^1)$
18 Require $N \notin \mathrm{NS}$
19 Require $|M^0| = |M^1|$
20 $C \leftarrow \mathsf{enc}(K, N, AD, M^b)$
21 $\mathrm{NS} \xleftarrow{\cup} \{N\}$
22 $\mathrm{NADC} \xleftarrow{\cup} \{(N, AD, C)\}$
23 Return $C$

**Oracle** $\mathcal{D}(N, AD, C)$
24 Require $(N, AD, C) \notin \mathrm{NADC}$
25 $M \leftarrow \mathsf{dec}(K, N, AD, C)$
26 If $M = \perp$: Return $\perp$
27 Return $M$
```

Fig. 1. Games $\mathrm{INT}, \mathrm{IND}^0, \mathrm{IND}^1$ for modeling integrity (of messages) and indistinguishability (of messages, under chosen-ciphertext attacks). Note how lines 04,06,18,21 enforce that the adversary be nonce-respecting.

2.3 Blockciphers that operate on finite fields

Blockciphers. For a key space $\mathcal{K}$ and a block length n, a blockcipher is a pair of functions $E, D \colon \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ such that for all $K \in \mathcal{K}$ and $X, Y \in \{0,1\}^n$ we have $D(K, E(K, X)) = X$ or, equivalently, $E(K, D(K, Y)) = Y$.

Finite Fields. The domain $\{0,1\}^n$ of a blockcipher can be identified with the set of elements of the finite field $\mathrm{GF}(2^n)$. More precisely, after fixing an irreducible degree-n polynomial $P \in \mathrm{GF}(2)[\mathtt{x}]$ (such a polynomial exists for all n), the elements of the field $\mathrm{GF}(2^n) := \mathrm{GF}(2)[\mathtt{x}]/(P)$ have a canonic representation as bitstrings of length n. We write $+$ and $\cdot$ for the field operations (where $+$ coincides with the xor operation). In the context of OCB2, the reduction polynomial P is chosen such that the field element $\mathtt{x}$ (the degree-1 monomial) is primitive in $\mathrm{GF}(2^n)$, that is, the sequence $\mathtt{x}^1, \mathtt{x}^2, \mathtt{x}^3, \ldots$ ranges over $2^n - 1$ different values.

2.4 Specification of OCB2

We reproduce details of the OCB2 nonce-based AEAD scheme from [7,8]. The scheme is based on a blockcipher (typically AES) and parameterized by a tag length τ (which kind of serves as a security parameter). In the following we actually do not give the full specification of OCB2; rather, in order to simplify the exposition, we remove some of its functionality (see upcoming paragraph). Note that any attack that is successful against the restricted scheme also applies to the full scheme. This holds in particular for the Inoue–Minematsu authenticity attacks from [3] as well as for the attack on confidentiality presented in the current article.

Assume a blockcipher (E, D) with key space $\mathcal{K}$ and block length n, and understand the cipher's domain $\{0,1\}^n$ as representing the elements of a finite field as suggested in Section 2.3. Then the algorithms of OCB2 are specified in Figure 2. The scheme has key space $\mathcal{K}$, nonce space $\{0,1\}^n$, and uses $\{0,1\}^*$ as associated-data space, message space, and ciphertext space. As announced above, our specification of OCB2 only covers a specific sub-case, namely the one where (a) the tag length coincides with the cipher's block length (this allows for disregarding the tag truncation step), (b) the associated-data input is always the empty string (this allows for removing the description of the auxiliary PMAC component), and (c) the length of considered messages and ciphertexts is always a multiple of the cipher's block length (this allows for neglecting padding operations). Note that the specifications of the enc and dec algorithms assume a length-encoding function len: $\{0,1\}^{\leq n} \to \{0,1\}^n$.[3]

```
**Algorithm** $\mathsf{enc}_\tau(K, N, AD, M)$
00 Require $\tau = n \wedge AD = \epsilon \wedge n \mid |M|$
01 $L \leftarrow E(N)$
02 $M[1] \parallel \ldots \parallel M[m] \overset{n}{\leftarrow} M$
03 For $i \leftarrow 1$ to $m-1$:
04     $C[i] \leftarrow \mathtt{x}^i L + E(\mathtt{x}^i L + M[i])$
05 $C[m] \leftarrow M[m] + E(\mathtt{x}^m L + \mathrm{len}(0^n))$
06 $\Sigma \leftarrow M[1] + \ldots + M[m]$
07 $T \leftarrow E(\mathtt{x}^m(\mathtt{x}+1)L + \Sigma)$
08 $C \leftarrow C[1] \parallel \ldots \parallel C[m] \parallel T$
09 Return $C$
```

```
**Algorithm** $\mathsf{dec}_\tau(K, N, AD, C)$
10 Require $\tau = n \wedge AD = \epsilon \wedge n \mid |C|$
11 $L \leftarrow E(N)$
12 $C[1] \parallel \ldots \parallel C[m] \parallel T \overset{n}{\leftarrow} C$
13 For $i \leftarrow 1$ to $m-1$:
14     $M[i] \leftarrow \mathtt{x}^i L + D(\mathtt{x}^i L + C[i])$
15 $M[m] \leftarrow C[m] + E(\mathtt{x}^m L + \mathrm{len}(0^n))$
16 $\Sigma \leftarrow M[1] + \ldots + M[m]$
17 $T^* \leftarrow E(\mathtt{x}^m(\mathtt{x}+1)L + \Sigma)$
18 $M \leftarrow M[1] \parallel \ldots \parallel M[m]$
19 Return $(T = T^*)$ ? $M$ : $\perp$
```

Fig. 2. Specification of OCB2 (for the sub-case enforced by lines 00,10). For compactness we abbreviate $E(K,\cdot)$ with $E(\cdot)$ and $D(K,\cdot)$ with $D(\cdot)$.

3 Attacks on OCB2

We first recall one of the recent attacks on OCB2's authenticity by Inoue and Minematsu (IM) (Section 3.1) and then show how the attack can be adapted to instead break the scheme's confidentiality (Section 3.2). We intentionally design the corresponding IND-CCA adversary as minimalistic as possible, and the result leaves open whether also a plaintext recovery attack is feasible. Before we clarify on the latter, in Section 3.3 we take a step back and develop a general OCB2-specific attack framework. The techniques formalized in this section clearly borrow from those of Sections 3.1 and 3.2, and indeed we see its contribution mainly in abstracting and concentrating our understanding of the OCB2 vulnerabilities into two easy-to-use procedures. Finally, in Sections 3.4 and 3.5, we describe a universal forgery attack and a couple of plaintext recovery attacks. These attacks are based on the abstractions developed in Section 3.3 and can thus be described rather informally, in particular without touching any of the technical details of OCB2.

3.1 Recap: Inoue–Minematsu attack on authenticity

We reproduce the most simple attack on the authenticity of OCB2 from [3]. The attack gets along with a single encryption query and succeeds with finding a forgery with probability 1. (The delivery of the forgery requires, of course, an additional decryption query.) The details of a corresponding adversary A for the INT game from Figure 1 (left) are in Figure 3 (left). We trace the values of some variables

[3] The details of is function are specified in the OCB2 standard, but they are not relevant for our analysis.

throughout an execution of the adversary within the game:

$$
\begin{aligned}
M[1] &= \mathrm{len}(0^n)\\
L_0 &= E(N_0)\\
C[1] &= \mathtt{x}^1 L_0 + E(\mathtt{x}^1 L_0 + M[1]) = \mathtt{x}^1 L_0 + E(\mathtt{x}^1 L_0 + \mathrm{len}(0^n))\\
C[2] &= M[2] + E(\mathtt{x}^2 L_0 + \mathrm{len}(0^n))\\
\varSigma &= M[1] + M[2] = \mathrm{len}(0^n) + M[2]\\
T &= E(\mathtt{x}^2(\mathtt{x}+1)L_0 + \varSigma) = E(\mathtt{x}^2(\mathtt{x}+1)L_0 + \mathrm{len}(0^n) + M[2])\\
C'[1] &= C[1] + \mathrm{len}(0^n) = \mathtt{x}^1 L_0 + E(\mathtt{x}^1 L_0 + \mathrm{len}(0^n)) + \mathrm{len}(0^n)\\
T' &= M[2] + C[2] = M[2] + M[2] + E(\mathtt{x}^2 L_0 + \mathrm{len}(0^n)) = E(\mathtt{x}^2 L_0 + \mathrm{len}(0^n))\\
M'[1] &= C'[1] + E(\mathtt{x}^1 L_0 + \mathrm{len}(0^n)) = \mathtt{x}^1 L_0 + \mathrm{len}(0^n)\\
\varSigma' &= M'[1] = \mathtt{x}^1 L_0 + \mathrm{len}(0^n)\\
T^* &= E(\mathtt{x}^1(\mathtt{x}+1)L_0 + \varSigma') = E(\mathtt{x}^1(\mathtt{x}+1)L_0 + \mathtt{x}^1 L_0 + \mathrm{len}(0^n)) = E(\mathtt{x}^2 L_0 + \mathrm{len}(0^n))\\
M' &= M'[1] = \mathtt{x}^1 L_0 + \mathrm{len}(0^n)
\end{aligned}
$$

Note that by $T' = T^*$ the decryption oracle accepts ciphertext C' and returns the message $M' = \mathtt{x}^1 L_0 + \mathrm{len}(0^n)$. As M' and M have different lengths we in particular have $M' \neq M$ and the forgery counts. Thus $\mathbf{Adv}^{\mathrm{int}}(\mathsf{A}) = 1$, i.e., the adversary breaks authenticity with probability 1.

```
Adversary $\mathsf{A}^{\mathcal{E}(\cdot,\cdot,\cdot),\mathcal{D}(\cdot,\cdot,\cdot)}$
00 Step 1:
01     $M[1] \leftarrow \mathrm{len}(0^n)$
02     Pick any $M[2] \in \{0,1\}^n$
03     $M \leftarrow M[1] \parallel M[2]$
04     Pick any $N_0 \in \{0,1\}^n$
05     Query $C \leftarrow \mathcal{E}(N_0, \epsilon, M)$
06 Step 2:
07     $C[1] \parallel C[2] \parallel T \xleftarrow{n} C$
08     $C'[1] \leftarrow C[1] + \mathrm{len}(0^n)$
09     $T' \leftarrow M[2] + C[2]$
10     $C' \leftarrow C'[1] \parallel T'$
11     Query $M' \leftarrow \mathcal{D}(N_0, \epsilon, C')$
12 Stop
```

```
Adversary $\mathsf{B}^{\mathcal{E}(\cdot,\cdot,\cdot,\cdot),\mathcal{D}(\cdot,\cdot,\cdot)}$
13 Steps 1+2 as in the IM attack,
14     obtaining $M' = \mathtt{x}^1 L_0 + \mathrm{len}(0^n)$
15 Step 3:
16     $L_0 \leftarrow \mathtt{x}^{-1}(M' + \mathrm{len}(0^n))$
17     Find pairs $(X_i, Y_i) \in E(K, \cdot)$
18     Pick $(N_1, L_1) \in E(K, \cdot)$ s.t. $N_1 \neq N_0$
19     $M^0[1] \leftarrow \mathtt{x}^1(\mathtt{x}+1)L_1 + N_0$
20     Pick any $M^1[1] \in \{0,1\}^n \setminus \{M^0[1]\}$
21     $M^0 \leftarrow M^0[1]$; $M^1 \leftarrow M^1[1]$
22     Query $C'' \leftarrow \mathcal{E}(N_1, \epsilon, M^0, M^1)$
23     $C''[1] \parallel T'' \xleftarrow{n} C''$
24     $b' \leftarrow (T'' = L_0)\ ?\ 0 : 1$
25 Stop with $b'$
```

Fig. 3. Left: IM attack on authenticity [3, Sec. 4.1]. **Right:** Our new attack on confidentiality. See text for the meaning of lines 17,18.

We note that IM propose in their report a total of four different attacks on the authenticity of OCB2, and in Figure 3 we reproduce just the most basic one. This version is actually a special case of a more general attack that uses arbitrary-length messages. Also the more general attack is by IM [3], and for completeness we reproduce it in Appendix A.

3.2 A basic distinguishing attack on confidentiality

The IM attack from Figure 3 (left) breaks OCB2 by coming up with an unauthentic yet valid ciphertext C'. Perhaps surprisingly, the message M' corresponding to this ciphertext does not play a role in the attack; it is just discarded (line 11). In the following we show how the release of M' actually allows for conducting an attack on the confidentiality of OCB2. More precisely, after first emulating the steps of the IM attack to come up with ciphertext C', our confidentiality attacker uses the corresponding message M' to craft two challenge messages M^0, M^1 that can be distinguished within our left-or-right style security definition. In brief, the idea is to deduce from M' (and other public values) a set of 'raw' input-output pairs of $E(K,\cdot)$.[4] (Normal operation of OCB2 would discard unauthentic ciphertexts with

[4] A step like this also appears in IM's 'Almost Universal Forgery, Variant 1' attack [3, Sec 4.3].

the consequence that such pairs would remain hidden.) From these input-output pairs a fresh nonce N_1 and a message M^0 are derived such that at least one of the internal blockcipher invocations of operation $\mathsf{enc}(K, N_1, \epsilon, M^0)$ is on one of the known input values. This turns out to be sufficient for a distinguishing attack.

In Figure 3 (right) we provide the details of an adversary B for the IND^b games from Figure 1 (right). Attack steps 1 and 2 are just the ones from Figure 3 (left), where of course the original $\mathcal{E}(N_0, \epsilon, M)$ query (line 05) has to be replaced by the equivalent $\mathcal{E}(N_0, \epsilon, M, M)$ query. In particular adversary B obtains the message $M' = \mathtt{x}^1 L_0 + \mathrm{len}(0^n)$ from which it readily can derive value L_0. From the identities

$$\begin{aligned} C[1] &= \mathtt{x}^1 L_0 + E(\mathtt{x}^1 L_0 + \mathrm{len}(0^n)) \\ C[2] &= M[2] + E(\mathtt{x}^2 L_0 + \mathrm{len}(0^n)) \\ T &= E(\mathtt{x}^2(\mathtt{x}+1) L_0 + \mathrm{len}(0^n) + M[2]) \end{aligned}$$

that we established in Section 3.1, combined with the fact that all coefficients appearing in these equations are public values (with exception of the implicit blockcipher key K), we can derive three pairs $(X_i, Y_i) \in \{0,1\}^n \times \{0,1\}^n$ such that $E(K, X_i) = Y_i$.[5] Let $(N_1, L_1) = (X_i, Y_i)$ be one such pair, and assume w.l.o.g. that $(N_1, L_1) \neq (N_0, L_0)$.[6] Lines 17,18 in Figure 3 (right) implement these two steps. The remaining steps of our attack identify a message M^0 such that if M^0 is encrypted under N_1 then the tag-computing blockcipher invocation (line 07 in Figure 2) will be on input N_0, that is, the tag is arranged to be L_0; further, a second message M^1 is identified for which this is not the case. To analyze the success rate of our attack, observe that if B is executed in game IND^0 then the internal variables Σ'' and T'' of the encryption query in line 22 evaluate to

$$\begin{aligned} \Sigma'' &= M^0[1] = \mathtt{x}^1(\mathtt{x}+1) L_1 + N_0 \\ T'' &= E(\mathtt{x}^1(\mathtt{x}+1) L_1 + \Sigma'') = E(N_0) = L_0 \end{aligned}$$

and thus the adversary stops with output 0, while in game IND^1, as $E(K, \cdot)$ is a permutation, we have $T'' \neq L_0$ and the adversary stops with output 1. In any case the adversary is nonce-respecting and outputs $b' = b$, that is we have $\mathbf{Adv}^{\mathrm{ind}}(\mathsf{B}) = 1$ and the confidentiality of OCB2 is broken.

3.3 Raw access to the blockcipher

We describe two attacks on OCB2 that are more general in nature than 'just' breaking the authenticity or confidentiality of the scheme: We show that an adversary with access to encryption and decryption routines of a (keyed) OCB2 instance effectively can evaluate the underlying blockcipher $E(K, \cdot)$ on points of their choosing (here, key K is specific to the OCB2 instance and remains unknown to the adversary). More formally, we show that oracle access to $\mathsf{enc}(K, \cdot, \cdot, \cdot)$ and $\mathsf{dec}(K, \cdot, \cdot, \cdot)$ allows for emulating an oracle for $E(K, \cdot)$. This powerful property allows us to propose in Sections 3.4 and 3.5 further attacks on the authenticity and confidentiality of OCB2 that go far beyond those of Sections 3.1 and 3.2. In the following we proceed in two steps: First we show how to obtain randomly distributed input-output pairs of $E(K, \cdot)$, then we show how to query the evaluation of $E(K, \cdot)$ on specific inputs. Note that this two-step approach is necessary as the second step relies on the first step to generate nonces.

Extracting random blockcipher mappings. A central ingredient of our IND-CCA attack from Section 3.2 was the derivation of three pairs (X_i, Y_i) such that $E(K, X_i) = Y_i$. Computing these pairs was possible because the value returned by the decryption oracle on input a forged ciphertext leaked the OCB2-internal value L, which allowed for working out the blockcipher input-output pairs from the (known) message and ciphertext. Focusing on a minimal example we arranged that the forged ciphertext would be as short as possible, leading to 'only' three such pairs (X_i, Y_i).

[5] Concretely, $X_1 = \mathtt{x}^1 L_0 + \mathrm{len}(0^n)$, $Y_1 = \mathtt{x}^1 L_0 + C[1]$, $X_2 = \mathtt{x}^2 L_0 + \mathrm{len}(0^n)$, $Y_2 = M[2] + C[2]$, $X_3 = \mathtt{x}^2(\mathtt{x} + 1) L_0 + \mathrm{len}(0^n) + M[2]$, $Y_3 = T$.

[6] Actually, formally showing that this step is w.l.o.g. seems to require an additional mild restriction, namely that value $M[2]$ is picked such that it is unequal to 0^n (line 02 of Figure 3). With this restriction in place the argument would be that $L_0 \neq 0^n \Rightarrow X_1 \neq X_2$ and $L_0 = 0^n \Rightarrow X_2 \neq X_3$, and in both cases one of the values X_1, X_2, X_3 would be different from N_0. As in practice we will always have $L_0 \neq 0^n$ anyway, and in this case the $M[2] \neq 0^n$ condition is not relevant, for notational clarity we abstain from adding it to Figure 3.

In Figure 4 (left) we specify the SamplePairs procedure which mechanizes the input-output pair extraction steps of our IND-CCA attack, simultaneously extending and simplifying them: It extends on the relevant steps of Figure 3 (right) by extracting a larger number of input-output pairs in a single shot (input parameter $m \geq 2$ controls how many), and it simplifies on them by picking message blocks such that computations are simplified as much as possible. The procedure is to be invoked relative to OCB2 encryption and decryption oracles $\mathcal{E}, \mathcal{D}$, it assumes having access to a global variable $\mathbb{E}$ (of type 'set', initially set to $\emptyset$), and each invocation of SamplePairs(m) is expected to add $m+1$ fresh pairs (X, Y) to $\mathbb{E}$ such that $E(K, X) = Y$.[7] The argument for the correctness of the procedure is almost identical with the arguments given in Sections 3.1 and 3.2 and in Appendix A, and we thus abstain from walking the reader through the individual steps. (To be on the safe side we developed a C implementation of the procedure; running it confirms that blockcipher pairs can be extracted as expected.)

```
Procedure SamplePairs$^{\mathcal{E}(\cdot,\cdot,\cdot),\mathcal{D}(\cdot,\cdot,\cdot)}(m)$
00 Global variable: $\mathbb{E}$
01 $M[1,\ldots,m-2,m] \leftarrow 0^n$
02 $M[m-1] \leftarrow \mathrm{len}(0^n)$
03 $M \leftarrow M[1] \,\|\, \ldots \,\|\, M[m]$
04 $N \leftarrow_{\$} \{0,1\}^n$
05 $C \leftarrow \mathcal{E}(N, \epsilon, M)$
06 $C[1] \,\|\, \ldots \,\|\, C[m] \,\|\, T \xleftarrow{n} C$
07 $C[m-1] \xleftarrow{+} \mathrm{len}(0^n)$
08 $C' \leftarrow C[1] \,\|\, \ldots \,\|\, C[m]$
09 $M' \leftarrow \mathcal{D}(N, \epsilon, C')$
10 $M'[1] \,\|\, \ldots \,\|\, M'[m-1] \xleftarrow{n} M'$
11 $L \leftarrow \mathtt{x}^{-(m-1)}(M'[m-1] + \mathrm{len}(0^n))$
12 For $i \leftarrow 1$ to $m-2$:
13     $(X_i, Y_i) \leftarrow (\mathtt{x}^i L, \mathtt{x}^i L + C[i])$
14 $X_{m-1} \leftarrow \mathtt{x}^{m-1} L + \mathrm{len}(0^n)$
15 $Y_{m-1} \leftarrow M'[m-1] + C[m-1]$
16 $X_T \leftarrow \mathtt{x}^{m-1}(\mathtt{x}+1)L + M'[m-1]$
17 $Y_T \leftarrow C[m]$
18 $\mathbb{E} \xleftarrow{\cup} \{(N, L)\}$
19 $\mathbb{E} \xleftarrow{\cup} \{(X_1, Y_1), \ldots, (X_{m-1}, Y_{m-1})\}$
20 $\mathbb{E} \xleftarrow{\cup} \{(X_T, Y_T)\}$
21 Return

Procedure Encipher$^{\mathcal{E}(\cdot,\cdot,\cdot)}(X_1, \ldots, X_m)$
22 Global variable: $\mathbb{E}$
23 $(N, L) \leftarrow_{\$} \mathbb{E}$
24 For $i \leftarrow 1$ to $m-1$:
25     $M[i] \leftarrow \mathtt{x}^i L + X_i$
26 $\Sigma \leftarrow \mathtt{x}^m(\mathtt{x}+1)L + X_m$
27 $M[m] \leftarrow M[1] + \ldots + M[m-1] + \Sigma$
28 $M \leftarrow M[1] \,\|\, \ldots \,\|\, M[m]$
29 $C \leftarrow \mathcal{E}(N, \epsilon, M)$
30 $C[1] \,\|\, \ldots \,\|\, C[m] \,\|\, T \xleftarrow{n} C$
31 For $i \leftarrow 1$ to $m-1$:
32     $Y_i \leftarrow \mathtt{x}^i L + C[i]$
33 $X' \leftarrow \mathtt{x}^m L + \mathrm{len}(0^n)$
34 $Y' \leftarrow M[m] + C[m]$
35 $Y_m \leftarrow T$
36 $\mathbb{E} \xleftarrow{\cup} \{(X_1, Y_1), \ldots, (X_m, Y_m)\}$
37 $\mathbb{E} \xleftarrow{\cup} \{(X', Y')\}$
38 Return $(Y_1, \ldots, Y_m)$
```

Fig. 4. Left: A procedure that generates a random collection of pairs (X_i, Y_i) such that $Y_i = E(K, X_i)$ for all i. **Right:** A procedure that finds $Y_1, \ldots, Y_m$ such that $Y_i = E(K, X_i)$ for all i. **Both:** The procedures share a common set variable $\mathbb{E}$ that is assumed to initially be $\emptyset$.

Extracting specific blockcipher mappings. We describe a procedure that takes an arbitrary vector $(X_1, \ldots, X_m)$ of blockcipher inputs and returns the vector $(Y_1, \ldots, Y_m)$ such that $Y_i = E(K, X_i)$ for all i. The underlying idea is to use the SamplePairs procedure from above to generate a random input-output pair (N, L), to use N as a (very likely fresh) nonce in an encryption query of a message M, and to exploit the a priori knowledge of value L (that would normally remain hidden) to carefully prepare this message such that the blockcipher invocations induced by the encryption process coincide exactly with the points X_i. The corresponding values Y_i can then be extracted from the ciphertext.

The specification of the corresponding Encipher procedure is in Figure 4 (right). The nonce generation in line 23 assumes that set $\mathbb{E}$ was populated before by at least one invocation of procedure SamplePairs. The likely most interesting detail of the procedure is that while the first $m-1$ values X_i are embedded directly into (the first $m-1$ blocks of) the message M, the one remaining value X_m is only implicitly

[7] We cannot prove that all these pairs are fresh (don't coincide with other pairs from the current or prior invocations of the procedure), meaning that the number of newly discovered pairs might be less than $m+1$. However, this case is extremely unlikely to happen.

embedded: We carefully choose the last message block $M[m]$ such that the sum $\Sigma = M[1] + \ldots + M[m]$ used to derive the authentication tag is such that the tag is computed as per $T = E(K, X_m)$. (That is, value Y_m coincides with the tag.) Verifying the correctness of the procedure is again straightforward. (But also for this procedure we have a C implementation that confirms that everything works as expected.)

3.4 A universal forgery attack

We note that all four attacks on the authenticity of OCB2 reported by IM [3] impose certain restrictions on the messages on which forgeries can be crafted. In contrast, the following attack on integrity is absolutely universal: forged messages can have arbitrary length, can have arbitrary content, and can come with an arbitrary associated-data string.

Given the results of Section 3.3, our attack is actually straightforward: For any combination of nonce, associated-data string, and message of their choosing, the adversary simply computes the corresponding ciphertext by simulating the regular OCB2 encryption algorithm from Figure 2, emulating all evaluations of $E(K, \cdot)$ via the Encipher procedure from Figure 4 (right). Note that by OCB2's parallelizability each such encryption simulation requires precisely one invocation of Encipher, and thus precisely one invocation of the encryption oracle $\mathcal{E}$.

3.5 Recovering plaintexts from ciphertexts

While in Section 3.2 we formally broke the indistinguishability, and thus privacy, of OCB2, an attack allowing for plaintext recovery is still outstanding. In the following we discuss several approaches towards recovering partial of full message contents from challenge ciphertexts.[8]

Recovering the final message block. A detail of OCB2 encryption (Figure 2) is that the last message block $M[m]$ is treated differently than the others: It is one-time-pad encrypted using a pad generated from the overall message length. The decryption of this block happens by recomputing the pad and xor-ing it into the ciphertext. Note that this step can be emulated via the Encipher procedure from Section 3.3, so that the last (potentially partial) message block can be recovered with probability 1.

Recovering individual plaintext bytes. Assume a situation in which in one block of a (possibly longer) message all but one bytes are known to the adversary, and the attack target is recovering this one missing byte. We propose identifying its value, with probability 1, via trial encryption: The encryption of the block is simulated a total of 256 times (using the technique from Section 3.4), each time with a different candidate for the target byte patched in. If the resulting ciphertext block coincides with the recorded one, the correct plaintext is found. Note that also this attack gets along with a single invocation of the Encipher procedure.

Recovering a range of plaintext bytes. Assume a 16 digit password is embedded into a longer message such that the only information not known to the adversary is the password.[9] An attack in the spirit of the just-described individual-byte approach will not work in this setting, simply as too many bytes would have to be guessed simultaneously. However, if 16 encryptions of the embedded password are available, each with a slightly different surrounding message, then the individual-byte technique can be leveraged to fully recover the password.

For example, if $P = P_0 \ldots P_{15}$ is the password and encryptions of the 16 messages M_i, $0 \leq i \leq 15$,

$$M_i = 1^{1024} \parallel 0^{8i} \parallel P \parallel 0^{128-8i} \parallel 1^{1024}$$

are available, then by exploiting the block-based structure of OCB2 the individual bytes of the password can be recovered one by one. For concreteness, observe that in our example one of the blocks of message M_{15} has the form $0^{120}P_0$, and P_0 can be recovered using the individual-byte recovery technique

[8] While in Section 3.3 we constructed a universal $E(K, \cdot)$ oracle, we are not aware of a similar technique to emulate a $D(K, \cdot)$ oracle. This makes message recovery more challenging than universal forging (see Section 3.4).

[9] Session cookies embedded into (predictable) HTTP headers are a classic example for this.

from above. Once P_0 is found, the next target would be P_1 as it appears in M_{14}, namely in a block of the form $0^{112}P_0P_1$. The last byte to be recovered would be P_{15}, and it would be recovered from the block $P_0 \dots P_{15}$ appearing in M_0. We note that this iterated single-byte recovery technique is also used by attacks in other cryptographic settings, e.g. by the BEAST attack against TLS [2].

Universal message recovery. Observe that a crucial assumption underlying our attacks of this and the previous sections is that the decryption oracle returns the message obtained from decrypting a forged ciphertext. In this setting, as we show next, it is possible to recover the full contents of (almost) any challenge message that is longer than two blocks.

Let $M = M[1] \shortparallel M[2] \shortparallel M[3] \shortparallel \dots$ be a secret target message that consists of more than two blocks. Encrypting it with nonce N results in a ciphertext of the form $C = C[1] \shortparallel C[2] \shortparallel C[3] \shortparallel \dots$, where $C[1] = \mathtt{x}^1L + E(\mathtt{x}^1L + M[1])$ and $C[2] = \mathtt{x}^2L + E(\mathtt{x}^2L + M[2])$ and $L = E(N)$. After learning L using the $\mathsf{Encipher}$ algorithm from Section 3.3, the adversary prepares the ciphertext $C' = C'[1] \shortparallel C'[2] \shortparallel C[3] \shortparallel \dots$ that coincides with C on all but the first two blocks, the latter being replaced by

$$C'[1] = \mathtt{x}^1L + \mathtt{x}^2L + C[2] \qquad \text{and} \qquad C'[2] = \mathtt{x}^1L + \mathtt{x}^2L + C[1] \ .$$

The adversary queries the decryption of C' to the decryption oracle as per $M' \leftarrow \mathcal{D}(N, \epsilon, C')$. Assuming for now that ciphertext C' is valid and thus not rejected, for the first two blocks $M'[1], M'[2]$ of M' it holds that

$$\begin{aligned} M'[1] &= \mathtt{x}^1L + D(\mathtt{x}^1L + C'[1]) \\ &= \mathtt{x}^1L + D(\mathtt{x}^1L + \mathtt{x}^1L + \mathtt{x}^2L + C[2]) \\ &= \mathtt{x}^1L + D(\mathtt{x}^2L + \mathtt{x}^2L + E(\mathtt{x}^2L + M[2])) \\ &= \mathtt{x}^1L + \mathtt{x}^2L + M[2] \end{aligned}$$

and, symmetrically, $M'[2] = \mathtt{x}^2L + \mathtt{x}^1L + M[1]$. Note that $M[1] + M[2] = M'[1] + M'[2]$ and that thus the sums Σ and Σ' computed over all message blocks of M and M', respectively, are identical (in line 16 of Figure 2). This ultimately means that ciphertext C' is considered valid and M' is indeed returned by the decryption oracle. Recovering M from M' is now trivial. (This step is actually *identical* with the step of deriving C' from C.)

4 Conclusion

We improve on recent results by Inoue and Minematsu [3] on OCB2 by (a) describing a universal forgery attack against the scheme's authenticity (that allows for the creation of valid ciphertexts on arbitrary messages and associated-data strings), (b) showing that the scheme formally does not achieve the IND-CCA notion of confidentiality, and (c) proposing a couple of plaintext recovery attacks. Our adversaries require little resources and achieve high success rates. We consider all mentioned attacks (whether by IM or us) rather severe. Given that OCB2 is crucially broken, all corresponding implementations should be upgraded as soon as possible (e.g., to AES-GCM [5] or a winner of the CAESAR competition [9]).

We note that IM in [3] propose a fix for OCB2 that would rule out their attacks on authenticity. We believe that any such fix would also resolve the confidentiality issues that we identified.

References

1. 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, Germany, Kyoto, Japan (Dec 3–7, 2000)
2. Duong, T., Rizzo, J.: Here come the ⊕ ninjas. Unpublished manuscript (2011)
3. Inoue, A., Minematsu, K.: Cryptanalysis of OCB2. Cryptology ePrint Archive, Report 2018/1040 (2018), https://eprint.iacr.org/2018/1040
4. Jutla, C.S.: Encryption modes with almost free message integrity. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 529–544. Springer, Heidelberg, Germany, Innsbruck, Austria (May 6–10, 2001)

5. McGrew, D.A., Viega, J.: The security and performance of the galois/counter mode of operation (full version). Cryptology ePrint Archive, Report 2004/193 (2004), http://eprint.iacr.org/2004/193
6. Rogaway, P.: OCB Mode. Personal website, http://web.cs.ucdavis.edu/~rogaway/ocb/
7. 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, Germany, Jeju Island, Korea (Dec 5–9, 2004)
8. Rogaway, P.: Efficient instantiations of tweakable blockciphers and refinements to modes OCB and PMAC (2004), http://web.cs.ucdavis.edu/~rogaway/papers/offsets.pdf
9. Various authors: CAESAR – Competition for Authenticated Encryption: Security, Applicability, and Robustness. Website (2018), https://competitions.cr.yp.to/caesar.html

A Generalized Inoue–Minematsu attack on authenticity

In Section 3.1 we reproduce from IM [3] the most basic attack on the authenticity of OCB2. However, in their report IM also propose a close variant that involves longer messages. Our findings and algorithms from Section 3 are actually much closer in spirit to the more general variant, so we reproduce its details here.[10] In that sense we specify in Figure 5 a corresponding family of adversaries A_m for the INT game from Figure 1 (left), where index $m \geq 2$ indicates the message length to be used (measured in blocks). Note that adversary A_2 is identical to adversary A from Figure 3 (left).

```
Adversary $\mathsf{A}_m^{\mathcal{E}(\cdot,\cdot,\cdot),\mathcal{D}(\cdot,\cdot,\cdot)}$
Step 1:
  For $i \leftarrow 1$ to $m-2$:
    Pick any $M[i] \in \{0,1\}^n$
  $M[m-1] \leftarrow \mathrm{len}(0^n)$
  Pick any $M[m] \in \{0,1\}^n$
  $M \leftarrow M[1] \parallel \ldots \parallel M[m]$
  Pick any $N_0 \in \{0,1\}^n$
  Query $C \leftarrow \mathcal{E}(N_0, \epsilon, M)$
Step 2:
  $C[1] \parallel \ldots \parallel C[m] \parallel T \overset{n}{\leftarrow} C$
  For $i \leftarrow 1$ to $m-2$:
    $C'[i] \leftarrow C[i]$
  $C'[m-1] \leftarrow M[1] + \ldots + M[m-2] + C[m-1] + \mathrm{len}(0^n)$
  $T' \leftarrow M[m] + C[m]$
  $C' \leftarrow C'[1] \parallel \ldots \parallel C'[m-1] \parallel T'$
  Query $M' \leftarrow \mathcal{D}(N_0, \epsilon, C')$
Stop
```

Fig. 5. Generalized IM attack on authenticity [3, Sec. 4.2].

We trace the values of some variables throughout an execution of adversary A_m within game INT. Note that we obtain $T' = T^*$ and $M' \neq M$, so the attack is successful.

$$\begin{aligned}
M[m-1] &= \mathrm{len}(0^n) \\
L_0 &= E(N_0) \\
C[i] &= \mathtt{x}^i L_0 + E(\mathtt{x}^i L_0 + M[i]) \qquad \text{for } 1 \leq i \leq m-2 \\
C[m-1] &= \mathtt{x}^{m-1} L_0 + E(\mathtt{x}^{m-1} L_0 + M[m-1]) = \mathtt{x}^{m-1} L_0 + E(\mathtt{x}^{m-1} L_0 + \mathrm{len}(0^n)) \\
C[m] &= M[m] + E(\mathtt{x}^m L_0 + \mathrm{len}(0^n)) \\
\Sigma &= M[1] + \ldots + M[m] \\
T &= E(\mathtt{x}^m (\mathtt{x}+1) L_0 + \Sigma) \\
C'[m-1] &= M[1] + \ldots + M[m-2] + \mathtt{x}^{m-1} L_0 + E(\mathtt{x}^{m-1} L_0 + \mathrm{len}(0^n)) + \mathrm{len}(0^n) \\
T' &= M[m] + C[m] = M[m] + M[m] + E(\mathtt{x}^m L_0 + \mathrm{len}(0^n)) = E(\mathtt{x}^m L_0 + \mathrm{len}(0^n)) \\
M'[i] &= M[i] \qquad \text{for } 1 \leq i \leq m-2 \\
M'[m-1] &= C'[m-1] + E(\mathtt{x}^{m-1} L_0 + \mathrm{len}(0^n)) = M[1] + \ldots + M[m-2] + \mathtt{x}^{m-1} L_0 + \mathrm{len}(0^n) \\
\Sigma' &= M'[1] + \ldots + M'[m-1] = \mathtt{x}^{m-1} L_0 + \mathrm{len}(0^n) \\
T^* &= E(\mathtt{x}^{m-1}(\mathtt{x}+1) L_0 + \Sigma') = E(\mathtt{x}^{m-1}(\mathtt{x}+1) L_0 + \mathtt{x}^{m-1} L_0 + \mathrm{len}(0^n)) = E(\mathtt{x}^m L_0 + \mathrm{len}(0^n)) \\
M' &= M[1] \parallel \ldots \parallel M[m-2] \parallel (\mathtt{x}^{m-1} L_0 + \mathrm{len}(0^n))
\end{aligned}$$

[10] To be precise, the attack in [3] is even more general than the version presented here: our version only considers multi-block messages while their version is a little more tolerant.