A weakness in OCB3 used with short nonces allowing for a break of authenticity and confidentiality

Jean Liénardy[a,*], Frédéric Lafitte[a]

[a] *Royal military Academy, Rue Hobbema 8, 1040, Bruxelles, Belgium*

Abstract

OCB3 is a mature and provably secure authenticated encryption mode of operation which allows for associated data (AEAD). This note reports a small flaw in the security proof of OCB3 that may cause a loss of security in practice, even if OCB3 is correctly implemented in a trustworthy and nonce-respecting module. The flaw is present when OCB3 is used with short nonces. It has security implications that are worse than nonce-repetition as confidentiality and authenticity are lost until the key is changed. The flaw is due to an implicit condition in the security proof and to the way OCB3 processes nonce. Different ways to fix the mode are presented.

1. Introduction

Authenticated encryption (AE) is the process of ensuring the confidentiality and authenticity of plaintexts using a single algorithm with a single symmetric key. Modern AE algorithms allow for associated data (AD) to be authenticated along with the plaintext, in which case we refer to the algorithm as AEAD.

Designed in early 2000's, OCB1 is the first generation of a series of variant of OCB which provides a provably-secure AE mode for a blockcipher [1]. Although initially encumbered by patents, the designers of OCB have since renounced to all intellectual property related to OCB1 and successors.

The authors of OCB1 published a second version of OCB in 2004 called OCB2 to allow for the authentication of AD [2]. They have recast the security proof of the mode into the framework of *tweakable blockcipher* [3, 4]. Modifications to the mode itself were made in order to achieve better performances. The mode was standardized in ISO/IEC 19772 [5]. It was shown that the very changes from OCB1 to OCB2 caused the mode to have a fatal flaw that allows to break its authenticity and confidentiality (see [6] for a comprehensive review).

This note focuses on OCB3, the final version of OCB, that has been published in 2011 [7]. This last generation fixes the bug of OCB2 and has the best performance amongst its predecessors, mainly due to a convoluted pretreatment of its nonce.

The mode is now seen as mature and is well accepted by the cryptographic community. It has been specified in RFC 7253 [8] and has been selected in 2019 in the final portfolio of the CAESAR competition, which aimed to identify several AE schemes in order to replace AES-GCM, for use in high-performance applications. Moreover, the mode OCB3 has also been used as a building block by candidates of the NIST Lightweight Cryptography Standardization Process

*Corresponding author
Email addresses: jean.lienardy@mil.be (Jean Liénardy), frederic.lafitte@mil.be (Frédéric Lafitte)

such as Pyjamask [9]. OCB3 is implemented in widespread cryptographic libraries such as Botan, Bouncy Castle, Crypto++, Libgcrypt or OpenSSL.

The outline of this note is as follows. Section 2 describes OCB3 and recalls its security claim. In section 3, we present a rather trivial forgery that was not previously reported in the literature. Essentially, the attacker uses short nonces of less than 6 bits to cause internal collisions in the mode and forge authentication tags. We spot that the flaw in the security proof is due to a somewhat hidden restriction in the proof that is not respected by OCB3. While the weakness may appear harmless, we show in section 4 that a single query is sufficient to loose all security (both confidentiality and authenticity) with high success probability, and we present some use cases where this flaw could be exploited in practice. Ways to fix OCB3 are proposed in section 5.

2. Description of OCB3

In this section, we introduce preliminary material. We give the description of the mode in section 2.1 and recall the security claim of OCB3 in section 2.2.

2.1. Description of OCB3

Notation. OCB3 consists of an encryption algorithm $\mathcal{E}$ and a decryption algorithm $\mathcal{D}$. The underlying blockcipher is written $E : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$. In this note, a block refers to a bitstring of length n whereas it refers to a bitstring of length at most n in [7]. We also fix $n = 128$. We write 0^i the bitstring made of $i \geq 0$ zeros, ε the empty bitstring, $\|$ bitstring concatenation, and $|x|$ the bitlength of bitstring x. When the context is clear, the symbol $\|$ is omitted. The function $\mathsf{msb}_i(x)$ (resp. $\mathsf{lsb}_i(x)$) outputs a bitstring composed of the $i > 0$ left-most (resp. right-most) bits of x. Shifting x by i positions to the left is written $x \ll i$ (the bits $\mathsf{msb}_i(x)$ are lost and the bits $\mathsf{lsb}_i(x \ll i)$ are 0). The function $\mathsf{ntz}(x)$ returns the number of trailing zeros in the binary encoding of x. Finally, the function $\mathsf{double}(x)$ returns the value $(x \ll 1)$ when the block x has most significant bit 1 and $(x \ll 1) \oplus 0^{121}10000111$ otherwise, with $\oplus$ the bitwise exclusive OR.

Input of $\mathcal{E}$. A key $K \in \{0,1\}^k$, a nonce $N \in \{0,1\}^{\leq 120}$, a plaintext $M = M_1 \| \cdots \| M_m \| M_*$, and associated data $A = A_1 \| \cdots \| A_a \| A_*$. Here, $M_1, \ldots, M_m, A_1, \ldots A_a$ are blocks whereas $M_*, A_* \in \{0,1\}^{<n}$

Output of $\mathcal{E}$. Ciphertext $C = C_1 \| \cdots \| C_m \| C_*$ of same bitlength as that of the input plaintext, and an authentication tag T of predefined length $\tau \in \{0, \ldots, 128\}$.

Description of $\mathcal{E}$.

1. Compute $L_* = E_K(0^n)$, $L_\$ = \mathsf{double}(L_*)$, and L_i for $i \in \{0, 1, 2, ...\}$:
$$\begin{aligned} L_0 &= \mathsf{double}(L_\$) \\ L_i &= \mathsf{double}(L_{i-1}), \qquad i > 0. \end{aligned}$$
2. Compute blocks Δ'_i for $i \in \{0, \ldots, a\}$ and Δ_i, $i \in \{0, \ldots, m\}$ as follows:
$$\begin{array}{lcllcl} \Delta'_0 & = & 0^n & \Delta_0 & = & \mathsf{Init}_K(N) \\ \Delta'_i & = & \Delta'_{i-1} \oplus L_{\mathsf{ntz}(i)} & \Delta_i & = & \Delta_{i-1} \oplus L_{\mathsf{ntz}(i)} \end{array}$$
in order to process the blocks of associated data and plaintext.
3. Compute the ciphertext and tag as shown in Figure 1 where 10^* denotes the padding (a single bit 1 followed by as many zero bits as needed to form a block). The tag is then truncated to its predefined length τ.

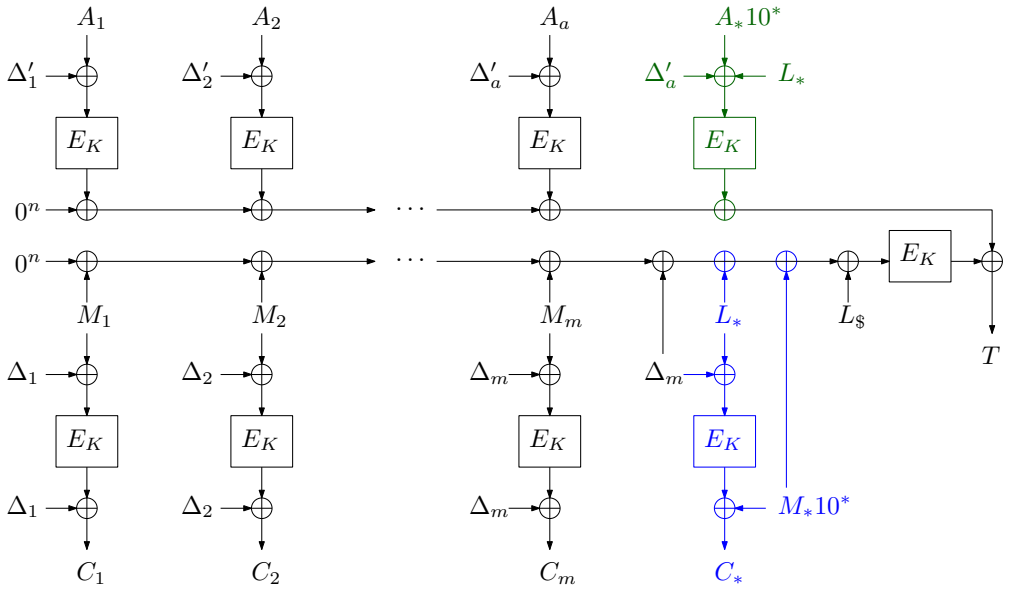

Figure 1: OCB3 encryption with 128 bit tag. The parts in green and blue are only performed when $A_* \neq \varepsilon$ and $M_* \neq \varepsilon$, respectively.

This note focuses on the function $\mathsf{Init}_K(N)$ which performs the following steps:

1. $\mathsf{PadN} \leftarrow \mathsf{lsb}_7(\tau) \,\|\, 0^{120-|N|} \,\|\, 1 \,\|\, N$
2. $\mathsf{Top} \leftarrow \mathsf{msb}_{122}(\mathsf{PadN}) \,\|\, 0^6$
3. $\mathsf{Bot} \leftarrow \mathsf{lsb}_6(\mathsf{PadN})$
4. $\mathsf{KTop} \leftarrow E_K(\mathsf{Top})$
5. $\mathsf{Stretch} \leftarrow \mathsf{KTop} \,\|\, (\mathsf{KTop} \oplus (\mathsf{KTop} \ll 8))$
6. **Return** $\mathsf{msb}_{128}(\mathsf{Stretch} \ll \mathsf{Bot})$

Decryption algorithm $\mathcal{D}$. The decryption algorithm takes in (K, N, A, C, T), computes a message M and returns either M or $\perp$ ("authentication failure") whether or not the tag computed with M corresponds to T. The precise description of $\mathcal{D}$ is straightforward from the definition of $\mathcal{E}$.

2.2. Security claim

The security of OCB3 is expressed in terms of advantages in the concrete security framework [10]. The security model considers the key K to be a random variable and gives the adversary access to encryption and decryption oracles. The adversary must distinguish the case where these oracles correspond to $\mathcal{E}_K$ and $\mathcal{D}_K$ from the case they are their ideal counterparts, denoted \$ and $\perp$. The oracle \$ returns a random string of expected length and the oracle $\perp$ always returns "authentication failure". The advantage of an attacker $\mathcal{A}$ in breaking the authenticity and confidentiality is defined as

$$\mathbf{Adv}^{\mathrm{ae}}(\mathcal{A}) = \mathbf{P}_K[\mathcal{A}^{\mathcal{E}_K,\mathcal{D}_K} \Rightarrow 1] - \mathbf{P}_K[\mathcal{A}^{\$,\perp} \Rightarrow 1].$$

Here and in the following, all probabilities are taken over the random variable K, unless otherwise stated. Furthermore, the attacker $\mathcal{A}$ does not make a decryption query (N, A, C, T) if (C, T) is the output of a previous encryption query (N, A, M).

The security of OCB3 is summarized by Theorem 1 of [11]. For any $0 \leq \tau \leq n$, this result states that the advantage of any nonce-respecting adversary $\mathcal{A}$ is upper-bounded according to

$$\mathbf{Adv}^{\mathrm{ae}}(\mathcal{A}) \leq 6(\sigma + 2q)^2/2^n + 1.1q_d/2^\tau + \mathbf{Adv}_E^{\mathrm{sprp}}(\mathcal{B}), \tag{1}$$

where q denotes the total number of queries, from which q_d are decryption queries, and σ is the total number of blocks queried. Furthermore, $\mathbf{Adv}_E^{\mathrm{sprp}}(\mathcal{B})$ is the advantage of any adversary $\mathcal{B}$ with resources comparable to $\mathcal{A}$ in breaking the strong prp-security of the blockcipher E.

3. A distinguisher for OCB3

In this section, we give a distinguisher for OCB3 with $\tau = n$ (no truncation of the final tag). From now, we choose a blockcipher E that is ideal ($\mathbf{Adv}_E^{\mathrm{sprp}}(\mathcal{B}) = 0$) as the attack does not use any weakness of the blockcipher.

According to equation (1), an attacker that performs a single decryption query (with $q = q_d = \sigma = 1$) cannot have an advantage greater than $1/2^{n-7}$. However, we present in this section a simple distinguisher $\mathcal{A}$ making a single query that has advantage

$$\mathbf{Adv}^{\mathrm{ae}}(\mathcal{A}) = 1/8.$$

3.1. Observation

We start by observing that, for $N = \varepsilon$, the following equality holds with probability $1/4$:

$$\Delta_0 = L_\$. \tag{2}$$

Indeed, for a given key K, the setup phase computes $L_* = E_K(0^n)$ and $L_\$ = \mathsf{double}(L_*)$. We note that, depending on the most significant bit of L_*, we have $L_\$ = E_K(0^n) \ll 1$ with probability $1/2$. Moreover, for $N = \varepsilon$, the procedure Init_K computes $\mathsf{PadN} = 0^{127} \,\|\, 1$, $\mathsf{Top} = 0^n$ and $\mathsf{Bot} = 1$, and we have

$$\begin{aligned}
\mathsf{KTop} &\leftarrow E_K(\mathsf{Top}) = E_K(0^n) \\
\mathsf{Stretch} &\leftarrow E_K(0^n) \,\|\, (E_K(0^n) \oplus (E_K(0^n) \ll 8)) \\
\Delta_0 &\leftarrow \mathsf{msb}_{128}(\mathsf{Stretch} \ll 1).
\end{aligned}$$

After the initialisation phase, the value of Δ_0 is $(E_K(0^n) \ll 1) \oplus (0^{127} \,\|\, c)$ where $c \in \{0, 1\}$ is a single bit, and thus $\Delta_0 = E_K(0^n) \ll 1$ with probability $1/2$.

Therefore, equation (2) holds with probability $1/4$. We discuss a similar observation that holds for every nonce such that $|N| < 6$ in section 4.3.

3.2. Simple forgery

The following forged message contradicts the theorem claim: choose $N = \varepsilon$, and $A = c$, $C = c' = c$ single bit associated data and ciphertext. We claim that $T = 0^n$ is a valid tag for (N, A, C) with high probability $1/8$. Indeed, upon decryption, the ciphertext c' is decrypted into c with probability $1/2$. The tag T' is computed according to

$$T' = E_K((c \,\|\, 10^*) \oplus \Delta_0 \oplus L_* \oplus L_\$) \oplus \mathsf{Auth},$$

where Auth is

$$\mathsf{Auth} = E_K((c \,\|\, 10^*) \oplus L_*).$$

As shown in equation (2), $\Delta_0 = L_\$$ holds with probability $1/4$, therefore the tag $T = 0^n = T'$ is valid with high probability $1/8$, and $\mathcal{D}_K$ will output the message c as valid.

This adversary algorithm requires no encryption query and a single decryption query but allows the attacker to forge a message with probability $1/8$, hence its advantage is exactly $\mathbf{Adv}^{\mathrm{ae}}(\mathcal{A}) = 1/8$ which contradicts the statement of the security claim recalled in section 2.2.

3.3. Assumptions used in the proof

In [7] as well as in [12, 11], the mode comes with a proof of security. The proof is done in two steps: First, prove the security of the mode instantiated with an ideal tweakable blockcipher (Lemma 2 in [7]). Second, prove that the tweakable blockcipher construction with an ideal blockcipher is ideal (Lemma 3 in [7]).

In the proof of Lemma 3, the following restriction is introduced: "*the adversary [does] not ask a query with* $\mathsf{Top} = 0$". This constraint is mandatory for the proof of Lemma 3 to hold. As seen in the previous section, an adversary can violate this requirement by querying nonces with $|N| < 6$.

The statement of Lemma 3 is therefore incomplete and must be modified in order to account for the additional restriction on the nonce. Furthermore, the mode itself must have additional restrictions or must be modified. We discuss this in section 5.

4. Security implications

The preceding section showed that the specification of OCB3 has a flaw and we explore in this section its implications. Let us assume that for a given key, equation (2) is valid. This holds with probability 1/4. In that case, an attacker can learn the value of L_* by a single encryption query with $N = A = M = \varepsilon$. Indeed, $\Delta_0 \oplus L_\$ = 0^n$, $\mathsf{Auth} = 0^n$ and the output tag is L_*:

$$T \leftarrow E_K(0^n \oplus \Delta_0 \oplus L_\$) \oplus \mathsf{Auth} = E_K(0^n) = L_*.$$

We show in section 4.1 a few practical scenarios in which this attack is plausible. Section 4.2 presents actual consequences of the adversarial knowledge of L_*, namely the complete loss of authenticity and confidentiality. In that sense, the leaking of L_* has a much higher impact on security than the nonce-reuse [8].

4.1. Practical attack scenario

This section explains why the defect reported in this note is relevant in practice. We do note that several public implementations check that $N \in \{0,1\}^{120}$, thereby preventing an empty nonce but this is not necessarily the case. To give an example, the widespread cryptographic library BouncyCastle closely follows OCB3's specification by just checking that $N \in \{0,1\}^{\leq 120}$.

Where security is critical, it is unlikely that an implementation will rely on these libraries; the algorithm will more likely be implemented from scratch in a Hardware Security Module (HSM) that will undergo a formal evaluation process for certification.

Given that nonce-repetition has a critical impact on the security of OCB3, a sound design choice is to implement the nonce generation process in the HSM. However, in order to make the product as interoperable as possible, nothing precludes letting the user choose nonce lengths as long as the HSM checks that chosen lengths comply with algorithm requirements, i.e. $N \in \{0,1\}^{\leq 120}$.

Therefore, an API that offers an encryption function that inputs the nonce length and tag length, while checking that the nonce bytelength is less than or equal to 15, is an API compliant with the RFC (the HSM having the role of preventing nonce repetition, for example by keeping track of a counter for each byte length). Such a scenario would be vulnerable to the presented flaw.

Learning the value of L_* may justify the cost of an attack, mainly because the attacker can exploit the knowledge of L_* at will until the key is changed. The next section shows that knowledge of $L_* = E_K(0^n)$ is sufficient to break confidentiality and authenticity for a fixed K.

4.2. Consequences of the leakage of L_*

The leakage of $L_* = E_K(0^n)$, and therefore of $L_\$$ and all the L_i's, has disastrous consequences for the security of the mode that are worse than the nonce reuse: it looses its authenticity and confidentiality (even for message encrypted before the leak). We illustrate this with some attack scenarios requiring either none or a few additional queries. We stress that in all cases, the attacker is nonce-respecting.

Breaking the authenticity. A selective forgery is requiring no additional query is possible for an attacker that has L_*: given any known ciphertext, he can swap any two ciphertext blocks up to the addition of a known constant. For example, if T correspond to a ciphertext $C_1 \| C_2 \| \dots$, the new ciphertext $C_2 \oplus L_{\mathsf{ntz}(2)} \| C_1 \oplus L_{\mathsf{ntz}(2)} \| \cdots$ is valid under the same tag.

The knowledge of L_* also allows for a two-step universal forgery of a tag on any chosen plaintext blocks $M_1 \| \cdots \| M_m$, without constraining the nonce value N (i.e. chosen nonce). The first step consists in obtaining the value of $E_K(\mathsf{Top}^N)$. Here and in this section, the uppercase superscripts indicate the corresponding nonce. This can be done by an encryption query under a nonce $\tilde{N}$ such that $|\tilde{N}| < 6$ that has not been previously queried. As the value of $L_0 \oplus \Delta_0^{\tilde{N}}$ is known to the attacker, the query of $\mathsf{Top}^N \oplus L_0 \oplus \Delta_0^{\tilde{N}}$ returns to the attacker the value of $E_K(\mathsf{Top}^N)$.

The second step consists in doing an encryption query under a nonce N' such that $\mathsf{Top}^N = \mathsf{Top}^{N'}$ (e.g. $N' \oplus N = 0^* \| 1$). Together with N' and $A = \varepsilon$, the message queried is

$$M'_1 \| \cdots \| M'_m \| M'_{m+1} = M_1 \oplus \delta_0^{NN'} \| \cdots \| M_m \oplus \delta_0^{NN'} \| \bigoplus_{i=1}^m M_i \oplus L_{\mathsf{ntz}(m+1)} \oplus \delta_0^{NN'} \oplus L_\$,$$

where $\delta_0^{NN'} = \Delta_0^N \oplus \Delta_0^{N'}$ is known to the attacker. The resulting ciphertext blocks C'_i ($i \le m$) give the encryption of the message under the nonce N, up to the addition of $\delta_0^{NN'}$:

$$C'_i \oplus \delta_0^{NN'} = E_K(M'_i \oplus \Delta_i^{N'}) \oplus \Delta_i^{N'} \oplus \delta_0^{NN'} = E_K(M_i \oplus \Delta_i^N) \oplus \Delta_i^N = C_i$$

and $C'_{m+1} = E_K(\bigoplus M_i \oplus \Delta_m^N \oplus L_\$) \oplus \Delta_{m+1}^{N'}$. By construction, the ciphertext $C_1 \| \cdots \| C_m$ with tag $T = C'_{m+1} \oplus \Delta_{m+1}^{N'}$ is a valid one for the nonce N, that has not been used.

Breaking the confidentiality. The sole knowledge of L_* allows the attacker to detect relations between plaintext blocks (akin to the lack of confidentiality of the ECB mode). Indeed, if two known ciphertext blocks verify $C_i \oplus C_j = \bigoplus_{k=i+1}^{j} L_{\mathsf{ntz}(k)}$, a value independent of the nonce; then the attacker knows that the corresponding plaintext satisfies $M_i \oplus M_j = C_i \oplus C_j$.

Moreover, given a known ciphertext $(N, C = C_1 \| \cdots \| C_m, T)$ with odd m and the knowledge of L_*, an attacker can obtain the corresponding plaintext using a few additional queries.

The first step is as in the universal forgery: obtain the value of $E_K(\mathsf{Top}^N)$. The second step consists in doing a decryption query (N', C', T) under a nonce N' such that $\mathsf{Top}^{N'} = \mathsf{Top}^N$ and with $C'_i = C_i \oplus \delta_0^{NN'}$. The resulting plaintext will be $M'_i = M_i \oplus \delta_0^{NN'}$. As for the tag, it will be considered as valid as

$$T' = E_K(\bigoplus (M_i \oplus \delta_0^{NN'}) \oplus \Delta_m^{N'} \oplus L_\$) = E_K(\bigoplus M_i \oplus \Delta_m^N \oplus L_\$) = T.$$

(For even m, an additional step to compute a valid T in a nonce-respecting setting is required.)

4.3. Non-empty nonce generalisation

The attacks of the preceding sections make usage of the rather special case of an empty nonce $N = \varepsilon$. In that case, the probability of success is $1/4$. For the sake of completeness, we show that if a non-empty nonce has length $0 < |N| < 6$, a similar attack can be mounted. Given the (integer) value $\mathsf{Bot} = 1 \,\|\, N$ for the nonce N ($2 \leq \mathsf{Bot} \leq 63$), we have

$$\Delta_0^N = (E_K(0^n) \ll \mathsf{Bot}) \oplus (0^{128-\mathsf{Bot}} \,\|\, C)$$

for a certain $C \in \{0,1\}^{\mathsf{Bot}}$. This is equal to $E_K(0^n) \ll \mathsf{Bot}$ with probability $1/2^{\mathsf{Bot}}$. Furthermore, we have $L_{\mathsf{Bot}-2} = E_K(0^n) \ll \mathsf{Bot}$ which also holds with probability $1/2^{\mathsf{Bot}}$, hence $\Delta^N_{2^{\mathsf{Bot}-1}-1} = \Delta_0^N \oplus L_{\mathsf{Bot}-2} = 0^n$ with a probability at least $1/2^{2\,\mathsf{Bot}}$. One must just query any message such that the block $2^{\mathsf{Bot}-1} - 1$ is 0^n. The corresponding ciphertext will have $E_K(0^n)$ as block indexed by $2^{\mathsf{Bot}-1} - 1$. This variant of the attack is successful with probability at least $1/2^{2\,\mathsf{Bot}}$.

5. Fixing OCB3

To make the flaw unexploitable, the mode description must be changed. The most immediate fix therefore consists in ensuring that the nonce has length $|N| \geq 6$ and to make that constraint mandatory (especially in [8]). Indeed, this ensures that Top is different from the block 0^n and therefore prevents the collision between Δ_0 derived from $E_K(\mathsf{Top})$ and one of the L's.

We present another way to fix OCB3 which does not require adding a length constraint, hence allowing any $|N| < 120$ (even empty nonces, as permitted in RFC5116 [13]). This fix consists in randomizing the value of PadN using L_*, a key-dependent value that has been precomputed for the given key. Therefore, after the line 1 of the procedure Init, we suggest to apply $\mathsf{PadN} \leftarrow \mathsf{PadN} \oplus L_*$. This has the effect of randomizing the value of Top and Bot, hence preventing the attack.

We note that this modification is akin to how the nonce is treated in OCB1 [1]. Indeed, in OCB1, the nonce is a block $N \in \{0,1\}^{128}$ and the value of $\Delta_0^{(\mathrm{OCB1})}$ is obtained through

$$\Delta_0^{(\mathrm{OCB1})} = E_K(N \oplus L_*).$$

For this reason, OCB1 is not subject to our attack.

5.1. Specific nonce requirements

We finally acknowledge that the documents [7, 12] include the following footnote: *In practice one would either restrict nonces to byte strings of 1-15 bytes, or else demand that nonces have a fixed length, say exactly 12-bytes. Under RFC 5116, a conforming scheme should use a 12-byte nonce.* As we have seen, such a restriction is not superfluous and the mode is not secure in the given form. As for the RFC 5116 [13], while it recommends to fix the nonce bytelength to 12, it does not include the OCB3 algorithm as an AEAD mode and the fixed nonce-length is not specified as a requirement (in fact, the variable nonce-length scenario is let as optional). The RFC 7253 [8] that specifies the OCB mode does not impose any other requirement on the length of N apart from $|N| < 120$ and should be modified accordingly.

6. Conclusion

In this short note, we have presented a weakness in OCB3: even if the mode is correctly implemented, a rather simple attack can be mounted and will likely succeed, breaking both the confidentiality and authenticity.

We have shown that, despite of the mode being provably secure, the security proof hides an additional internal constraint whose fulfilment is not properly enforced by specification documents. This highlights the fact that small and seemingly insignificant details can have severe security consequences and illustrates that all assumptions required by the proof should be exhaustively listed so that it is sufficient for practitioners to focus on their fulfilment.

In the case of OCB3, it is easy to fix the algorithm's specification in order to avoid the weakness and abide to the full assumptions of the security proof. If the description is unchanged, the requirement $N \geq 6$ must become an absolute requirement.

References

[1] P. Rogaway, M. Bellare, J. Black, OCB: A block-cipher mode of operation for efficient authenticated encryption, ACM Transactions on Information and System Security (TISSEC) 6 (3) (2003) 365–403.

[2] P. Rogaway, Efficient instantiations of tweakable blockciphers and refinements to modes OCB and PMAC, in: International Conference on the Theory and Application of Cryptology and Information Security, Springer, 2004, pp. 16–31.

[3] M. Liskov, R. L. Rivest, D. Wagner, Tweakable block ciphers, in: Annual International Cryptology Conference, Springer, 2002, pp. 31–46.

[4] M. Liskov, R. L. Rivest, D. Wagner, Tweakable block ciphers, Journal of cryptology 24 (3) (2011) 588–613.

[5] ISO/IEC JTC 1/SC 27, Information technology-Security techniques-Authenticated encryption, Standard 19772, OCB scheme within standard deprecated (First ed. 2009).

[6] A. Inoue, T. Iwata, K. Minematsu, B. Poettering, Cryptanalysis of OCB2: attacks on authenticity and confidentiality, Journal of Cryptology 33 (4) (2020) 1871–1913.

[7] T. Krovetz, P. Rogaway, The software performance of authenticated-encryption modes, in: International Workshop on Fast Software Encryption, Springer, 2011, pp. 306–327.

[8] T. Krovetz, P. Rogaway, The OCB authenticated-encryption algorithm, RFC 2753, RFC Editor (2014).

[9] D. Goudarzi, J. Jean, S. Kölbl, T. Peyrin, M. Rivain, Y. Sasaki, S. M. Sim, Pyjamask: Block cipher and authenticated encryption with highly efficient masked implementation, IACR Transactions on Symmetric Cryptology (2020) 31–59.

[10] M. Bellare, A. Desai, E. Jokipii, P. Rogaway, A concrete security treatment of symmetric encryption, in: Proceedings 38th Annual Symposium on Foundations of Computer Science, IEEE, 1997, pp. 394–403.

[11] T. Krovetz, P. Rogaway, The Design and Evolution of OCB, Journal of Cryptology 34 (4) (2021) 1–32.

[12] T. Krovetz, P. Rogaway, OCB (v1. 1), Submission to the CAESAR Competition (2016).

[13] D. McGrew, An interface and algorithms for authenticated encryption, RFC 5116, RFC Editor (2008).