Insecurity of RCB: Leakage-Resilient Authenticated Encryption

Farzaneh Abed[1], Francesco Berti[2], Stefan Lucks[1]

[1] Bauhaus-Universität Weimar, Germany, [2] ICTEAM/ELEN/Crypto Group, Université catholique de Louvain, Belgium.
farzaneh.abed@uni-weimar.de, stefan.lucks@uni-weimar.de, francesco.berti@uclouvain.be

Abstract. Leakage-resilient cryptography is about security in the presence of leakage from side-channels. In this paper, we present several issues of the RCB block cipher mode. Agrawal et al [2] proposed recently RCB as a leakage-resilient authenticated encryption (AE) scheme. Our main result is that RCB fails to provide authenticity, even in the absence of leakage.

Keywords: authenticated encryption, leakage-resilience, block cipher, attack

1 Introduction

One of the main issues of modern cryptography is the vulnerability of cryptosystem implementations against side-channel attacks. To thwart this kind of attack, countermeasures such as masking [14], shuffling [19] and noise addition [9] have been proposed. For constrained devices, which are likely exposed to side-channel attacks, those countermeasures are quite expensive.

Another approach, initiated with high hopes [6,11], is to design "leakage-resilient" schemes. The goal is to maintain a certain level of security, even when the implementation leaks some information about internal secrets to the adversary. During the last decade many methods have been proposed, yet few did focus on Symmetric Cryptography, or block cipher based schemes.

There have been a handful of proposals for leakage-resilient encryption (or rather, for leakage-resilient pseudorandom generators (PRGs) and pseudorandom function generators (PRFs)), such as [6,7,13,17,18,7,20,21]. The requirements for the underlying primitives are different; for some, e.g., a "weak PRF" would suffice. But all those proposals can be naturally instantiated using a block cipher. Block cipher based leakage-resilient message authentication has not been studied much, but some proposals exist, see [15,12].

To the best of our knowledge, RCB is the first leakage resilient authenticated encryption scheme that is proposed by Agrawal et.al [2]. Later, Berti et.al [4] and Dobraunig et.al [5] proposed DIV and ISAP, in two independent works, as a new leakage resilient authenticated encryption schemes.

The RCB Mode. The RCB mode from Agrawal et.al [2] is based on the OCB mode [8] for authenticated encryption in the black-box model (i.e. without leakage). In addition to some seemingly minor modifications, RCB enhances OBC by a re-keying scheme, similar to the one from [10]. Using re-keying makes sure that the block cipher is never called twice using the same key, and this is where the claimed leakage resilience comes from.

Our Contribution and Results. In this paper we analyze security of RCB. As it will turn out, RCB suffers from a couple of issues as follow:

1. Neither RCB nor OCB are robust to nonce-misuse or to the release of unverified plaintexts (RUP), see [3,1].[1]
2. RCB requires sender and receiver to synchronise counters; if synchronisation fails, then RCB requires interactive resynchronisation (cf. [2, Figure 2]),
3. RCB is inherently vulnerable to denial-of-service attacks,
4. RCB is insecure when one key is used for full-duplex communication,
5. In many practical side-channel settings, RCB fails to provide meaningful privacy, also
6. RCB fails to provide secure Authenticated Encryption. More precisely, RCB is vulnerable to forgery attacks. I.e., RCB fails at providing INT-CTXT security.

We stress that none of the issues 2–6 applies to OCB, when used with random nonces. Issue 5 is a symptom of a general problem for block cipher based leakage-resilient cryptography, which we will discuss later. Issue 2, is an intentional design decision from [2]. We demonstrate the other issues by presenting attacks. All our attacks are in the black-box model. I.e., though RCB has been designed to withstand side-channel attacks, our attacks don't even require a side-channel.

We agree with [2] that there is an urgent need for leakage-resilient AE schemes, and issues 1–4 might be an acceptable trade-off for a leakage-resilient scheme. Issue 5 is not specific for RCB, but a general problem for block-cipher based leakage-resilient privacy. But issue 6, the lack of secure authentication, is damning for any AE scheme – with or without leakage-resilience.

Outlook. We define some related preliminary and security notions in Section 2. Section 3 provides an overview over the RCB scheme, and Section 4 describes our attacks. Section 5 discusses the privacy of RCB under leakage, and at the end in Section 6 we conclude our paper.

2 Preliminaries and Security Notions

In this section, we define some related security notions that we use for our attack.

[1] The authors of OCB did never claim robustness, but [2] made such claims for RCB.

Definition 1 (Authenticated Encryption (AE)). *A nonce-based authenticated encryption (AE) scheme is a tuple $\Pi = (\mathcal{E}, \mathcal{D})$ of a deterministic encryption algorithm $\mathcal{E} : \mathcal{K} \times \mathcal{N} \times \mathcal{H} \times \mathcal{M} \to \mathcal{C} \times \mathcal{T}$, and a deterministic decryption algorithm $\mathcal{D} : \mathcal{K} \times \mathcal{N} \times \mathcal{H} \times \mathcal{C} \times \mathcal{T} \to \mathcal{M} \cup \{\perp\}$, with associated non-empty key space $\mathcal{K}$, non-empty nonce space $\mathcal{N}$, and $\mathcal{H}$, $\mathcal{M}$, $\mathcal{C} \subseteq \{0,1\}^*$ denote the header, message, and ciphertext, respectively. We define a tag space $\mathcal{T} = \{0,1\}^\tau$ for a fixed $\tau \geq 0$. If a given tuple (N, H, C, T) is valid, $\mathcal{D}_K^{N,H}(C, T)$ returns the corresponding plaintext M, and $\perp$ otherwise.*

Correctness and tidiness of the authenticated encryption schemes are as follows:

- *Correctness*: if $\mathcal{E}_K^{N,H}(M) = (C, T)$, then $\mathcal{D}_K^{N,H}(C, T) = M$.
- *Tidiness*: if $\mathcal{D}_K^{N,H}(C, T) = M \neq \perp$, then $\mathcal{E}_K^{N,H}(M) = (C, T)$, $\forall C \in \mathcal{C}, T \in \mathcal{T}$.

We require $\mathcal{M}$ to contain at least two strings; if $\mathcal{M}$ contains a string of length m, it contains all strings of length m, the same condition applies to the $\mathcal{H}$. When $\mathsf{Enc}_K(N, H, M)$ is a string, then its length $l(|N|, |H|, |M|)$ depends only on $|N|, |H|$ and $|M|$. We require Enc and Dec to be the inverse of one of each other.

Now we define a security notions in the black-box model:

Definition 2 (IND-CPA Security). *Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an AE scheme as defined in Definition 1. Let $\mathcal{A}$ be a computationally bounded adversary. Then, the IND-CPA advantage of $\mathcal{A}$ is defined as*

$$\mathbf{Adv}_{\Pi}^{\mathsf{IND\text{-}CPA}}(\mathcal{A}) = \left| \Pr\left[\mathcal{A}^{\mathcal{E}_{\mathcal{K}}(\cdot,\cdot)} \Rightarrow 1\right] - \Pr\left[\mathcal{A}^{\$(\cdot,\cdot)} \Rightarrow 1\right] \right|,$$

where the probabilities are taken over $K \twoheadleftarrow K$ and the random coins of $\mathcal{A}$. Further, we define $\mathbf{Adv}_{\Pi}^{\mathsf{IND\text{-}CPA}}(q, \ell, t)$ as the maximum advantage over all IND-CPA adversaries $\mathcal{A}$ on Π that run in time at most t, and make at most q queries of total length ℓ to the available oracles.

Definition 3 (INT-CTXT Security). *Let $\Pi = (\mathcal{E}, \mathcal{D})$ be a nonce-based AE scheme, $K \twoheadleftarrow K$, and $\mathcal{A}$ a computationally bounded adversary with access to two encryption and decryption oracles such that $\mathcal{A}$ never queries encryption before decryption. Then, the* INT-CTXT *advantage of $\mathcal{A}$ wrt. Π is defined as*

$$\mathbf{Adv}_{\Pi}^{\text{INT-CTXT}}(\mathcal{A}) := \Pr\left[\mathcal{A}^{\mathcal{E}_K, \mathcal{D}_K} \text{ forges}\right] = \underset{\mathcal{A}}{\Delta}\left(\mathcal{E}_K, \mathcal{D}_K; \mathcal{E}_K, \perp\right),$$

where "forges" means that $\mathcal{D}_K$ returns anything other than $\perp$ for a query of $\mathcal{A}$.

Now let $L : \{0,1\}^* \to \{0,1\}^\lambda$ be the *leakage function*. This function takes the secret key and the message as inputs and outputs $f_K^L(M)$. What is the right leakage function is still an open problem and not every function should be allowed (e.g. $f_K^L(M) = K$). Anyway, since our attacks are in the black-box model we do not care about $f_K^L(M)$.

3 Genral Overview of RCB

RCB [2] is a new leakage resilient authenticated encryption scheme. It uses a block cipher E, which may leak, and a re-keying scheme g, which is assumed not to leak. I.e., given a public ctr, making many calls to $g_{K^*}(\cdot)$ to compute $g_{K^*}(\text{ctr})$, $g_{K^*}(\text{ctr}+1)$, ... are assumed not to leak the secret long-term key K^*.

To make sure both sender and receiver use a single secret key, both parties need to maintain a shared ctr value. The authors of RCB describe how to resynchronise ctr, if necessary. At the set-up phase, both the sender and the receiver initialize the ctr value to 1.

For the encryption, RCB encrypts a message $M = (m_1, \ldots, m_L)$, where $m_1, \ldots, m_{L-1}$ are b-bit blocks, and the size of m_L is at most b bits, into a ciphertext $C = (c_1, \ldots, c_L)$, with $|c_i| = |m_i|$, and authentication tag $T \in \{0,1\}^\tau$ for some tag size $\tau \leq b$. The value ctr' of the internal counter before the start of the encryption is also part of the output. See Algorithm 1. For the decryption, given ctr', C and T, it first computes M, then its own authentication tag T', and returns M if $T' = T$, else it returns $\perp$.

Algorithm 1 RCB encryption.

1: **state** long-term key K^*, counter ctr ($*$ K^* is constant, ctr always increases $*$)
2: **input** message $M = (m_1, \ldots, m_L)$
3: $\text{ctr}' \leftarrow \text{ctr}$
4: **for** $i \in \{1, \ldots, L-1\}$ **do**
5: $\quad K_i \leftarrow g_{K^*}(\text{ctr})$
6: $\quad \text{ctr} \leftarrow \text{ctr} + 1$
7: $\quad c_i \leftarrow E_{K_i}(m_i)$
8: $\text{ctr} \leftarrow \text{ctr} + 1$ ($*$ skip one value $*$)
9: $K_L \leftarrow g_{K^*}(\text{ctr})$
10: $\text{ctr} \leftarrow \text{ctr} + 1$
11: $X \leftarrow \text{len}(m_L) \oplus \text{ctr}'$
12: $Y \leftarrow E_{K_L}(X)$
13: $c_L \leftarrow Y \oplus m_L$
14: $S \leftarrow m_1 \oplus \cdots \oplus m_{L-1} \oplus (c_L 0^{b\tau}) \oplus Y$ ($*$ checksum $*$)
15: $K_{L+1} \leftarrow g_{K^*}(\text{ctr})$
16: $\text{ctr} \leftarrow \text{ctr} + 1$
17: $T \leftarrow E_{K_{L+1}}(S)[\text{first } \tau \text{ bits}]$
18: **return** $(\text{ctr}', \overbrace{(c_1, \cdots, c_L)}^{C}, T)$

Message blocks M_i, ..., M_{L-1} are full b-bit blocks and encrypted to the b-bit blocks C_i, ..., C_{L-1} in lines 4–7. Lines 8–13 deal with the encryption of the last block M_L, which may be smaller than b bits, to C_L. The authentication tag is computed in 14–17.

Security Claims for RCB [2] For the purpose of this paper, we do not need to introduce formal security definitions, or to copy the security definitions from [2]. Regarding privacy, we assume the adversary attempts to choose two messages M_0 and M_1 of the same length, then a challenger encrypts M_b for a random $b \in \{0,1\}$. The adversary succeeds if it finds b. For correctness, we assume any message honestly encrypted by the sender will be decrypted to the same message by the receiver. For authenticity, the adversary tries to create a triple (ctr, C, T) of counter, ciphertext and authentication tag, which the challenger will accept. The constraint is, that this triple has not been produced by the challenger, before.

The main security claims for RCB are as follow:

- privacy under chosen plaintext attacks with leakage, and
- authenticity against chosen-message existential forgery attacks under leakage.

There are other security claims for RCB which are related to robustness, namely:

- resistance to nonce-misuse, and
- security under the release of unverified plaintexts (RUP).

The adversary is always free to ignore the side-channel information from the leakage, and our attacks actually don't require such side-channel information. In other words, RCB is not just insecure against side-channel attacks, RCB is even insecure against ordinary black-box attacks.

4 Attacks on RCB

Below we assume Alice uses RCB to send authenticated and encrypted messages to the receiver Bob. The adversary $\mathcal{A}$ is trying to attack Alice and Bob.

4.1 Attacks on Robustness

Misuse-resistance : RCB is resistant to nonce-misuse because "it does not have the nonce requirement" [2, Section 6]. This is absolutely wrong. Firstly RCB has the requirement to use an always incremental counter. While not every nonce is a counter, such a counter *is* a nonce. I.e., the counter requirement logically implies the nonce requirement. Secondly, if counters repeat (i.e., if nonces are misused), then there is a simple attack on RCB as follow:

1. $\mathcal{A}$ sets $Z := \mathrm{ctr}$ (Alice's current ctr), and chooses $m_1 \neq m_2 \in \{0,1\}^b$.
2. Alice encrypts (m_1, m_2) to $(Z, (c_1, c_2), T)$.
3. $\mathcal{A}$ resets Alice's counter to $\mathrm{ctr} := Z$ (nonce-reuse!)
4. $\mathcal{A}$ chooses messages $M_0 = (m_1, m_1)$ and $M_1 = (m_2, m_2)$.
5. Alice encrypts M_b to $(Z, (c_1', c_2'), T')$ for random $b \in \{0,1\}$.
   (Due to the nonce reuse, the value Z is the same as in step 2.)
6. $\mathcal{A}$ computes b: If $c_1' = c_1$ then $b = 0$, else $c_2' = c_2$ and $b = 1$.

RUP-secure : RCB is claimed to be plaintext-aware in a RUP-setting [2, Section 6], without giving any reasons. In fact, as the following attack shows the claim is wrong:

1. $\mathcal{A}$ sets $Z := \text{ctr}$ (Alice's current ctr).
2. $\mathcal{A}$ chooses $m_1 \neq m_2$, $M_0 = (m_1, m_2)$ and $M_1 = (m_2, m_1)$.
3. Alice encrypts M_b to $(Z, (c_1, c_2), T)$ for $b \in \{0, 1\}$.
4. $\mathcal{A}$ chooses $c_2' \neq c_2$.
5. Bob performs unauthenticated decryption of $(Z, (c_1, c_2'), T)$ to (m_1', m_2').
6. $\mathcal{A}$ computes b: If $m_1' = m_1$ then $b = 0$, else $m_1' = m_2$ and $b = 1$.

4.2 A Denial-of-Service (DoS) Attack

In general, an encryption scheme is *correct*, if decrypting a ciphertext, which has been generated by properly encrypting a message M, will recover M again. RCB is only correct in a slightly weaker sense. By tampering with the counter, the attacker can deny the service, such that Bob will reject a valid ciphertext. Our first DoS attack works as follows.

1. Alice initialize her counter to $Z = \text{ctr}$.
2. Alice encrypts a message M to (Z, C, T).
   Alice's counter is now $\text{ctr} = Z + a$ for some $a > 0$.
3. Alice encrypts another message M' to $(Z + a, C', T')$.
   Alice's counter is now $\text{ctr} = Z + a + b$ for some $b > 0$.
4. $\mathcal{A}$ forwards $(Z + a, C', T')$ to Bob.
5. If a does not exceed a pre-defined threshold,[2] then Bob decrypts $(Z + a, C', T')$ to M'. Bob's new counter is now $Z + a + b$.
6. $\mathcal{A}$ forwards (Z, C, T) to Bob. Because $X < \text{ctr} + a + b$, Bob decrypts (Z, C, T) to $\perp$, not to M.

Our second DoS attack does not even require Alice to encrypt two messages:

1. Alice initialize her counter to $Z = \text{ctr}$.
2. Alice encrypts M to (Z, C, T). Alice's new counter is $Z + a$.
3. $\mathcal{A}$ chooses C' and T' and sends (Z, C', T') to Bob.
4. Bob decrypts (Z, C', T') to $\perp$. Bob's new counter is $X + b$.[3]
5. $\mathcal{A}$ forwards (Z, C, T) to Bob. Because $X < X + b$, Bob decrypts (X, C, T) to $\perp$, not to M.

[2] Else, Alice and Bob would perform interactive resynchronisation [2, Figure 2].

[3] Bob must increase the counter, even if the message turns out to be invalid. Otherwise, Bob would use the same internal key more than once, thus destroying the main purpose of using RCB, namely its claimed leakage-resilience.

4.3 Attack on Full-Duplex Communication

From time to time, Bob may also respond to Alice's messages. If Alice sends a message to Bob using a key K, and Bob sends a message to Alice, using the same K, then $\mathcal{A}$ can exploit the following:

1. Alice and Bob share the same initial counter $Z = \text{ctr}$.
2. $\mathcal{A}$ chooses $m_1 \neq m_2$.
3. Alice encrypts (m_1, m_2) to $(Z, (c_1, c_2), T)$. Bob does not see this message.
4. $\mathcal{A}$ chooses $M_0 = (m_1, m_1)$ and $M_1 = (m_2, m_2)$.
5. Now it is Bob's turn. Bob encrypts M_b to $(Z, (c'_1, c'_2), T')$ for $b \in \{0, 1\}$.
6. $\mathcal{A}$ computes b: If $c_1 = c'_1$, then $b = 0$. Else $b = 1$ and $c_2 = c'_2$.

It is easy to evade this attack by using two independent (key, counter)-pairs, one for messages from Alice to Bob, and the other one for messages from Bob to Alice. On the other hand, a random-nonce instantiation of OCB does not need synchronised counters and allows the same key to be used for both directions.

4.4 Forgery Attack

The idea of the attack is to use one valid ciphertext to produce another valid ciphertext. In order to do this, we need to prevent Bob from receiving this ciphertext with the aim that the counter will not change.

Recall that E is b-bit block cipher. We define $\text{trunc}_t : \{0,1\}^b \longmapsto \{0,1\}^t$ for $t \leq b$ by dropping the last $t - b$ bits of the b-bit input. Let K_i be the ephemeral key used to cipher the i-th block. The attack asks first for the authentication of a q-block message and then forges a $q - 3$-block message. So the attack is done in the following steps:

1. Initially Alice and Bob share the same counter $Z = \text{ctr}$.
2. $\mathcal{A}$ chooses $q \geq 4$.
3. $\mathcal{A}$ chooses arbitrary $m_1, \ldots, m_{q-3}$ in $\{0,1\}^b$.
4. $\mathcal{A}$ chooses arbitrary $m'_{q-3} \in \{0,1\}^b$.
5. $\mathcal{A}$ chooses $m_{q-2} = len(m'_{q-3}) \oplus Z$.
6. $\mathcal{A}$ computes $m_{q-1} = \left(\bigoplus_{i=1}^{q-4} m_i\right) \oplus m'_{q-3}$.
7. $\mathcal{A}$ chooses arbitrary m_q in $\{0,1\}^b$ (any string of at most b bits).
8. $\mathcal{A}$ asks for the encryption of $(m_1, ..., m_q)$.
9. Alice encrypts this to $(Z, (c_1, ..., c_q), T)$.
10. $\mathcal{A}$ sets c'_{q-3} to $c_{q-2} \oplus m'_{q-3}$.
11. $\mathcal{A}$ sends $(Z, (c_1, ..., c_{q-4}, c'_{q-3}), \text{trunc}_t(c_{q-1}))$ to Bob.

We argue that $(Z, (c_1, ..., c_{q-4}, c'_{q-3}), \text{trunc}_t(c_{q-1}))$ is accepted by Bob. More precisely, it is the legitimate encryption of the message $(m_1, \ldots, m_{q-4}, m'_{q-3})$ with initial counter Z.

Firstly, it is easy to see that message blocks $m_1, \ldots, m_{q-4}$ are encrypted to $c_1, \ldots, c_{q-4}$.

Secondly, the block c_{q-2} is the encryption of $m_{q-2} = (\mathrm{len}(m'_{q-3}) \oplus Z)$ under the key $g_{K^*}(X + q - 2)$. The ciphertext block c'_{q-3} has been chosen as $c_{q-2} \oplus m'_{q-3}$, as required by lines 8–13 of Algorithm 1.

Thirdly, as can be seen, m_{q-1} has been chosen as the checksum S for the message $(m_1, \ldots, m_{q-4}, m'_{q-3})$, and the tag is the encryption of S under the key $g_{K^*}(X + q - 1)$.

Finally, note that though our attack assumes the forged message block m'_{q-3} to be a b-bit block, the attack works probabilistically even if m'_{q-3} is a $(b - \delta)$-bit block, namely, if the last δ bits of C_{q-2} are zero. I.e., the attack then succeeds with probability $2^{-\delta}$.

5 Privacy by RCB

In many practical leakage settings, RCB even fails to provide privacy. Actually, as we mentioned before, this does not contradict the security claims made in [2], but it is related to a more general problem for block-cipher based authenticated encryption.

If Alice encrypts either of two same-length messages M_0 and M_1, and the adversary can decide which message has been encrypted, then privacy, as understood by most cryptographers, is gone. This is not much different in [2].

Now recall line 7 in Algorithm 1:

$$c_i \leftarrow E_{K_i}(m_i).$$

The message block m_i is part of the input to the block cipher. If information about the m_i leaks, then the privacy of RCB will fail.

As it turns out, typical side-channels allow the adversary to gather information about m_i (and also c_i). See Figure 1 for a power analysis trail. The "peaks" may, e.g., reveal information about the Hamming weight of the data currently processed, including the Hamming weight of m_i at the beginning and the Hamming weight of c_i at the end.

We stress that this observation does not contradict the security claims made in [2]. Implicitly, RCB assumes that information about K_i may leak, but information about m_i must not leak. More explicitly, but actually a little bit subtle for the casual reader, the privacy claim in [2] relates the privacy of RCB to the privacy of the block cipher implementation. If a side-channel, such as the one in Figure 1, leaks information about m_i, the privacy of the block cipher implementation is low, and then the privacy claimed for RCB becomes negligible.

But at the end of the day, the user who expects decent security against side-channel attacks may be disappointed from RCB. In practice, many side-channels reveal information about the block cipher's plaintexts.

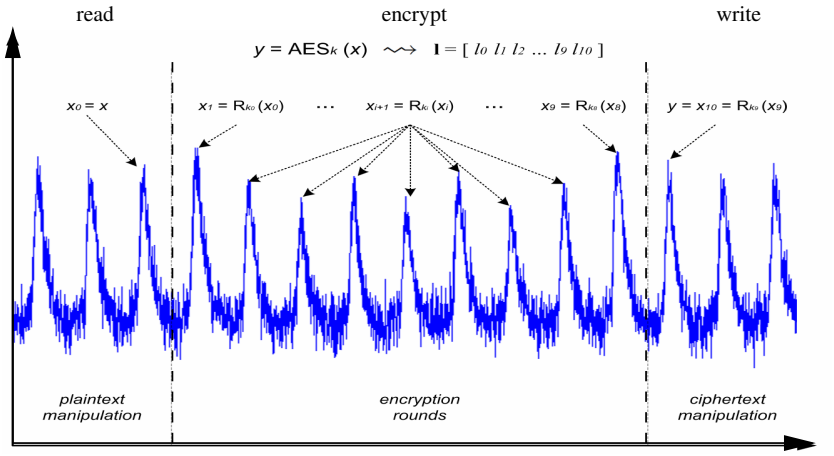

Fig. 1: A leakage trail for a block cipher encryption, computing $c_i = E_{K_i}(m_i)$ [16].

We do not consider this as an specific RCB problem. Maintaining good privacy for block-cipher based leakage-resilient cryptosystems seems to be an unsolved issue, if every block cipher evaluation to compute $Y = E_K(X)$ leaks information about both X and Y.

In fact, as done in [12] the best achievable privacy with leakage presence is to reduce the security of multiple iterations to the security of a single iteration. That is, that whatever the adversary is able to do against multiple iterations of a scheme, he is able to do against a single iteration.

6 Conclusion

What went wrong? And can one repair RCB? RCB has been derived from the well-established OCB mode. OCB is neither leakage-resilient nor robust, but it provides secure authenticated encryption in the black-box model (without leakage). The fact that RCB, as a modified OCB even fails in the black-box model is, at least, surprising.

[2] list the following modifications they made to turn OCB into RCB:

1. There is no masking for the input and output of the block cipher in the re-keying scheme. (Instead, the keys change.)
2. The starting counter is XORed to the input of block cipher during processing the last block of the message, to prevent adversary to create a valid pair of message and a tag. (Note that the starting counter is not secret. OCB uses a secret mask derived from the key at this point.)
3. One fresh key is omitted before processing the last message block, with the aim of thwarting forgery attack by the adversary.

The second modification clearly weakens RCB, in contrast to OCB. It seems that [2] actually discovered this issue and thus applied the third modification. As our forgery attack shows, the third modification is a sufficient countermeasure to solve this issue.

To thwart this attack, one could propose a modification of the original OCB which is more in the spirit of the original OCB. E.g., RCB-modified could use $K_i = g_{K^*}(X + 2i)$ as the ephemeral keys to encrypt the first $L - 1$ message blocks and the checksum. The ephemeral key for the final block m_L could be $g_{K^*}(X + 2L + 1)$.

We conjecture that this would defend black-box forgery attacks such as ours. But it would not solve any of the other issues.

Summary. This work described several attacks on RCB leakage resilient authenticated encryption scheme. RCB is not robust, neither against nonce misuse, nor against release of unverified plaintexts. The requirement to maintain synchronised counters between sender and receiver opens the door to Denial-of-Service attacks. RCB cannot securely be used for full-duplex communication. For many practical side-channel scenarios, RCB does not provide meaningful privacy. But the most severe issue is a forgery attack in the black-box model. It shows that RCB fails to match a core requirement for any authenticated encryption scheme, even without leakage.

In spite of these negative results, we appreciate the authors of [2] to study leakage-resilient authenticated encryption at all. To the best of our knowledge, they where the first to study this problem.

Acknowledgments. Farzaneh Abed was supported by the Simple Scry project with Cisco, and Francesco Berti by the INNOVIRIS project SCAUT.

References

1. Farzaneh Abed, Scott Fluhrer, John Foley, Christian Forler, Eik List, Stefan Lucks, David McGrew, and Jakob Wenzel. Pipelineable online encryption (poe). In *21st International Workshop on Fast Software Encryption (FSE 2014)*, Lecture Notes in Computer Science (LNCS), 3 2014.
2. Megha Agrawal, Tarun Kumar Bansal, Donghoon Chang, Amit Kumar Chauhan, Seokhie Hong, Jinkeon Kang, and Somitra Kumar Sanadhya. Rcb: leakage-resilient authenticated encryption via re-keying. *The Journal of Supercomputing*, pages 1–26, 2016.
3. Elena Andreeva, Andrey Bogdanov, Atul Luykx, Bart Mennink, Nicky Mouha, and Kan Yasuda. How to Securely Release Unverified Plaintext in Authenticated Encryption. In *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I*, pages 105–125, 2014.
4. Francesco Berti, François Koeune, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. Leakage resilient cryptography in practice. *IACR Cryptology ePrint Archive*, 2016:996, 2016.

5. Christoph Dobraunig, Maria Eichlseder, Stefan Mangard, Florian Mendel, and Thomas Unterluggauer. Isap – authenticated encryption inherently secure against passive side-channel attacks. *IACR Cryptology ePrint Archive*, 2016:952, 2016.
6. Stefan Dziembowski and Krzysztof Pietrzak. Leakage-resilient cryptography. In *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008, October 25-28, 2008, Philadelphia, PA, USA*, pages 293–302. IEEE Computer Society, 2008.
7. Sebastian Faust, Krzysztof Pietrzak, and Joachim Schipper. Practical leakage-resilient symmetric cryptography. In Emmanuel Prouff and Patrick Schaumont, editors, *Cryptographic Hardware and Embedded Systems - CHES 2012 - 14th International Workshop, Leuven, Belgium, September 9-12, 2012. Proceedings*, volume 7428 of *Lecture Notes in Computer Science*, pages 213–232. Springer, 2012.
8. Ted Krovetz and Phillip Rogaway. The software performance of authenticated-encryption modes. In Antoine Joux, editor, *Fast Software Encryption - 18th International Workshop, FSE 2011, Lyngby, Denmark, February 13-16, 2011, Revised Selected Papers*, volume 6733 of *Lecture Notes in Computer Science*, pages 306–327. Springer, 2011.
9. Stefan Mangard. Hardware countermeasures against dpa–a statistical analysis of their effectiveness. In *Cryptographers' Track at the RSA Conference*, pages 222–235. Springer, 2004.
10. Marcel Medwed, Christophe Petit, Francesco Regazzoni, Mathieu Renauld, and François-Xavier Standaert. Fresh re-keying II: securing multiple parties against side-channel and fault attacks. In Emmanuel Prouff, editor, *Smart Card Research and Advanced Applications - 10th IFIP WG 8.8/11.2 International Conference, CARDIS 2011, Leuven, Belgium, September 14-16, 2011, Revised Selected Papers*, volume 7079 of *Lecture Notes in Computer Science*, pages 115–132. Springer, 2011.
11. Silvio Micali and Leonid Reyzin. Physically observable cryptography (extended abstract). In Moni Naor, editor, *Theory of Cryptography, First Theory of Cryptography Conference, TCC 2004, Cambridge, MA, USA, February 19-21, 2004, Proceedings*, volume 2951 of *Lecture Notes in Computer Science*, pages 278–296. Springer, 2004.
12. Olivier Pereira, François-Xavier Standaert, and Srinivas Vivek. Leakage-resilient authentication and encryption from symmetric cryptographic primitives. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-6, 2015*, pages 96–108. ACM, 2015.
13. Krzysztof Pietrzak. A leakage-resilient mode of operation. In Antoine Joux, editor, *Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings*, volume 5479 of *Lecture Notes in Computer Science*, pages 462–482. Springer, 2009.
14. Matthieu Rivain and Emmanuel Prouff. Provably secure higher-order masking of aes. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 413–427. Springer, 2010.
15. Joachim H. Schipper. Leakage Resilient Authentication. Master's thesis, Utrecht University, the Netherlands, 2010.
16. F.-X. Standaert. Leakage resilient cryptography (a practical overview). Slides from SKEW 2011, 2011. `http://skew2011.mat.dtu.dk/slides/LRC_practical_overview.pdf`.

17. François-Xavier Standaert, Olivier Pereira, Yu Yu, Jean-Jacques Quisquater, Moti Yung, and Elisabeth Oswald. Leakage resilient cryptography in practice. *IACR Cryptology ePrint Archive*, 2009:341, 2009.
18. François-Xavier Standaert, Olivier Pereira, Yu Yu, Jean-Jacques Quisquater, Moti Yung, and Elisabeth Oswald. Leakage resilient cryptography in practice. In Ahmad-Reza Sadeghi and David Naccache, editors, *Towards Hardware-Intrinsic Security - Foundations and Practice*, Information Security and Cryptography, pages 99–134. Springer, 2010.
19. Nicolas Veyrat-Charvillon, Marcel Medwed, Stéphanie Kerckhof, and François-Xavier Standaert. Shuffling against side-channel attacks: A comprehensive study with cautionary note. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 740–757. Springer, 2012.
20. Yu Yu and François-Xavier Standaert. Practical leakage-resilient pseudorandom objects with minimum public randomness. In Ed Dawson, editor, *Topics in Cryptology - CT-RSA 2013 - The Cryptographers' Track at the RSA Conference 2013, San Francisco,CA, USA, February 25-March 1, 2013. Proceedings*, volume 7779 of *Lecture Notes in Computer Science*, pages 223–238. Springer, 2013.
21. Yu Yu, François-Xavier Standaert, Olivier Pereira, and Moti Yung. Practical leakage-resilient pseudorandom generators. In Ehab Al-Shaer, Angelos D. Keromytis, and Vitaly Shmatikov, editors, *Proceedings of the 17th ACM Conference on Computer and Communications Security, CCS 2010, Chicago, Illinois, USA, October 4-8, 2010*, pages 141–151. ACM, 2010.